Efficient Multiplication over Extension Fields

Nadia EL MRABET[1] and Nicolas GAMA[2]

[1] LIASD - Université Paris 8
[2] Université de Versailles - PRISM - CNRS

Abstract. The efficiency of cryptographic protocols rely on the speed of the underlying arithmetic and finite field computation. In the literature, several methods on how to improve the multiplication over extensions fields $\mathbb{F}_{q^m}$, for prime q were developped. These optimisations are often related to the Karatsuba and Toom Cook methods. However, the speeding-up is only interesting when m is a product of powers of 2 and 3. In general cases, a fast multiplication over $\mathbb{F}_{q^m}$ is implemented through the use of the naive school-book method. In this paper, we propose a new efficient multiplication over $\mathbb{F}_{q^m}$ for any power m. The multiplication relies on the notion of Adapted Modular Number System (AMNS), introduced in 2004 by [3]. We improve the construction of an AMNS basis and we provide a fast implementation of the multiplication over $\mathbb{F}_{q^m}$, which is faster than GMP and NTL.

1 Introduction

The efficiency of an algebraic cryptosystem is directly related to the speed of the underlying arithmetic computation over finite fields $\mathbb{F}_q$ and their extensions $\mathbb{F}_{q^m}$, wherein q is a prime number and m an integer. Depending on the mathematics required for the cryptosystem, the prime q is either 2, 3 or a large prime number. For cryptographic applications, the NIST provides recommendations for the appropriate size of q and m [2,1], for elliptic curves cryptography [16,7], for pairing based cryptography [5,6] or torus based cryptography [21,26].

For a large prime number q the arithmetic over $\mathbb{F}_{q^m}$ is rather expensive and several methods to improve it are described in the literature [27]. In characteristic 2, different optimizations of the multiplication are described in [14,25], when the field $\mathbb{F}_{2^m}$ is defined modulo a cyclotomic polynomial, or as the evaluations of polynomials over a root of unity. Usually, the finite field is embedded in a larger field using the cyclotomic polynomials, and the multiplication is then optimized via the Fast Fourier Transform. When the characteristic of the finite field is different from 2, classical optimizations depend on the value of m. For extensions of degree 2, the Karatsuba method is the most efficient [27], while the Toom-Cook method is recommended for extensions of degree 3 [27]. Consequently, when m is the product of a power of 2 with a power of 3 ($m = 2^i 3^j$), the multiplication is done by recursive applications of the Karatsuba and Toom Cook methods [4,9]. When m is a very large power of 2, the Fast Fourier Transformation (FFT) can be used to improve the multiplication over $\mathbb{F}_{q^{2^n}}$, with q being a prime and

n being an integer such that the number q^{2^n} has several thousand digits [27]. In [18], Montgomery proposed efficient multiplications for extensions of small degrees m =5, 6 and 7. The case of degree 5 has been recently improved in [19].

For other extension degrees m, very few improvements have been proposed.

In [10], El Mrabet and Nègre proposed an efficient multiplication for several values of m. This multiplication is based on the Discrete Fourier Transform (DFT) scheme [27]. One multiplication using the DFT scheme costs $(2m-1)$ multiplications in $\mathbb{F}_q$ and $\mathcal{O}(m^2)$ multiplications by a root of unity in $\mathbb{F}_q$. In a classical representation of the field $\mathbb{F}_q$, multiplication by a root of unity or by a random element of $\mathbb{F}_q$ would both have the same complexity. Consequently, the $\mathcal{O}(m^2)$ multiplications by a root of unity in $\mathbb{F}_q$ would be very expensive. In [10], the authors use an original representation of finite fields: the Adapted Modular Number System (AMNS) introduced by Bajard et al. in [3]. The main advantage of the AMNS representation is that all multiplications by a root of unity correspond to cyclic shifts, and become very cheap. This novel representation was designed in order to improve the multiplication over $\mathbb{F}_q$ when q is a pseudo Mersenne prime. In [8], a new class of specific moduli for cryptography was introduced, the more generalized Mersenne numbers. In [3], Bajard et all extend the work of [8] with the definition of the AMNS representation. The AMNS can in fact be applied for every prime q, and as well when q is a power of prime. Furthermore, the AMNS representation can be used to improve the arithmetic for the protocol RSA or more generally for a set $\mathbb{Z}/q\mathbb{Z}$.

We improve the result of [10] and we propose an efficient implementation of the multiplication for the extensions of field of prime degree. Our article is organized as follows. In Section 2 we recall the definition of the AMNS representation. In Section 2.2 we review the arithmetic and the multiplication in $\mathbb{F}_{q^m}$ with the AMNS representation, and we propose a simplified multiplication. In Section 3 we improve the multiplication in $\mathbb{F}_{q^m}$ by an efficient construction of an AMNS basis. In Section 4 we describe the first implementation of an AMNS representation. We use it in order to implement an efficient multiplication over $\mathbb{F}_{q^m}$, then we compare our results with the GMP implementation. We conclude in Section 5.

2 AMNS

The efficiency of the arithmetic in a field $\mathbb{F}_q$ relies on the representation of its elements. Usually the representation of a finite field is based on a positional number system representation. An element a of a field $\mathbb{F}_q$ is decomposed in the base β as $a = \sum_{i=0}^{\ell-1} a_i \beta^i$, with $a_i \in [0; \beta - 1]$. The element β is such that $\beta^l \geq q$. As an example, we can cite the binary decomposition (β =2), or the hexadecimal ($\beta = 8$) and decimal ($\beta = 10$) decomposition. The Adapted Modular Number System (AMNS) was introduced by Bajard et al. in [3]. Initially, this representation was described for pseudo Mersenne primes, but it was extended to any group $\mathbb{Z}/q\mathbb{Z}$.

2.1 Definition of an AMNS representation

The AMNS representation relies on the decomposition of numbers as a combination $\sum a_i \gamma^i$ of powers of a base γ, but with more drastic constraints on γ and on the size of the coefficients a_i. Namely, given an integer n, the base γ of an AMNS representation verifies that γ^n equals a very small number λ modulo q, for example $\lambda = \pm 1, \pm 2 \mod q$. The coefficients a_i in an AMNS representation are all bounded by $|a_i| \leq \rho$ for some parameter $\rho \approx q^{1/n}$. The AMNS representation is defined as follow.

