

Reduction Algorithms for Polynomials

F. ANGELINI

angelini@unipg.it

*Istituto di Matematica generale e finanziaria, Facoltà di Economia, Università di Perugia, 06100 Perugia
Italy*

M. BUCCI

bucci@fub.it

Fondazione U. Bordoni, Via B. Castiglione, 59, 00143 Rome, Italy

Abstract. We present two reduction algorithms for polynomials to the end of an efficient implementation of arithmetic operations in finite fields of large size. These are the version for polynomials of two algorithms originally conceived for integers: a refinement of Barrett's reduction algorithm and the Montgomery's reduction algorithm. The analogy between the polynomial and the integer version suggests a way to design dual mode arithmetic devices that can operate on both algebraic structures.

1. Introduction

For cryptographic applications it is very important to perform in an efficient way arithmetic operations in finite fields of large size. Due especially to the use of elliptic curves in cryptography, like in the Diffie-Hellman key exchange algorithm [10] or in some signature schemes (see [13]), in the last years there has been much work and progress towards fast software and hardware implementations of the arithmetic operations of the field $GF(2^n)$, for large n ([1], [2], [3], [7], [8], [12], [15], [16]). When using polynomial representation for the finite field, the basic and expensive operation is the reduction modulo a fixed irreducible polynomial of high degree. The aim of this note is to present Barrett and Montgomery type reduction algorithms for polynomials. Montgomery type multiplication in finite fields is already proposed in [12] and we do not have any real novelty to add. However we decided to include here also the Montgomery type reduction algorithm RID-2 because it comes naturally, and in a simple way, from a certain algebraic frame, together with the Barrett type one. The latter, which we call RID-1, has a very similar structure of RID-2, but it has the advantage of giving as an output the required result, avoiding the final multiplication typical in the Montgomery reduction.

RID-1 presents a strong analogy with a multi-iteration refinement of Barrett's algorithm ([4]) for integers given in [5] and [6]. This would also suggest a way to design arithmetic

devices that can operate both on polynomials and integers, simply by switching from the polynomial addition mode to the integer one. Such devices would be useful for instance in some applications related to elliptic curves, like the implementation of the Diffie-Hellman key exchange algorithm, where finite field arithmetic is needed, together with a signature scheme, where also modular arithmetic comes in.

We remark that some adaptations of Montgomery's original algorithm ([14]) to polynomials appear also in [9], one done in only one step and the other done one bit by one bit, with a clear generalization to a word-level algorithm using lookup tables. On the other hand an adaptation of Barrett's original work to polynomials is already contained in [9] where the reduction is done in just one step. A Barrett type algorithm for polynomials with more than one step is proposed in [8], but they need a pre-calculated lookup table, limiting the length of the pace according to memory requirements, and also in [16] where they use a particular trinomial as modulo. Algorithm RID-1 is in fact similar to the one in [8] but it does not need lookup tables and it has a flexible pace size. Also it reduces to the one in [16] for that particular choice of the modulo.

The key ingredient of both algorithms is an "approximation" of the inverse of the modulo in the field of formal power series in y and y^{-1} truncated either to the left (in the case of RID-1) or to the right (in the case of RID-2). In Section 2 we give an overview of the algebraic objects for completeness and for convenience of the reader, even if one does not strictly need all this formalism. These approximations are the analogous for polynomials of the ones given respectively by Bucci-Di Porto and Montgomery. Once we have done that, the algorithms, which are exposed with proofs in Section 3, work at the same way as the ones for integers, of course after changing the underlying addition.

2. Algebraic Preliminaries

Let F be a field. In the applications it is particularly interesting the case $F=\mathbb{Z}_2$ the field of integers modulo 2.

We recall some definitions from algebra. In practice our algorithm does not strictly need the language and the results we will mention and therefore we will be kind of sketchy here. However we felt that the algebraic frame, in which the algorithm can be put, is worth to be shown.

