Algorithms for Finite Field Arithmetic

Éric Schost
Computer Science Department
Western University
eschost@uwo.ca

Categories and Subject Descriptors

F.2.1 [**Theory of computation**]: Analysis of algorithms and problem complexity—*Computations in finite fields*

General Terms

Algorithms,Theory

Keywords

Finite fields, irreducible polynomials, extensions.

ABSTRACT

We review several algorithms to construct finite fields and perform operations such as field embedding. Following previous work by notably Shoup, de Smit and Lenstra or Couveignes and Lercier, as well as results obtained with De Feo and Doliskani, we distinguish between algorithms that build "towers" of finite fields, with degrees of the form $\ell, \ell^2, \ell^3, \ldots$ and algorithms for composita. We show in particular how techniques that originate from algorithms for computing with *triangular sets* can be useful in such a context.

1. INTRODUCTION

Finite fields appear in many branches of pure and applied mathematics, prominently so in areas such as number theory, cryptography and coding theory. As a result, building and computing in arbitrary finite fields is a fundamental task for any computer algebra system; for instance, the implementation available in Magma [4] remains one of the most complete known to us.

Let us have a look at two very simple computations, both in Magma. The first one,

```
k4:=GF(5^4);
k6:=GF(3^4);
a4:=Random(k4);
a6:=Random(k6);
a:=a4+a6;
 Runtime error in '+': Arguments are not compatible
 Argument types given: FldFinElt, FldFinElt
```

obviously doesn't make sense, while the second one does:

```
k4:=GF(5^4);
k6:=GF(5^6);
a4:=Random(k4);
a6:=Random(k6);
a:=a4+a6;
Parent(a);
 Finite field of size 5^12
```

In the second example, we see that the system has to be able to construct two extensions of a base field, here $\mathbb{F}_5$, build their compositum when needed, and embed elements into this compositum.

Our goal here is to present algorithms for these tasks, from results dating back to the 1980's and 1990's to more recent progress. In most of the text, we fix a prime p and we count operations in the base field $\mathbb{F}_p$ at unit cost – that is, we work in an algebraic model; another cost measure, in a boolean model, will be used at times as well.

Precisely, the questions we want to discuss are related to the construction of arbitrary extensions of $\mathbb{F}_p$: for any integer n, we want to be able to define an extension of degree n of $\mathbb{F}_p$, together with mechanisms for computing embeddings $\mathbb{F}_{p^n} \to \mathbb{F}_{p^m}$ (and invert them, when possible) in such a way that the obvious compatibility condition is satisfied.

These requirements are weaker than the framework of "compatibly embedded finite fields" used in Magma, since the latter allows for several isomorphic versions of the same finite to be present in the system. Closer to us is the construction of "standard model of finite fields" by de Smit and Lenstra [21], but as the name implies, the latter construction possesses further canonicity properties.

An outstanding question regarding the arithmetic of finite fields is whether one can construct irreducible polynomials of an arbitrary degree n over $\mathbb{F}_p$ in time polynomial in n and $\log(p)$. So far, no such result is known, although this is known to be feasible under the Extended Riemann Hypothesis [1]. Our point of view here is the following: using probabilistic algorithms (of the Las Vegas type), the questions we are considering can indeed be solved in polynomial time; then, how fast can we really do it? Can we obtain algorithms with quasi-linear cost?

As we will see, this is not known to be the case, as long as we rely on our algebraic computation model; however, in

ISSAC'15, July 6–9, 2015, Bath, United Kingdom.
 978-1-4503-3435-8/15/07 ...$15.00.
DOI: http://dx.doi.org/10.1145/2755996.2756637.