Definition 1. *A basis $\mathcal{B}$ of an AMNS representation is a tuple $(q, n, \gamma, \rho, \lambda)$ such that: $\gamma^n = \lambda \mod q$; ρ is a covering radius such that for each element $a \in \mathbb{F}_q$, it exists a polynomial $\mathfrak{a}(X) = \sum_{i=0}^{n-1} a_i X^i$ of degree $< n$ and $\|\mathfrak{a}\|_\infty \leq \rho$ such that $\mathfrak{a}(\gamma) = a \mod q$.*

Then for each polynomial $\mathfrak{p} \in \mathbb{Z}_n[X]$, we say that $\mathfrak{p}$ is an AMNS representation *of $\mathfrak{p}(\gamma) \mod q$ when $\|\mathfrak{p}\|_\infty = \max_{i=1\ldots n} |p_i| \leq 2n\lambda\rho^2$ and that it is a reduced AMNS representation when $\|\mathfrak{p}\|_\infty \leq \rho$.*

Note that in general, the smallest covering radius ρ will be a small multiple of $q^{1/n}$. Also, a good basis for an AMNS representation should for efficiency be designed so that the largest entries fits into raw integer types (namely $2n\lambda\rho^2$ should be smaller than 2^{63} on a classical 64-bit processors).

Example 1. Let $q = 19$, an AMNS basis for $\mathbb{F}_q$ is $\mathcal{B} = (q = 19, n = 3, \gamma = 7, \rho = 2, \lambda = 1)$. Each element of $\mathbb{F}_q$ in signed representation is a polynomial in $\gamma = 7$, of degree at the most 2, and with coefficients in $[-1, 0, 1]$. We can check that $\gamma^3 \equiv 1 \mod q$. The decomposition of $\mathbb{F}_q$ is given in Table 1

Table 1. Decomposition of $\mathbb{F}_q$

1	2	3	4	5	6
1	$-X^2 - X + 1$	$X^2 - X - 1$	$X^2 - X$	$-X^2 - X + 1$	$X - 1$

7	8	9	10	11	12
X	$X + 1$	$-X^2 + 1$	$X^2 - 1$	X^2	X^2+1

13	14	15	16	17	18
$-X + 1$	$-X^2 + X - 1$	$-X^2 + X$	$-X^2 + X + 1$	$X^2 + X - 1$	-1

We could note that the evaluation of the polynomial $-1 - X + X^2$ in γ is $3 \mod q$ and that $\left\|-1 - X + X^2\right\|_\infty = 1 < 2$. Since $\|a_i\|_\infty < \rho$, it follows that the multiplication of two coefficients is efficient. This consideration leads to an efficient multiplication in AMNS systems where the prime q is a pseudo Mersenne number. For general primes, we will use a Montgomery's like multiplication.

Remark 1. When q is not a prime but is a prime power, the construction of an AMNS basis presented in [3] is not efficient. Indeed, we have to find an $n^{th}-$root of λ. Or, in a field $\mathbb{F}_{p^t}$ for p a prime and t an integer greater than 2; finding an $n^{th}-$root of λ is difficult because it is equivalent to find the factorisation of $q-1$, which is a difficult issue for a large value of q.

Remark 2. We could note that the AMNS representation of an integer is redundant. For example, 12 can be seen as X^2+1 or $-X$. If the protocol needs to compare two elements we can either go back to classical representation (we can use the Horner scheme to efficiently evaluate the polynomial.), or we can use Babai nearest plane algorithm to test whether the difference is in the lattice.

Both are indeed costly, but this cost can be amortized if the scheme requires a great number of aritmetic operations before being evaluated.

2.2 Arithmetic in AMNS representation

As the AMNS representation is quite original, we could wonder if the arithmetic in AMNS representation is more or less efficient than with a a classical representation. In [20], Nègre and Plantard present an efficient multiplication in the AMNS representation. This multiplication is constructed over Montgomery's scheme, and works for a prime q or for a prime power.

In order to give a simple explanation, we can say that the arithmetic in AMNS representation is based on these three primitives:

1. Addition and substraction of two elements $\mathfrak{a}$ and $\mathfrak{b}$ (denoted respectively by $\mathfrak{a}+\mathfrak{b}$ and $\mathfrak{a}-\mathfrak{b}$) is performed term by term on the AMNS representation, and the norm of the result is bounded by $\|\mathfrak{a}\|_\infty + \|\mathfrak{b}\|_\infty$.
2. Multiplication of $\mathfrak{a}$ and $\mathfrak{b}$ (denoted by $\mathfrak{a}\times\mathfrak{b}$) is performed by multiplying the two polynomials and reducing modulo $(X^n-\lambda)$. This leads to an element of norm at the most $n(\lambda+1)\|\mathfrak{a}\|_\infty \|\mathfrak{b}\|_\infty$.
3. Division by the constant parameter ϕ is used to reduce the size of coefficients. Given a representation $\mathfrak{a}$ of norm $|\mathfrak{a}|_\infty \leq 2n\lambda\rho^2$, Algorithm 1 outputs a representation of $\mathfrak{a}(\delta)\times\phi^{-1}$ of norm $\leq \rho$.

Algorithm 1 Division by ϕ in AMNS

Require: An AMNS basis $\mathcal{B}$, a polynomial $\mathfrak{m}$ and its inverse $\mathfrak{m}'$ mod ϕ, and an element $\mathfrak{a}$ of norm $\leq 2n\lambda\rho^2$.
Ensure: A reduced AMNS representation of $\mathfrak{a}(\gamma)\times\phi^{-1}$ of norm $\leq \rho$.
1: Compute $\mathfrak{a}' = \mathfrak{a}\times\mathfrak{m}' \mod \phi$ having all its coefficients in $[-\phi/2, \phi/2]$.
2: **Return** $\frac{1}{\phi}(\mathfrak{a}-\mathfrak{a}'\times\mathfrak{m})$ ▷ the division is exact

By applying successively the last two primitives, one obtains an efficient Montgomery's like multiplication between any two reduced AMNS representations $\mathfrak{a}$ and $\mathfrak{b}$. Like in the classical Montgomery's multiplication, the result is not

exactly $\mathfrak{a} \times \mathfrak{b} \mod q$ but $\mathfrak{a} \times \mathfrak{b} \times \phi^{-1} \mod q$. Note that the whole process performs only three multiplications of polynomials with small coefficients modulo $(X^n - \lambda)$, which makes the AMNS arithmetic efficient. The Montgomery's like multiplication is resume in Algorithm 2.