We would like first to consider the (local) ring of formal power series, denoted by $F[[x]]$. A typical element of $F[[x]]$ can be written as $\sum_{i=0}^{\infty} a_i x^i$, with $a_i \in F$. We have an obvious inclusion $F[x] \subset F[[x]]$. The sum of two elements $\sum_{i=0}^{\infty} a_i x^i$, $\sum_{i=0}^{\infty} b_i x^i \in F[[x]]$ is $\sum_{i=0}^{\infty} (a_i + b_i) x^i$ and the product is $\sum_{i=0}^{\infty} c_i x^i$, where $c_i = \sum_{k=0}^i a_k b_{i-k}$. We recall also that an element

$f = \sum_{i=0}^{\infty} a_i x^i \in F[[x]]$ has an inverse if and only if $a_0 \neq 0$. Indeed if $a_0 \neq 0$, then we can find $g = \sum_{i=0}^{\infty} b_i x^i \in F[[x]]$ such that $gf = fg = 1_F$, by taking $b_0 = a_0^{-1}$ and then by solving recursively the system $\sum_{k=0}^i a_k b_{i-k} = 0$. The viceversa is easier (see for example [11], Ch. III).

Now we set some terminology, which we do not pretend to be standard, but which is convenient to the end of the paper. Let us consider a formal power series in x and x^{-1} , which can be written as $Z = \sum_{i=-\infty}^{+\infty} z_i x^i$. If $z_k = 0$ for all $k > d$, we say that Z has degree less than or equal to d (note that d could be negative) and if moreover $z_d \neq 0$ then we say that Z has degree d . If $z_k = 0$ for $k < d$, we say that Z has degree greater than or equal to d . For such an element Z , we define, for an integer u ,

$${}_u Z = \sum_{i \leq u-1} z_i x^i$$

the part of degree less than or equal to $u-1$ (of course, if $\deg(Z) \geq d$ and $u < d$, then ${}_u Z = 0$), and

$$Z_u = \sum_{i \geq u} z_i x^i$$

the part of degree greater than or equal to u (if $u > \deg(Z)$, we set $Z_u = 0$). We then may write $Z = {}_u Z + Z_u$.

Having said that, we consider $F[[x, x^{-1}]]$ the set of formal power series in x and x^{-1} which have only finitely many nonzero coefficients in the negative part, i.e. an element Z belonging to $F[[x, x^{-1}]]$ can be written as

$$Z = \sum_{i \geq r} z_i x^i$$

for a certain integer (positive or negative) r . $F[[x, x^{-1}]]$ is actually a field, namely the *field of fractions* of $F[[x]]$. Its construction, starting from $F[[x]]$, is analogous to the construction of rational numbers from integers (see again [11], Ch. III, for the definition of field of fractions). $F[[x]]$ is included in $F[[x, x^{-1}]]$ as the elements of degree greater than or equal to 0. The inverse of the element Z (supposing $z_r \neq 0$) is obtained by writing $Z = x^r Z'$, with $Z' \in F[[x]]$ invertible in $F[[x]]$, and so $Z^{-1} = x^{-r} (Z')^{-1}$.

Similarly we have $F[[y^{-1}, y]]$, the field of formal power series in y^{-1} and y truncated to the left, i.e. with bounded degree, which is the field of fractions of the ring of power series in y^{-1} $F[[y^{-1}]]$. An element S of $F[[y^{-1}, y]]$ can be written as

$$S = \sum_{i \leq p} s_i y^i$$

for a certain integer (positive or negative) p .

We note that there is a transformation from $F[[x, x^{-1}]]$ to $F[[y^{-1}, y]]$ simply obtained sending x to y^{-1} and viceversa.

We will concentrate our attention on $F[[y^{-1}, y]]$, being $F[[x, x^{-1}]]$ very similar. Sums and products are inherited from the ring of formal power series, namely if $\sum_{i \leq p} a_i y^i$ and $\sum_{i \leq q} b_i y^i$ are two elements of $F[[y^{-1}, y]]$, with $p \geq q$, then $\sum_{i \leq p} a_i y^i + \sum_{i \leq q} b_i y^i = \sum_{i \leq p} (a_i + b_i) y^i$, and

$$\left(\sum_{i \leq p} a_i y^i\right) \times \left(\sum_{i \leq q} b_i y^i\right) = \sum_{i \leq p+q} c_i y^i, \text{ where } c_i = \sum_{\substack{k, h \\ k+h=i}} a_k b_h.$$

Note that, if $S = \sum_{i \leq p} a_i y^i$ and $T = \sum_{i \leq q} b_i y^i$, by definition of product we have the following property:

$$(1.1) \quad (S \times T)_{p+q-(t-1)} = (S_{p-(t-1)} \times T_{q-(t-1)})_{p+q-(t-1)};$$