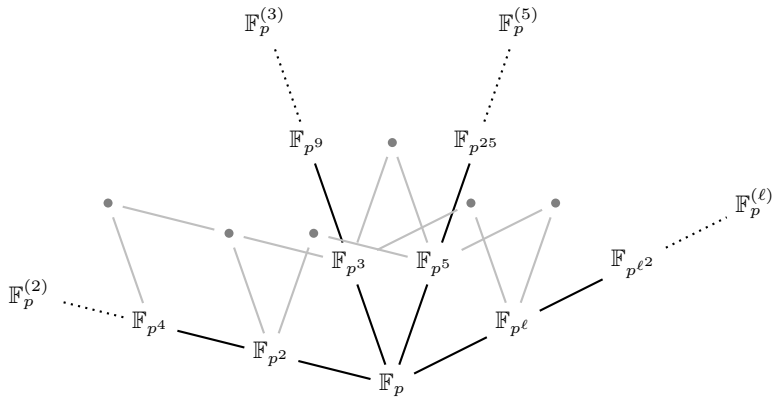

Figure 1: The algebraic closure of $\mathbb{F}_p$

a boolean model, better results can actually be obtained. Surprisingly, the main difference between these two situations is the existence of a fast algorithm due to Kedlaya and Umans [30] to perform an operation called *modular composition*.

Several previous algorithms, for instance to construct irreducible polynomials [47, 48, 15], rely on the description of $\overline{\mathbb{F}_p}$ given in Figure 1 (that figure is borrowed from [20]): in order to construct an irreducible polynomial of degree n, it is enough to factor n into prime powers, as $n = \ell_1^{e_1} \cdots \ell_s^{e_s}$, with ℓ_i's pairwise distinct primes, then construct irreducibles of degrees $\ell_1^{e_1}, \ldots, \ell_s^{e_s}$, and combine them. Possibly, a coarser factorization may be used, as in [1].

The same description underlies the algorithms we describe here: instead of focusing on the construction of irreducible polynomials, we will describe algorithms for building towers of finite fields of degrees $\ell, \ell^2, \ell^3, \ldots$ over $\mathbb{F}_p$, for ℓ prime, and for navigating through these towers; in a second time, we discuss the construction of composita, in order to be able to work with extensions of arbitrary degree n. First, however, we describe algorithms in a seemingly unrelated context, computations with triangular sets, and show in particular how modular composition techniques come into play.

Let us finally mention materials not discussed here: representation based on Zech logarithms or Conway polynomials [44], deterministic algorithms [1, 47], as well as the construction of irreducible polynomials with extra properties (sparseness, primitivity, ...), or of irreducibles whose degree satisfies prescribed bounds [23, 50].

2. TRIANGULAR SETS

We start with a subject that may not appear directly related to computations with finite fields: computations with *triangular sets*.

In what follows, we will call triangular set a sequence of polynomials $(T_1, \ldots, T_n)$ in $\mathbb{F}_p[x_1, \ldots, x_n]$, such that the following holds: for all i, T_i is in $\mathbb{F}_p[x_1, \ldots, x_i]$, monic in x_i and reduced with respect to $T_1, \ldots, T_{i-1}$; these polynomials form a Gröbner basis for the lexicographic order induced by $x_1 < \cdots < x_n$ (of course, the definition carries over to any base field). In addition, we will suppose that the ideal $\langle T_1, \ldots, T_n \rangle$ is radical.

Such a data structure, together with more general objects called *regular chains*, has a long history in the domain of polynomial system solving. Regular chains were introduced in [27], following previous work initiated by Ritt [42] and Wu [52]. Several definitions co-exist, with contributions by Lazard [32, 33], Aubry et al. [2] and Moreno Maza [38].

The complexity of basic arithmetic. Our interest here lies not in solving polynomial systems, or computing triangular sets, but rather in computing *with* triangular sets. Indeed, the structure of a family $(T_1, \ldots, T_n)$ as above shows that the quotient $\mathbb{A} = \mathbb{F}_p[x_1, \ldots, x_n]/\langle T_1, \ldots, T_n \rangle$ admits the natural multivariate basis $(x_1^{e_1} \cdots x_n^{e_n}$, with $0 \leq e_i < d_i$ for $1 \leq i \leq n)$, where we write $d_i = \deg(T_i, x_i)$. Later on, we will see that questions such as the cost of arithmetic operations in such a basis appear naturally; for the moment, note that our goal is to obtain algorithms of quasi-linear cost in $\delta = d_1 \cdots d_n$, which is the dimension of $\mathbb{A}$ over $\mathbb{F}_p$.