Algorithm 2 Montgomery's like multiplication in AMNS

Require: $\mathfrak{a}$ and $\mathfrak{b}$ in an AMNS basis $\mathcal{B} = (q, n, \gamma, \rho, \lambda)$, $\phi \geq 2n\lambda\rho$, a polynomial $\mathfrak{m}$ and its inverse $\mathfrak{m}' \mod (X^n - \lambda) \mod \phi$.
Ensure: The element $\mathfrak{t} \in \mathcal{B}$ such that $\mathfrak{t}(\gamma) = \mathfrak{a}(\gamma) \times \mathfrak{b}(\gamma) \times \phi^{-1} \mod q$.
1: Compute $\mathfrak{c} = \mathfrak{a} \times \mathfrak{b} \mod (X^n - \lambda)$
2: Compute $\mathfrak{q} = \mathfrak{c} \times \mathfrak{m}' \mod (X^n - \lambda) \mod \phi$
3: **Return** $\mathfrak{t} = \frac{1}{\phi}(\mathfrak{c} + \mathfrak{m}') \mod (X^n - \lambda)$ ▷ the division is exact

The third primitive requires the preprocessing of a valid parameters ϕ and two polynomials $\mathfrak{m}$ and $\mathfrak{m}'$ which are inverse to each other modulo $(X^n - \lambda)$ mod ϕ.

Lemma 1. *If ϕ is larger than $4n\lambda\rho^2$ and the polynomials $\mathfrak{m}$ and $\mathfrak{m}'$ satisfy the conditions*

$$\begin{aligned} \mathfrak{m}(\gamma) &\equiv 0 \mod q, \\ \|\mathfrak{m}\|_\infty &\leq \rho/n, \\ \mathfrak{m} \times \mathfrak{m}' &\equiv 1 \mod (X^n - \lambda) \mod \phi, \end{aligned}$$

then for any input AMNS representation $\mathfrak{a}$ of norm $\leq 2n\lambda\rho^2$, Algorithm 1 outputs a reduced AMNS representation (of norm $\leq \rho$) of $\mathfrak{a}(\gamma)/\phi$.

Proof. The division by ϕ in the last line of Algorithm 1 is exact, because $\mathfrak{a}' \times \mathfrak{m} = \mathfrak{a} \times \mathfrak{m} \times \mathfrak{m}' \mod \phi$ which is by definition equal to $\mathfrak{a} \mod \phi$. When taken modulo q, the evaluation in γ of $\phi^{-1} \times (\mathfrak{a} - a' \times \mathfrak{m})$ is $\mathfrak{a}(\gamma)/\phi$ because $\mathfrak{m}(\gamma) \equiv 0 \mod q$. Finally, the norm of $\mathfrak{a}'$ is $\|\mathfrak{a}'\|_\infty = \phi/2$, so the norm of $\|\mathfrak{a}' \times \mathfrak{m}\|_\infty$ is smaller than $\phi\rho/2$, i.e. ($\|a' \times m\|_\infty \leq \phi\rho/2$) and $\left\|\frac{1}{\phi}(\mathfrak{a} - \mathfrak{a}' \times \mathfrak{m})\right\|_\infty \leq (2n\lambda\rho^2 + \phi\rho/2)/\phi \leq \rho$.

The coefficients of the polynomials are smaller than ρ and ϕ. The execution of the algorithm needs to perform two different reductions. One reduction is performed modulo the polynomial $(X^n - \lambda)$, and one is an integer reduction modulo ϕ. The polynomial $(X^n - \lambda)$ is sparse, thus the polynomial reduction consists essentially in additions and shifts.

In the same way as in [20], the integer ϕ must be larger than $2n\lambda\rho$, and $\mathfrak{m}$ must be a polynomial with very small coefficients, which admits γ as a root modulo $\mathbb{F}_q$ and which is invertible mod ϕ. Unfortunately, the algorithm proposed in [20] to generate $\mathfrak{m}$ requires to approximate the shortest vector problem in a n-dimensional lattice within a constant factor, which can be difficult in high dimension. Furthermore, since a lot of integer divisions by ϕ occur in the Montgomery's like multiplication algorithm, in practice it would be ideal to have ϕ

equal to a power of 2 in order to speed-up those divisions. But once again, the construction of [20] does not guarantee that ϕ can be even.

In the next sections, we propose a polynomial time construction of all parameters which is inspired from the key generation algorithm used by Gentry et al. for the fully Homomorphic Encryption scheme challenges in [13].

2.3 Efficient multiplication in $\mathbb{F}_{q^m}$ using AMNS

In [10], El Mrabet and Nègre use the AMNS representation of $\mathbb{F}_q$ to improve the multiplication in an extension $\mathbb{F}_{q^m}$. They combine the Discrete Fourier Transform (DFT) multiplication with the AMNS representation. The DFT multiplication needs several multiplications by roots of unity. In a classical representation, a root of unity can be any element of $\mathbb{F}_q$ and consequently a multiplication by roots of unity is equivalent to a random multiplication in $\mathbb{F}_q$. The advantage of the AMNS representation is that the element γ chosen to construct the base can be a root of unity.

The extension $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$ is defined as the quotient $\mathbb{F}_q[Y]/(P(Y)\mathbb{F}_q[Y])$, where $P(Y)$ is an irreducible polynomial of degree m over $\mathbb{F}_q$ and $\mathbb{F}_q[Y]$ represente the polynomial in Y and with coefficients in $\mathbb{F}_q$. An element of $\mathbb{F}_{q^m}$ is a polynomial in Y of degree smaller than m and with coefficients in $\mathbb{F}_q$,

$$\mathbb{F}_{q^m} = \{R(Y) \in \mathbb{F}_q[Y] \text{ such that } \deg(R(Y)) < m\}.$$

We resume the combination of the DFT multiplication of two polynomials $U(Y)$ and $V(Y)$ with the AMNS representation. We denote $W(Y) = U(\mathrm{Y})\times \mathrm{V(Y)}$.

Let l be an integer such that we can define α to be a $2l^{th}-$root of unity in $\mathbb{F}_q$, and let $\alpha_i = \alpha^i$ for $i = 0, \ldots, 2l-1$.

The DFT multiplication is the composition of three steps:

1. The evaluation of the polynomials U and V in the α_is,
2. The $2l$ multiplications $U(\alpha_i) \times V(\alpha_i)$, in order to find the evaluation in the α_is of $W(Y)$,
3. The interpolation of $W(Y) = U(Y) \times V(Y)$.

The evaluation and interpolation steps can be performed with a matrix vector product. The evaluation corresponds to the product $\Omega \times^t U$ and $\Omega \times^t V$, where $U = [u_0, u_1, \ldots, u_{l-1}]$ and $V = [v_0, v_1, \ldots, v_{l-1}]$ are the vectors of the coefficients of $U(Y)$ and $V(Y)$. The interpolation step is composed by the product $\Omega^{-1} \times^t W$, where Ω^{-1} is the matrix inverse of Ω.