Similarly, if $S = \sum_{i \geq p} a_i x^i$ and $T = \sum_{i \geq q} b_i x^i$ belong to $F[[x, x^{-1}]]$, we have:

$$(1.2) \quad {}_{p+q+t}(S \times T) = {}_{p+q+t}({}_{p+t}S \times {}_{q+t}T).$$

Now a polynomial C of degree n can be considered as an element of both $F[[y^{-1}, y]]$ and $F[[x, x^{-1}]]$. We are interested in finding the inverse of C in these fields. Let us do so explicitly in $F[[y^{-1}, y]]$. Write $C = c_0 + c_1 y + \dots + c_n y^n$, with $c_n \neq 0$. The unique inverse of C has to be of the form

$$B = \dots + b_{-(n+2)} y^{-(n+2)} + b_{-(n+1)} y^{-(n+1)} + b_{-n} y^{-n}$$

and the first thing we see is that $b_{-n} = c_n^{-1}$. To find the other coefficients of B we set

$$b_{-n} c_{(n-1)} + b_{-(n+1)} c_n = 0$$

from which we find $b_{-(n+1)}$. Then recursively we solve

$$b_{-n} c_{(n-2)} + b_{-(n+1)} c_{(n-1)} + b_{-(n+2)} c_n = 0$$

...

$$b_{-n} c_{(n-s)} + b_{-(n+1)} c_{(n-s+1)} + \dots + b_{-(n+s)} c_n = 0$$

...

B is then again a truncated formal power series in y and y^{-1} , with degree $-n$, i.e. an element of $F[[y^{-1}, y]]$ and has the property that $C \times B = B \times C = 1$. Moreover the coefficients of the t highest degrees of B , namely b_{-n} , $b_{-(n+1)}$, ..., $b_{-(n+t-1)}$, depend only on the highest t coefficients of C , c_n , $c_{(n-1)}$, ..., $c_{(n-t+1)}$. It makes sense then to consider the "approximation" of $B = C^{-1}$

$$B_{-n-(t-1)} = b_{-(n+t-1)} y^{-(n+t-1)} + \dots + b_{-(n+2)} y^{-(n+2)} + b_{-(n+1)} y^{-(n+1)} + b_{-n} y^{-n}$$

which can be computed only from $C_{(n-t)}$ and which satisfies the following property:

$$C \times B_{-n-(t-1)} = B_{-n-(t-1)} \times C = 1 + H \text{ with } H \in F[[y^{-1}]] \text{ and } \deg(H) \leq -t ;^1$$

i.e.

$$(1.3) \quad (C \times B_{-n-(t-1)})_{-(t-1)} = (B_{-n-(t-1)} \times C)_{-(t-1)} = 1$$

Property (1.3) holds by construction.

Similarly, if we view $C = c_0 + c_1x + \dots + c_nx^n$ as an element of $F[[x, x^{-1}]]$ and we suppose, which it will be true in the applications, that $c_0 \neq 0$, we can find its inverse recursively in the form

$$B = b_0 + b_1x + b_2x^2 + \dots$$

with $b_0 = c_0^{-1} \neq 0$, and its approximation $B = b_0 + b_1x + b_2x^2 + \dots + b_{(t-1)}x^{(t-1)}$ has the property

$$C \times B = B \times C = 1 + K, \text{ with } K \in F[[x]] \text{ and } t \leq \deg(K);^2$$

i.e.

$$(1.4) \quad (C \times B)_t = (B \times C)_t = 1.$$

3. The Algorithms

We are given two polynomials D and C in the ring of polynomials with coefficients in F , of degrees m and n respectively and we suppose $m \geq n$. The problem is to find the residue of $D \bmod C$, i.e. to find a polynomial R , with degree of R strictly less than n such that

$$D = QC + R$$

for a certain $Q \in F[x]$.

The problem is fundamental in doing arithmetic of finite fields, viewing the elements of the field as polynomials modulo a given fixed irreducible polynomial. For example the Galois field with 2^n elements may be realized as $Z_2[x]/(C)$, where C is an irreducible polynomial of degree n .

Algorithm RID-1

We will work in the field $F[[y^{-1}, y]]$ and to simplify computations F will be Z_2 , the field of integers modulo 2. Let us fix a positive integer t . Suppose for simplicity that $m = kt - 1$ and $n = ht - 1$, with $k \geq h$ (see observations 2. and 3. below for generalizations).