As it turns out, we do not have good control on the complexity of these operations. Of course, addition takes linear time, but it is not known to be the case for multiplication. Indeed, the natural approach to multiplication in a multivariate basis consists in a polynomial multiplication, followed by reduction modulo $\langle T_1, \ldots, T_n \rangle$. The initial product gives a polynomial of partial degrees $(2d_1 - 2, \ldots, 2d_n - 2)$, so the number of its monomials, $(2d_1 - 1) \cdots (2d_n - 1)$ is not linear in δ (except of course is n is fixed).

Except in a few particular cases [5], the rather direct approach outlined above is the best known so far; it leads to an overhead of about 3^n over a linear cost in δ, see [37, 35] for details.

Change of basis. All notation being as above, let us call *primitive element* an element f of $\mathbb{A}$ such that $(1, f, \ldots, f^{\delta-1})$ is an $\mathbb{F}_p$-basis of $\mathbb{A}$. In other words, knowing such an f allows us to write elements of $\mathbb{A}$ as univariate polynomials in f; this makes it possible to rely on fast FFT-based univariate polynomial arithmetic to perform computations $(+, \times)$ and $\div$ (when feasible) in $\mathbb{A}$ in quasi-linear time.

With such an approach, the main non-trivial tasks are the changes of basis, from the previous multivariate basis to the univariate one and back. As of now, no algorithm is known with a quasi-linear cost, at least in our algebraic complexity model; the best results to date take subquadratic time $C(\delta) = O^{\sim}(\delta^{(\omega+1)/2})$ [39, 41], where ω is the exponent of linear algebra.

To illustrate the issues at hand, let us consider a seemingly trivial case, taking $n = 1$. Discarding the index

$_1$, we are thus looking at a single polynomial T in $\mathbb{F}_p[x]$, and $\mathbb{A} = \mathbb{F}_p[x]/\langle T\rangle$; the "multivariate basis" here is simply $(1, x, \ldots, x^{d-1})$. Given a primitive element f, conversion from the basis $(1, f, \ldots, f^{d-1})$ to $(1, x, \ldots, x^{d-1})$ amounts to taking a polynomial g in $\mathbb{F}_p[x]$ of degree less than d, and computing $g(f) \bmod T$.

This problem is known as *modular composition*. The best known algorithms for it rely on baby-step / giant-step techniques, and involve both polynomial and matrix arithmetic [10, 26]; the former reference gives an algorithm of cost $O(d^{(\omega+1)/2})$, which is $O(d^{1.686})$ for the best known value of ω [34]; in [26], this is improved to $O(d^{1.667})$, which remains far from linear time. It is worth pointing out that modular composition lies at the heart of several other important algorithms, such as polynomial factorization over finite fields [24, 49, 29, 28].

Still in the univariate case, the inverse change-of-basis amounts to taking a polynomial h of degree less than d, and finding g such that $h = g(f) \bmod T$. As it turns out, this can be done in the same cost as in the other direction: the conversion algorithm relies on the non-degeneracy of the trace $\tau : \mathbb{A} \to \mathbb{F}_p$, to (essentially) reduce the computation to computing traces of the form $\tau(h^i)$. This can in turn be done by an algorithm for *power projection* that is dual to the one for modular composition [48, 8].

Kedlaya and Umans' algorithm. In [30], Kedlaya and Umans gave an algorithm for modular composition that takes almost linear time, in a boolean model. Precisely, for any $\varepsilon > 0$, there exists an algorithm for modular composition that runs in time $O^{\sim}(d^{1+\varepsilon}\log(p))$ in a boolean RAM; this is to be compared to the bit size of the input and output, which is proportional to $d\log(p)$.

The algorithm proceeds by lifting the computation from $\mathbb{F}_p$ to $\mathbb{Z}$, which indeed requires an analysis in a boolean model; a similar algorithm for power projection can be found in [30] as well.