The matrix Ω and Ω^{-1} are the following

$$\Omega = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \alpha & \alpha^2 & \ldots & \alpha^{l-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^i & (\alpha^i)^j & \ldots & (\alpha^i)^{(l-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{(l-1)} & \alpha^{2(l-1)} & \ldots & \alpha^{(l-1)(l-1)} \end{bmatrix}, \Omega^{-1} = \frac{1}{l} \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \alpha^{-1} & (\alpha^{-1})^2 & \ldots & (\alpha^{-1})^{(l-1)} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & (\alpha^{-i}) & (\alpha^{-i})^2 & \ldots & (\alpha^{-i})^{(l-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{-(l-1)} & \alpha^{-2(l-1)} & \ldots & \alpha^{-(l-1)(l-1)} \end{bmatrix}.$$

The complexity of the DFT multiplication is the sum of the complexity of 3 matrix vector products and the $2l$ products in $\mathbb{F}_q$. In an AMNS representation defined by $\mathcal{B} = (q, n, \gamma, \rho, \lambda)$, we can choose the parameters such that the matrix-vector products are composed only with shifts and additions in $\mathbb{F}_q$. Indeed the matrix-vector products are only composed with multiplications of the α_is with the coefficients of $U(X)$ and $V(X)$. If we choose $l = 2n$, γ and λ in the AMNS base $\mathcal{B}$ to be such that $\lambda = -1$ and then $\gamma^n \equiv -1 \mod q$. With this choice, γ is an $l-$root of unity in $\mathbb{F}_q$ and can be use to define the matrix Ω. As a consequence, the multiplications by powers of γ consist only in shifts and additions in $\mathbb{F}_q$, as explained in the following lemma.

Lemma 2. *Let u_i be the i^{th} coefficient of $U(Y)$, $u_i \in \mathbb{F}_q$ and u_i is decomposed in the AMNS basis $\mathcal{B} = (q, n, \gamma, \rho, \lambda)$. Let $\mathfrak{u}_i$ be a representation of u_i in $\mathcal{B}$. Then the product $u_i \times \gamma^j = \mathfrak{u}_i \times X^j$ and*

$$\mathfrak{u}_i \times \gamma^j = (\sum_{k=0}^{n-1} u_i^k Y^k) \times Y^j \mod (Y^n + 1) = \sum_{k=0}^{j-1} u_i^{n-j+k} X^k + \sum_{k=j}^{n-1} u_i^{k-j} X^k.$$

Proof. The proof consists in writing down the equation and use the fact that we work modulo the polynomial $(X^n + 1)$. □

Remark 3. We can notice that the Lemma 2 is writen for $\lambda = -1$, but it is very easy to obtain the same result for a more generic value of λ. The important point is that λ must be choosen such that the multiplication by λ are for free. For example, λ can be ± 1, or $\pm 2^d$ with d being a small integer.

An important consequence of Lemma 2 is the fact that a multiplication by a power of γ is a very easy operation, which preserves the norm. Indeed, multiplying a vector $\mathfrak{u}$ by a power of γ consists only in a permutation of the coefficients of $\mathfrak{u}$, which is only linear in the bit-length.

Lemma 3. *The use of the DFT multiplication on powers of γ in an AMNS representation leads to an efficient multiplication over an extension field.*

Indeed, the evaluation of a polynomial in AMNS representation in a power of γ consists in multiplications by power of γ and additions. A multiplication by a power of γ in the AMNS representation is a mere shift in the representation. As a consequence, the evaluation of a polynomial in power of γ costs only $O(n)$ additions instead of $O(n)$ multiplications. The dual operation of interpolating a function from its values on powers of γ is also easy. Indeed, both operations can be viewed as a multiplication by a Lagrange matrix containing only powers of γ.

Multiplication of two polynomials $\mathfrak{f}, \mathfrak{g}$ can therefore be performed in a DFT manner by evaluating $\mathfrak{f}$ and $\mathfrak{g}$, pairwise multiplying the evaluations, interpolating the result, and reducing modulo P. Overall, this requires $O(n)$ multiplications in $\mathbb{F}_q$ instead of $O(n \log n)$ or $O(n^2)$ for classical algorithms. □

The above multiplication algorithm has the drawback that a multiplication requires $4n$ evaluations of the input polynomial on the roots of unity. Instead, it is

preferable to directly represent a polynomial $P(Y)$ in a Lagrange representation, by its evaluations $(P(\gamma^0), \ldots, P(\gamma^{2m-1}))$ rather than by its coefficients. Once again, the multiplication we propose is a Montgomery like algorithm, because the result of the product of $U(Y)$ times $V(Y)$ is $(m/\phi^3) \times UV$. The multiplication in Lagrange representation is described in Algorithm 3.

Algorithm 3 AMNS-Multiplication in $\mathbb{F}_{q^m}$

Require: Two reduced Lagrange-AMNS representations $\big(U(\gamma^0), \ldots, U(\gamma^{2m-1})\big)$ and $\big(V(\gamma^0), \ldots, V(\gamma^{2m-1})\big)$ of polynomials $U, V \in \mathbb{F}_q[Y]$ of degree $\leq m-1$.
Ensure: The reduced Lagrange-AMNS representation $(W(\gamma^0), \ldots, W(\gamma^{2m-1}))$ of the product $W \equiv (m/\phi^3).UV \mod Y^m - \alpha$ of degree $\leq m-1$
1: Compute $A = UV \times \phi^{-1} = (U(\gamma^i) \times V(\gamma^i) \times \phi^{-1})_{i=0,\ldots,2m-1}$ using the AMNS Montgomery's product
2: Compute the coefficients of $(\mathfrak{a}_0, \ldots, \mathfrak{a}_{2m-1})$ of $m \cdot A[Y] = \sum_{i=0}^{2m-1} \mathfrak{a}_i Y^i$ in AMNS representation using the inverse DFT
3: Reduce the polynomial modulo $Y^m - \alpha$ and divide it by ϕ to obtain $B = \sum_{i=0}^{m-1} \mathfrak{b}_i Y^i$ with $\mathfrak{b}_i = (\mathfrak{a}_i + \alpha \mathfrak{a}_{i+m}) \times \phi^{-1}$.
4: **Return** $B \times \phi^{-1}$ as $(B(\gamma^0) \times \phi^{-1}, \ldots, B(\gamma^{2m-1}) \times \phi^{-1})$ using DFT

If the integer n is a power of 2, the multiplication by the Fast Fourier Transform (FFT [27]) method can be an improvement of the DFT method. The FFT method is very interesting because it consists in factoring the computation in order to not to compute twice the same operation. In our case, this is very efficient, and the operations in the FFT method are only composed with multiplications by powers of γ. We recall here the major steps of the FFT multiplication.