We write

$$D = {}_{(k-1)t}D + D_{(k-1)t}$$

$$C = {}_{(h-1)t}C + C_{(h-1)t}$$

with $D_{(k-1)t} = d_{(k-1)t}y^{(k-1)t} + \dots + d_{kt-1}y^{kt-1}$ and $C_{(h-1)t} = c_{(h-1)t}y^{(h-1)t} + \dots + c_{ht-1}y^{ht-1}$. In our case, where we suppose $d_{(k-1)t} \neq 0$ and $c_{(h-1)t} \neq 0$, these are the t most significant coefficients of D and C . The approximation $B_{-(ht-1)-(t-1)}$ at the t highest coefficients of the inverse of C can be written as

$$B_{-(ht-1)-(t-1)} = b_{-(ht-1)-(t-1)}y^{(ht-1)-(t-1)} + \dots + b_{-(ht-1)-1}y^{(ht-1)-1} + b_{-(ht-1)}y^{(ht-1)}$$

and can pre-computed from $C_{(h-1)t}$.

RID-1

1. $X(0) = D$
2. for $i=1$ to $k-h$
3. $Y(i-1) = X(i-1)_{(k-i)t} \times B_{-(ht-1)-(t-1)}$
4. $X(i) = X(i-1) + Y(i-1)_{(k-h-i)t+1} \times C$
5. if $\deg(X(k-h)) = \deg(C)$ then
6. $R = X(k-h) + C$
7. else
8. $R = X(k-h)$.

We remark that we do addition instead of subtraction being in a field of characteristic 2. Also, in line 6, the computation to find R would be slightly different in characteristic different from 2. Note that $X(i)$ is a polynomial, for each i . At each step the algorithm reduces the length of the operand of at least t bits, the most significant ones, thank to property (1.1) and (1.3).

Proof: We have to show that $\deg(X(i)) \leq (k-i)t-1$, for all i , supposing $\deg(X(i-1)) \leq (k-i+1)t-1$, which is true for $i=1$. For this it is enough to show that

$$(Y(i-1)_{(k-h-i)t+1} \times C)_{(k-i)t} = X(i-1)_{(k-i)t}.$$

But this is true because, having in mind property (1.1),

$$\begin{aligned} (Y(i-1)_{(k-h-i)t+1} \times C)_{(k-i)t} &= (X(i-1)_{(k-i)t} \times B_{-(ht-1)-(t-1)} \times C)_{(k-i)t} \\ &= (X(i-1)_{(k-i)t} \times (B_{-(ht-1)-(t-1)} \times C)_{-(t-1)})_{(k-i)t}. \end{aligned}$$

And now, by property (1.3), we get what we need.

Algorithm RID-2

Here we work with $F[[x, x^{-1}]]$, where F is again Z_2 . Now we write

$$D = {}_t D + D_t$$

$$C = {}_t C + C_t$$

and $B = b_0 + b_1 x + b_2 x^2 + \dots + b_{(t-1)} x^{(t-1)}$ the approximation of the inverse of C , which can be pre-computed from ${}_t C$.

RID-2

1. $X(0) = D$
2. for $i = 1$ to $k - h$
3. $Y(i-1) = {}_t X(i-1) \times {}_t B$
4. $X(i) = (X(i-1) + {}_t Y(i-1) \times C) \times x^t$
5. if $\deg(X(k-h)) = \deg(C)$ then
6. $R' = X(k-h) + C$
7. else
8. $R' = X(k-h)$.
9. $R = R' \times x^{(k-h)t} \bmod C$.

Proof: At each step the algorithm reduces the length of the operand of at least t bits, the least significant ones, thank to properties (1.2) and (1.4). First it is clear that $\deg X(i) \leq \deg(X(i-1)) - t$, for each i . We have now to show that

$$\deg(X(i-1) + {}_t Y(i-1) \times C) \geq t$$

for each i , so that $X(i)$ is a polynomial, i.e. it has degree greater than or equal to zero. It is enough to show that

$${}_t (Y(i-1) \times C) = {}_t X(i-1),$$

which is true, by properties (1.2) and (1.4). Indeed

$${}_t (Y(i-1) \times C) = ({}_t X(i-1) \times {}_t B \times C) = ({}_t X(i-1) \times ({}_t B \times C)) = {}_t X(i-1).$$

Lines 9 comes from the fact that, at each step, we multiply by x^t .