Going back to our conversion problems, these results allow us to handle the (rather artificial) case $n = 1$. An extension of Kedlaya and Umans' algorithms to multivariate situations is given in [40]; it makes it possible to perform the changes of basis in n variables in time $O^{\sim}(\delta^{1+\varepsilon}\log(p))$, for any $\varepsilon > 0$, still in a boolean model.

Altogether, we are able to do conversions, and thus perform arithmetic tasks in $\mathbb{A}$ in time close to linear, as long as we use a binary model. Unfortunately, the techniques involved in Kedlaya and Umans' algorithm and its extension in [40] lead to rather large constants hidden in the big-Os. As a result, as of now, there is no known implementation of these techniques that would be competitive with the baby-step / giant-step algorithm of Brent and Kung.

Back to finite fields. Recalling Figure 1, we will follow the approach below:

- For any prime ℓ, build what we will call the *ℓ-adic tower* over $\mathbb{F}_p$, that is, the extensions
$$\mathbb{F}_p \to \mathbb{F}_{p^\ell} \to \mathbb{F}_{p^{\ell^2}} \to \cdots \to \mathbb{F}_{p^{\ell^i}} \to \cdots \qquad (1)$$
of degrees $\ell, \ell^2, \ldots, \ell^i, \ldots$ over $\mathbb{F}_p$.
- Given any coprime degrees ℓ^i and m^j, devise a way to construct the compositum
$$\mathbb{F}_{p^{\ell^i m^j}} = \mathbb{F}_{p^{\ell^i}} \otimes \mathbb{F}_{p^{m^j}}.$$

Let us first discuss the question of data representation. Looking at the nested extensions in (1), one is led to consider polynomials $(T_{\ell,i})_{i\geq 1}$, such that for all i, $(T_{\ell,1}, \ldots, T_{\ell,i})$ is a triangular set in $\mathbb{F}_p[x_1, \ldots, x_i]$ with leading degrees $(\ell, \ldots, \ell)$, and the ideal $\langle T_{\ell,1}, \ldots, T_{\ell,i}\rangle$ is maximal; our model of the field with p^{ℓ^i} elements can then be
$$\mathbb{K}_{\ell^i} = \mathbb{F}_p[x_1, \ldots, x_i]/\langle T_{\ell,1}, \ldots, T_{\ell,i}\rangle.$$
Given two coprime extension degrees ℓ^i and m^j, the compositum $\mathbb{K}_{\ell^i} \otimes \mathbb{K}_{m^j}$ is simply
$$\mathbb{F}_p[x_1, \ldots, x_i, y_1, \ldots, y_j]/\langle T_{\ell,1}, \ldots, T_{\ell,i}, T^*_{m,1}, \ldots, T^*_{m,j}\rangle,$$
where the polynomials T^*'s are obtained by evaluating T's at $y_1, \ldots, y_j$. In this representation, embeddings are straightforward, but as we saw before, arithmetic operations are more costly.

Lift up, push down. Assuming we can compute polynomials $T_{\ell,1}, \ldots, T_{\ell,i}$, the discussion above shows that introducing a primitive element for these polynomials will allow us to reduce arithmetic operations to univariate polynomial arithmetic.

In the case at hand, it is easy to see that for all i, x_i has degree ℓ^i over $\mathbb{F}_p$, so that we can write
$$\mathbb{K}_{\ell^i} \simeq \mathbb{F}_p[x_i]/\langle Q_{\ell,i}\rangle,$$
where $Q_{\ell,i} \in \mathbb{F}_p[x_i]$ has degree ℓ^i. As per the discussion in the previous paragraphs, going from a multivariate representation to a univariate one, and back, can be done using $C(\ell^i)$ operations in $\mathbb{F}_p$, and in almost linear time in a boolean model.

Looking at two levels $i-1$ and i at once, we further have
$$\mathbb{F}_p[x_i]/\langle Q_{\ell,i}\rangle \simeq \mathbb{F}_p[x_{i-1}, x_i]/\langle Q_{\ell,i-1}, T'_{\ell,i}\rangle,$$
where $T'_{\ell,i}$ is obtained from $T_{\ell,i}$ by rewriting all expressions in $x_1, \ldots, x_{i-1}$ in terms of x_{i-1} only. We call this isomorphism and its inverse *push down* and *lift up*. This equivalence is crucial: not only does it allow us to perform embeddings or compute relative traces or norms, it may also help us break down the conversion from multivariate to univariate polynomials into simpler steps, improving on the cost seen above, provided we have efficient algorithms for push down and lift up.