Let $U(Y)$ be a polynomial that we want to evaluate for the DFT multiplication. The FFT method is a divide and conquer scheme. Let γ be a root of unity.

The FFT method consists in dividing the polynomial $U(Y)$ in two parts:

$$U_1 = \sum_{k=0}^{n/2-1} u_{2k} Y^{2k},$$
$$U_2 = \sum_{k=0}^{n/2-1} u_{2k+1} Y^{2k}, \text{ such that } U = U_1 + Y U_2.$$

We denote $\widehat{U} = [U(1), U(\gamma), \ldots, U(\gamma^{n-1})]$ and $\widehat{U_j} = [U_j(1), U_j(\gamma), \ldots, U_j(\gamma^{n-1})]$ for $j = 1, 2$. The element $\widehat{U}[i]$ is the i^{th} coefficient of $\widehat{U}$. The evaluation of U in a power of γ can be expressed as follow

$$\begin{aligned} &\text{for } i \in [0; n/2[\\ &\widehat{U}[i] = \widehat{U_1}[i] + \gamma^i \widehat{U_2}[i], \\ &\widehat{U}[i + n/2] = \widehat{U_1}[i] - \gamma^i \widehat{U_2}[i]. \end{aligned}$$

We use the fact that $\gamma^{i+n/2} = -\gamma^i$, which is evident since γ is a root of unity. Since n is a power of 2, when we apply the FFT method, we can recursively use this formula.

In [3], the AMNS representation was proposed randomly and the polynomial $\mathfrak{m}$ was not inversible for each construction. We propose below an efficient way to construct an AMNS representation and to assure the fact that the polynomial $\mathfrak{m}$ is invertible. We split the analyse in two parts. First, we analysis in Section 3.2, the simultaneous generation of $\mathfrak{m}$ and q, which can be done very efficiently even for large dimensions. Then in Section 3.3, we consider cases where q is fixed in advance, and the goal is to generate $\mathfrak{m}$ accordingsly. This second case, which occurs in pairing based cryptography, is much harder than the previous one, but can still be achieved in practice for extensions of degree ≈ 100. these cases occur for example in pairing based cryptography.

3 Theory

We propose a new construction of the AMNS parameters by adapting the key generation algorithm used by Gentry [13] in the fully homomorphic encryption scheme. Given as input an extension of degree n, the procedure generates a polynomial $\mathfrak{m}$ and a prime number q. This is for instance the case of almost all Elliptic Curve Cryptography scheme based on Diffie Hellman. However, this does not work for pairings, which have strong external constraints on q.

3.1 Some theory about lattices

The approach of [20] to generate AMNS parameters can be viewed as follow: given q a prime number, and γ such that $\gamma^n = \lambda \mod q$ is small, one construct the lattice $\mathcal{L}$ of all polynomials having the root γ modulo q.

$$\mathcal{L} = \{\mathfrak{a}(X) \in \mathbb{Z}[X], \text{ such that } \deg(\mathfrak{a}) < n \text{ and and } \mathfrak{a}(\gamma) = 0\}.$$

The polynomial $\mathfrak{m}$ must simply be a short vector of $\mathcal{L}$. However, it remains an open problem to efficiently construct this short vector for a large n. Once we identify the coefficients vectors $(a_0, \ldots, a_{n-1}) \in \mathbb{Z}^n$ with the corresponding polynomial $\mathfrak{a}(X) = \sum_{i=0}^{n-1} a_i X^i \in \mathbb{Z}[X]$, the lattice $\mathcal{L}$ is the set of all linear combinations of rows of its Hermite normal form basis $\mathcal{M}$.

$$\mathcal{M} = \begin{pmatrix} q & 0 & 0 & 0 & \dots & 0 \\ -\gamma & 1 & 0 & 0 & \dots & 0 \\ -\gamma^2 & 0 & 1 & 0 & \dots & 0 \\ \vdots & 0 & 0 & \ddots & 0 & \vdots \\ -\gamma^{n-2} & 0 & 0 & \vdots & 1 & 0 \\ -\gamma^{n-1} & 0 & 0 & \dots & 0 & 1 \end{pmatrix} \begin{matrix} \leftarrow q \\ \leftarrow X - \gamma \\ \leftarrow X^2 - \gamma^2 \\ \vdots \\ \leftarrow X^{n-2} - \gamma^{n-2} \\ \leftarrow X^{n-1} - \gamma^{n-1} \end{matrix}$$

The parameter $\mathfrak{m}$ which is used in Algorithm 1, and which is necessary for the Montgomery's multiplication in [20], must be a vector of $\mathcal{L}$ of norm $\|\mathfrak{u}\|_\infty \approx q^{1/n}$. On one hand, the existance of such short vector is guaranteed by a variant of Minkowski's theorem [17] for the $\|\cdot\|_\infty$. In [20], the authors experimentally used the LLL algorithm [15] in small dimension to reduce the lattice and produce a good $\mathfrak{m}$. In medium dimensions ($n \leq 115$), one could still use extreme pruning as described in [12] to find $\mathfrak{m}$. But for larger dimensions, no polynomial algorithm is known to produce such short vector: All known lattice reduction algorithm (either polynomial or practical) outputs vectors exponentially larger than the optimum in practice (namely, $\|\mathfrak{m}\|_2$ would be $\geq 1.01^n q^{1/n}$ see [11]).

Now suppose that we overcome this hardness and obtain a very short vector $\mathfrak{m}$ of $\mathcal{L}$, we still need to ensure that it is invertible $\mod \phi$. Note that when $\mathfrak{m}$ has a non-constant GCD with $X^n - \lambda$, this fails for all values of ϕ. Therefore, whenever the resultant of $\mathfrak{m}$ and $X^n - \lambda$ is even, ϕ cannot be chosen as a power of 2. Since Algorithm 1 involves a lot of exact divisions by ϕ, being able to take ϕ equal to a power of two would have a strong impact on the efficiency of the AMNS multiplication.

In the next subsection, we propose several solutions to overcome these problems, depending on how much freeness we have for the choice of the prime number q.

3.2 Construction when we can choose q

In ECC cryptography, for example the Diffie Helman protocol, we can freely choose the prime q, the only condition is that q must be large enough considering the security level we want to reach. We propose in that case an efficient way to construct an AMNS representation of the finite field $\mathbb{F}_q$. We adapt the key generation of the last challenges of Gentry's fully homomorphic cryptosystem, published in [13].