Observations:

1. The final step of the algorithms, lines 5 to 8, is just one step of the algorithm itself with $t=1$. Note that in case of repeated multiplications in the finite field, as it is the case in the applications, one could use $X(k-h)$ to proceed and do this final step at the very end.
2. If we suppose that $\deg(D) \leq kt-2$ in RID-1 (this is indeed the case when performing multiplication in the finite field $Z_2[x]/(C)$, where one first multiplies two polynomials of degree less than or equal to $ht-2$ getting a polynomial with degree less than or equal to $2ht-4$ and then divides the result by C) we can proceed in a slightly different way, avoiding the final anomalous step (line 5 to 8). Namely at each step we may take the truncations $X(i-1)_{(k-i)t-1}$ and $Y(i-1)_{(k-h-i)t}$ which still represent the highest t bits of the quantities considered. One can check that the algorithm still works and that we automatically get $\deg(X(k-h)) < \deg(C)$. This correspond to having divided D and C in a certain number of segments of length t in a non-aligned way, i.e. shifted by one bit, whereas in the algorithm as it was described the segmentation was aligned.
3. More generally RID-1 can be applied without the hypothesis $n=ht-1$ and $m=kt-1$, which was made for clarity. The algorithm would still reduce the dividend length of t bits at each step, with the restriction $t \leq \deg(X(i-1)) - \deg(C) + 1$ (so that the partial quotients have degree greater than or equal to zero), until we get a result with degree strictly less than C . One may even think of an algorithm with variable pace, by choosing at each step a suitable t_i (always subject to $t_i \leq \deg(X(i-1)) - \deg(C) + 1$), and with the (theoretically non strictly needed) further condition that the approximation of C^{-1} remains equal at each step. As an example of this one can look at the two step reduction algorithm contained in [16] where at the first step $t=93$, at the second $t=61$ and the approximation of C^{-1} is in both cases equal to y^{-155} .

Acknowledgment

We would like to thank Erik De Win for giving us reference [12].

Notes:

1. This is the analogous of Bucci-Di Porto's approximation of the inverse.
2. For a comparison with Montgomery note that ${}_tB = ({}_tC)^{-1} \bmod x^t$.

References

1. Agnew, T. Beth, R. Mullin, S. Vanstone, *Arithmetic operations in $GF(2^m)$* , Journal of Cryptology, 6 (1993), 3-13.

2. Agnew, R. Mullin, I. Onyszchuk, S. Vanstone, *An implementation for a fast public key cryptosystem*, Journal of Cryptology, 3 (1991), 63-79.
3. G. Agnew, R. Mullin, S. Vanstone, *An implementation of elliptic curve cryptosystems over F_2^{155}* , IEEE Journal on Selected Areas in Communications, 11 (1993), 804-813.
4. P. D. Barrett, *Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor*, Advances in Cryptology, Proc. Crypto '86, LNCS 263 (1987), Springer-Verlag, 311-323.
5. M. Bucci, A. Di Porto, *Algoritmi di divisione con inverso precalcolato del divisore*, Raccolta Pubblicazioni FUB 1987, Vol. Sicurezza e Protezione delle Informazioni, (Rel. 3b1087) 131-145. (in italian)
6. M. Bucci, A. Di Porto, *A fast algorithm for large number division*, Proc. of the 3rd Symposium on State and Progress of Research in Cryptography, Rome, Italy, 1993, William Wolfowicz Ed., 241-254.
7. T. Beth, D. Gollmann, *Algorithm engineering for public key algorithms*, IEEE Journal on Selected Areas in Communications, 7 (1989), 458-466.
8. E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gerssem, J. Vandewalle, *A fast software implementation for arithmetic operations in $GF(2^n)$* , Advances in Cryptology, Asiacrypt '96, LNCS 1163, Springer-Verlag, 1996, 65-76.
9. E. De Win, P. De Gerssem, *Studie en implementatie van arithmetische bewerkingen in $GF(2^n)$* , Master Thesis K. U. Leuven (1995). (in Dutch)
10. Diffie, M. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, 22 (1976), 644-654.
11. T. Hungerford, *Algebra*, Springer-Verlag, New York, 1974.
12. Ç. K. Koç, T. Acar, *Montgomery multiplication in $GF(2^k)$* , (to appear).
13. A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.
14. P. L. Montgomery, *Modular multiplication without trial division*, Mathematics of Computation, 44 (1985), 519-521.
15. R. Mullin, I. Onyszchuk, S. Vanstone, R. Wilson, *Optimal normal bases in $GF(p^n)$* , Discrete Applied Mathematics, 22 (1988), 149-161.
16. R. Schroepel, H. Orman, S. O'Malley, O. Spatscheck, *Fast key exchange with elliptic curve systems*, Advances in Cryptology, Proc. Crypto '95, LNCS 963 (1995), Springer-Verlag, 43-56.