The method is a reversal construction of an AMNS representation of a finite field, in a sense that the lattice is built around a chosen short vector $\mathfrak{m}$, which satisfies the most favorable properties. The expensive lattice reduction step is not needed any more. Furthermore, we can ensure that the resultant of $\mathfrak{m}$ and $X^n - \lambda$ is odd, which enables us to set ϕ as a power of 2, and ensures that the AMNS multiplication in $\mathbb{F}_q$ is efficient.

The generation algorithm we propose in Algorithm 4 takes as input the dimension n and a boundary s on the expected norm of $\mathfrak{m}$. First, we choose $\lambda = -1$.

This ensures the existance of $2n$-th roots of unity, and most of all, this removes the negative impact of all $|\lambda|$ in the bounds of Lemmas 2 and 1. Then we pick a short vector $\mathfrak{m}$ at random, with coefficients of s bits, and test whether its resultant with X^n+1 is either a prime number, or contains a large prime factor q if one just needs to represent a ring $\mathbb{Z}/q\mathbb{Z}$ instead of a field. Note that in Gentry's challenges [13], the condition was more restrictive, since the resultant itself had to be prime. However, in all cases, it is only necessary to repeat a polynomial number of times the process in order to get a valid polynomial $\mathfrak{m}$ and its associated modulus q. All other parameters (ρ,ϕ, $\mathfrak{m}'$,γ) are easy to deduce:

Theorem 1. *Given a polynomial $\mathfrak{m}$ of odd resultant r with $X^n-\lambda$ and q a divisor of r, one deduces a valid AMNS representation basis $\mathcal{B}=(q,n,\gamma,\rho,\lambda)$ and additional parameters $\mathfrak{m},\mathfrak{m}'$ and ϕ needed by Algorithm 1 as follows*:

- $\rho = n\lambda \left\|m\right\|_\infty$ *is a valid covering radius,*
- ϕ *is set to the smallest power of* 2 *larger than* $4n\lambda\rho$*, and* $\mathfrak{m}'$ *the inverse of* $\mathfrak{m} \mod X^n-\lambda \mod \phi$ *exists,*
- γ *is a* n*-th root of* λ*, which can be extracted from the Hermite Normal Form of the circulant lattice basis*

$$HNF\begin{pmatrix} \mathfrak{m}_0 & \mathfrak{m}_1 & \mathfrak{m}_2 & \cdots & \mathfrak{m}_{n-1} \\ \lambda\mathfrak{m}_{n-1} & \mathfrak{m}_0 & \mathfrak{m}_1 & \ddots & \mathfrak{m}_{n-2} \\ \lambda\mathfrak{m}_{n-2} & \lambda\mathfrak{m}_{n-1} & \mathfrak{m}_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathfrak{m}_1 \\ \lambda\mathfrak{m}_1 & \cdots & \lambda\mathfrak{m}_{n-2} & \lambda\mathfrak{m}_{n-1} & \mathfrak{m}_0 \end{pmatrix} = \begin{pmatrix} r & 0 & \cdots & 0 \\ -\gamma & 1 & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ -\gamma^{n-1} & 0 & 0 & 1 \end{pmatrix}$$

Proof. The rows of the above circulant matrix generate the lattice formed by coefficient vectors of all algebraic multiples of $\mathfrak{m}$ modulo $X^n-\lambda$. By definition, the determinant of this lattice is exactly r, which is also the resultant of $\mathfrak{m}$ and $X^n-\lambda$. This means that they share a common root γ such that $\mathfrak{m}(\gamma)=0 \mod r$ and $X^n=\lambda \mod r$. The right side HNF basis codes all polynomials which zero on γ modulo r: it contains the previous lattice, and has the same determinant r, so the generated lattices are equal. Therefore the right side HNF is the hermite normal form of the circulant basis, and allows to compute a root γ of λ even when the factorization of q is unknown. From the circulant basis, we see that a sublattice contains n independent vectors of euclidean norm $\leq \lambda \left\|\mathfrak{m}\right\|_2$, so the covering radius of the lattice is $\leq \sqrt{n}\lambda \left\|\mathfrak{m}\right\|_2 \leq n\lambda \left\|\mathfrak{m}\right\|_\infty$. Therefore we can choose $\rho = n\lambda \left\|m\right\|_\infty$.

The algorithm is sumarized in Algorithm 4.

The advantage of this construction is that we do not have to proceed to an LLL reduction in order to generate the parameters for the AMNS basis. Furthermore the vector m is invertible $\mod (P_{\mathcal{B}},\phi)$ by construction. This algorithm returns the parameters of an AMNS basis, also the vector m and its inverse mI modulo $P_{\mathcal{B}}$ and ϕ.

Algorithm 4 Generate AMNS parameters

Require: A dimension n, a size s, and a small integer λ
Ensure: An AMNS parameter set $\mathcal{B} = (q, n, \lambda, \gamma, \rho)$ and $(\mathfrak{m}, \mathfrak{m}', \phi)$

1: **repeat**
2: Choose a vector $(m_0, \ldots, m_{n-1})$ with all coefficients $\in [-2^s, 2^s]$
3: Compute the resultant q' of $m = \sum_{i=0}^{n} m_i X^i$ and $P_{\mathcal{B}}(X) = X^n - \lambda$.
4: **until** q' is prime or has a large prime factor q
5: Chose $\rho = \lambda n \left\|\mathfrak{m}\right\|_\infty$ and $\phi = 2^{\lceil 4n\lambda\rho \rceil}$
6: Compute the common root γ of $X^n - \lambda$ and $\mathfrak{m}$ modulo q (using a gcd or an HNF algorithm)
7: Compute the inverse $\mathfrak{m}'$ of $\mathfrak{m}$ modulo ϕ
8: **Return** the parameter set $\mathcal{B} = (q, n, \lambda, \gamma, \rho)$ and $(\mathfrak{m}, \mathfrak{m}', \phi)$

In cases where Gentry's construction cannot be used. Typically, in pairing based cryptography, where the prime q is imposed by the choices of the parameters. We have to construct an efficient base AMNS dealing with a fixed value of the prime q.

3.3 Construction when we cannot choose q

In pairing based cryptography, the prime q, the fields $\mathbb{F}_q$ and the extensions $\mathbb{F}_{q^m}$ are fixed during the construction of the elliptic curve. We do not have enough freedom to use the Gentry-like algorithm of Section 3.2 to generate the AMNS basis.

Of course, selecting another elliptic curve would produce another prime q. This fact can be used to tune q until $2m$ divides $q-1$. This ensures the existence of $2m$ roots of unity, and allows to set $\lambda = -1$. However, once q is chosen, the only way to generate $\mathfrak{m}$ is to find a short vector of the lattice spanned by

$$\mathcal{B} = \begin{pmatrix} q & 0 & 0 & 0 & \ldots & 0 \\ -\gamma & 1 & 0 & 0 & \ldots & 0 \\ -\gamma^2 & 0 & 1 & 0 & \ldots & 0 \\ \vdots & 0 & 0 & \ddots & 0 & \vdots \\ -\gamma^{n-2} & 0 & 0 & \vdots & 1 & 0 \\ -\gamma^{n-1} & 0 & 0 & \ldots & 0 & 1 \end{pmatrix}.$$

This lattice reduction phase already existed in [3]. It can be performed using the LLL algorithm. Here, we prove that after these steps, we can choose the small vector $\mathfrak{m}$ having an odd resultant with $X^m + 1$. This allows ϕ to be a power of 2.

Lemma 4. *Let $\mathcal{L}$ be the lattice of the AMNS representation, generated by the basis $\mathcal{B}$ above. Let $\mathcal{M}$ be another basis of $\mathcal{L}$. Then, for any $\phi = 2^k$ a power of 2, there exist at least one vector $\mathfrak{m}$ of $\mathcal{M}$ such that $\mathfrak{m}$ is invertible modulo* mod $(P_{\mathcal{B}}, \phi)$

Proof. First, note that a polynomial $\mathfrak{m}$ is invertible $\mod (P_{\mathcal{B}}, \phi)$ as soon as the evaluation of $\mathfrak{m}$ over all integers are odd. By interpolation arguments, it is enough to verify that $\mathfrak{m}(x)$ is odd on at least n integers $\in [0, \phi - 1]$. In the public basis $\mathcal{B}$, this is the case of the constant polynomial $\mathfrak{b}_0 = q$. All the other rows can be put on the form $\mathfrak{b}_k = X^k - \alpha_k$ where α_k is odd (by adding q if necessary). Of course, $\mathfrak{b}_1, \ldots \mathfrak{b}_{k-1}$ are not invertible since their evaluations on 1 are always even.

Let $\mathcal{M} = [\mathfrak{m}_0, \mathfrak{m}_1, \ldots, \mathfrak{m}_{n-1}]$ be another basis of $\mathcal{L}$, it is obtained by left multiplication of $\mathcal{B} = [\mathfrak{b}_0, \mathfrak{b}_1, \ldots, \mathfrak{b}_{n-1}]$ by a unimodular matrix. Consequently, there exists an index i such that $\mathfrak{m}_i = u_0\mathfrak{b}_0 + \cdots + u_{n-1}\mathfrak{b}_{n-1}$ where u_0 is odd. Then by construction, $\mathfrak{m}_i$evaluates to an odd number on every odd integer, and therefore, it is invertible $\mod (P_{\mathcal{B}}, \phi)$. □

By this lemma, once a lattice reduction algorithm (like LLL [15], BKZ [23], or better, HKZ [22]) has been run on $\mathcal{B}$, this property implies that we always have a vector in $\mathcal{M}$ invertible $\mod (P_{\mathcal{B}}, \phi)$. We can choose this vector for the Montgomery's like multiplication, together with $\phi = 2^k$. Even if this is not the first vector of $\mathcal{M}$, the norm of $\mathfrak{m}$ remains short.

4 Implementation and results

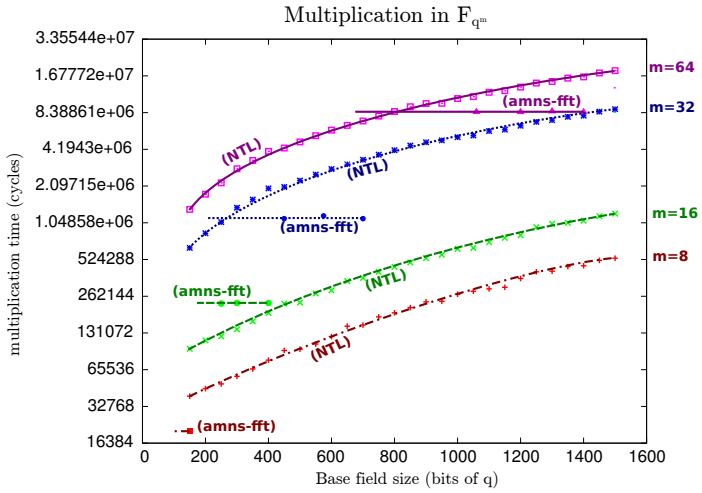

Fig. 1. Running time in cycles of multiplication in $\mathbb{F}_{q^m}$, AMNS with FFT versus NTL ZZ_pE

In order to illustrate our approach, we implemented the AMNS arithmetic for $\mathbb{F}_{q^m}$ where q had between 150 bits and 1500 bits, and for various extensions

$m = 8$, 16, 32 and 64 with a FFT. Other small non-power-of-two extensions, like $m = 17$ use the DFT. Both algorithms are given in Section 2.

Our implementation is written in C++ language, and is tested on an Intel Core i5 laptop with a 32-bit Linux platform[1]. The AMNS parameters are chosen such that the representation can fit on 32-bit integers, and during the computation, temporary variables are stored on 64-bit integers, so that they can store products without overflow. Note that on 64-bit platforms, the size of the parameters can be doubled, so that AMNS vectors fit on 64-bit integers and temporaries on 128-bit long integers.

By construction, the complexity of the arithmetic in AMNS representation only depends on the dimension n and the extension degree $m \leq n$. When these two dimensions are fixed, the running time does not depend on the bit-size of the corresponding base field q, as soon as there exist AMNS parameters for $\mathbb{F}_q$ (basically, $2n|q-1$ and and $q = O(\rho^n)$).

Note that the AMNS arithmetic requires some precomputations, for instance of the polynomial $\mathfrak{m}$. In order to make a fair comparison, we chose to compare our results with the NTL 5.4.2 [24][2] version of arithmetic on the same field extensions using the ZZ_pE module. This is the fastest module of NTL to perform arithmetic in a fixed field extension $\mathbb{F}_q[X]/P$, and allows for instance NTL to preprocess some data, based on q and P, in order to speed-up all operations. Of course, the arithmetic in the base field of NTL is performed by GMP, which is also highly optimized in assembly.

Figure 1 provides the reader with a comparison between the computational time required by our AMNS library and NTL for multiplication over $\mathbb{F}_{q^m}$.

The running time of a multiplication in cycles of AMNS and NTL based multiplications. We restricted the AMNS parameters to the case $m = n$, and the results are obtained with the -O9 -static -funroll-all-loops -masm=intel -ftree-vectorize -msse3 optimization flags of g++.

For $m = 8$, our AMNS implementation is more efficient than NTL for q smaller than 180 bit. For $m = 16$, our AMNS implementation less efficient than NTL. However, for larger extension degrees, according to the results reported in Figure 1, our implementation of AMNS multiplication is faster than NTL for large q, like extension degree $m = 32$ and $q \geq 300$ bits, or $m = 64$ and $q \geq 800$ bits.

5 Conclusion

We revised the study of AMNS bases in order to improve the arithmetic over finite field extensions. We propose new efficient routines to efficiently construct AMNS bases. An easy one, when the characteristic can be freely chosen, is derived from Gentry's algorithm. Else, when the prime number q is constrained by

[1] Ubuntu precise 12.04, with kernel 3.2.0-21-generic-pae, 4Gb RAM, and compiler g++ version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu3)

[2] libntl-dev, as packaged by default on ubuntu 11.10. It is statically linked with GMP 4.3.2 for large integer operations.

other parameters of the protocol, we develop an explicit construction of efficient AMNS bases. We also propose the first software implementation of the AMNS arithmetic over fields extensions, which runs faster than GMP over large fields. It could be interesting to compare an implementation of a pairing based protocole in AMNS representation with an implementation in a classical representation.

Open problems:

- As mentionned in Remark 2 the equality test in the AMNS representation is not straightforward. It would be interesting to compare the different methods to compare a number in the AMNS representation.
- According to the protocol, it could be interesting to find an efficient method to implement the division in the AMNS representation.
- We compare the multiplication in an extension field in the AMNS representation with the classical multiplication implemented in NTL. This work can be completed with the comparaison of the implementation of a cryptographic protocol over finite field in classical representation and in the AMNS representation.

Acknowledgments

The authors would like to thank Jean Luc Beuchat, Thomas Plantard, and Peter Schwabe for their invaluable comments and helpful suggestions on our preliminary manuscript, and the anonymous reviewers for their numerous suggestions and remarks which have enables us to substantially improve the paper.

References

1. NIST Key Length Recommendations. http://www.keylength.com/.
2. Recommendations for Key Management. Special Publication 800-57 Part 1, 2007.
3. J.C. Bajard, L. Imbert, and T. Plantard. Modular number systems: beyond the Mersenne family. In *Selected Areas in Crytography: 11th International Workshop, SAC 2004*, volume 3357 of *LNCS*, pages 159–169. Springer-Verlag, 2004.
4. M. Bodrato. Towards Optimal Toom-Cook Multiplication for Univariate and Multivariate Polynomials in Characteristic 2 and 0. In *WAIFI 2007*, volume 4547 of *Lecture Notes in Computer Science*, pages 116–133. Springer, 2007.
5. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. CRYPTO '01, pages 213–229, London, UK, 2001. Springer-Verlag.
6. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Journal of Cryptology*, volume 17 of *Lecture Notes in Computer Science*, pages 297–319, 2004.
7. E. Brier and M. Joye. Fast point multiplication on elliptic curves through isogenies. In M. P. C. Fossorier, T. Høholdt, and A. Poli, editors, *AAECC*, volume 2643 of *Lecture Notes in Computer Science*, pages 43–50. Springer, 2003.
8. J. Chung and A. Hasan. More Generalized Mersenne Numbers: (extended abstract). In M. Matsui and R. J. Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 335–347. Springer, 2003.

9. A. J. Devegili, C. Ó hÉigeartaigh, M. Scott, and R. Dahab. Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report 2006/471, 2006. http://eprint.iacr.org/.
10. N. El Mrabet and C. Nègre. Finite field multiplication combining AMNS and DFT approach for pairing cryptography. In *ACISP '09: Proceedings of the 14th Australasian conference on Information Security and Privacy*, volume 5594, pages 422–436, London, UK, 2009. Springer-Verlag.
11. N. Gama and P. Q. Nguyen. Predicting Lattice Reduction. In *proceedings of Eurocrypt'08, LNCS, Springer Verlag*, pages 31 – 51, 2008.
12. N. Gama, P. Q. Nguyen, and O. Regev. Lattice Enumeration using Extreme Pruning. In *proceedings of Eurocrypt 2010, LNCS, Springer Verlag*, 2010.
13. C. Gentry and S. Halevi. Implementing Gentry's Fully Homorphic Encryption Scheme. In *proceedings of Eurocrypt 2011, LNCS 6632, Springer Verlag*, pages 129 – 148, 2011.
14. R. Katti and J. Brennan. Low Complexity Multiplication in a Finite Field Using Ring Representation. *IEEE Transactions on Computers*, 52(4):418–427, 2003.
15. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Ann.*, 261:513–534, 1982.
16. V. Miller. Use of elliptic curves in cryptography. *Advances in Cryptology-Crypto 85*, 218:417–426, LNCS 1986.
17. H. Minkowski. *Geometrie der Zahlen.* Leipzig und Berlin, Druck und Verlag von B.G. Teubner, 1910.
18. P. L. Montgomery. Five, six, and seven-term Karatsuba-like formulae. *IEEE Transactions on Computers*, 54(3):362–369, 2005.
19. N. El Mrabet, A. Guillevic, and S. Ionica. Efficient multiplication in finite field extensions of degree 5. In *Proceedings of Africacrypt 2011*. Springer, LNCS, 2011.
20. C. Nègre and T. Plantard. Efficient modular arithmetic in adapted modular number system using Lagrange representation. In *ACISP '08: Proceedings of the 13th Australasian conference on Information Security and Privacy*, pages 463–477, Berlin, Heidelberg, 2008. Springer-Verlag.
21. K. Rubin and A. Silverberg. Torus-Based Cryptography. In D. Boneh, editor, *Advances in Cryptology CRYPTO 2003*, volume 2729, pages 349–365. Springer, 2003.
22. C-P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.
23. C-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math. Programming*, 66:181–199, 1994.
24. V. Shoup. Number Theory Library, http://www.shoup.net/ntl, 1996.
25. J. H. Silverman. Rings of low multiplicative complexity. In *Finite Fields and Their Applications*, volume 6 of *Academic Press*, pages 175–191, 2000.
26. M. van Dijk, R. Granger, D. Page, K. Rubin, A. Silverberg, M. Stam, and D. Woodruff. Practical cryptography in high dimensional tori. In *Advances In Cryptology eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 234–250. Springer Verlag, 2005.
27. J. Von ZurGathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, New York, NY, USA, 2003.