3. SOME USEFUL TOWERS

In addition to being one of the building blocks of algorithms for general finite field arithmetic, algorithms for computing in some specific ℓ-adic towers have found several applications. We briefly describe two of them here; both questions arise in relation to torsion subgroups of elliptic curves or Jacobians of genus 2 curves.

Quadratic extensions. The first particular case we consider is that of *quadratic extensions*. In this case, we rely on a well-known construction of such extensions [31, Th. VI.9.1], which was already put to use algorithmically in [47, 48]: if $p = 1 \bmod 4$, then for any quadratic non-residue $\alpha \in \mathbb{F}_p$, the polynomial $x^{2^k} - \alpha \in \mathbb{F}_p[X]$ is irreducible for any $k \geq 0$ (the case $p = 1 \bmod 4$ can be accommodated along the same lines, by first moving to a degree-2 extension). This allows

us to define the polynomials

$$\begin{aligned} &\vdots \\ T_{2,i} &= x_i^2 - x_{i-1} \\ &\vdots \\ T_{2,2} &= x_2^2 - x_1 \\ T_{2,1} &= x_1^2 - \alpha, \end{aligned}$$

that define what we called the 2-adic tower. Then, for all $i \geq 1$ we have the isomorphism

$$\mathbb{F}_p[x_1, \ldots, x_i]/\langle T_{2,1}, \ldots, T_{2,i}\rangle \simeq \mathbb{F}_p[x_i]/\langle x_i^{2^i} - \alpha\rangle,$$

which maps x_j to $x_i^{2^{i-j}}$; in this case, this simply amounts to exponent arithmetic. In addition, for this very particular case, push down amounts to decomposing a polynomial $A(x_i)$ into even and odd parts, as $A(x_i) = A_0(x_i^2) + x_i A_1(x_i^2)$, and return $A_0(x_{i-1}) + x_i A_1(x_{i-1})$; lift up is the inverse operation. When these operations show up in more general towers, they are by no means as straightforward.

Computations in quadratic extensions as above were used in the algorithm of [25] that computes the cardinality of the Jacobian of a curve of genus 2, following Schoof's elliptic curve point counting algorithm [45]. The Jacobian is a group, and the point-counting algorithm involves in particular the computation of elements of 2^k-torsion in this group, by means of successive divisions by two. Such a division by two boils down to several arithmetic operations $(+, \times)$, and four square root extractions; thus, the elements we are computing are defined over the 2-adic tower over of $\mathbb{F}_p$.

At the time of writing [25], the authors relied on a variant of the Kaltofen-Shoup algorithm [29] with running time $O(d^{(\omega+1)/2})$ for an extension of degree $d = 2^k$; this was improved in [22], where fast algorithms for square root computation were given for such a tower.

Artin-Schreier extensions. Another important particular case can be highlighted: *Artin-Schreier* towers, corresponding $\ell = p$, that is, to extensions of degrees $p, p^2, \ldots$. Early results for this situation are due to Cantor [11], with a description of polynomials that can play the role of what we call $T_{p,1}, T_{p,2}, \ldots$; another family of such polynomials is given given by Adleman and Lenstra in [1], and was reused by Shoup [47, 48].

This construction is used in an algorithm due to Couveignes [13] for computing *isogenies* between elliptic curves over finite fields. Remembering that elliptic come endowed with a group law, an isogeny is simply defined as a surjective regular map between two elliptic curves that preserves the group law; in cryptology, they are the core of Elkies' improvements to the Schoof point-counting algorithm [3], and show up as well in more recent constructions [43, 51].

Couveignes' algorithm computes isogenies of degree $\sim p^k$ between two elliptic curves E and E'; it relies on the interpolation of a rational function at special points in an Artin-Schreier tower of height $O(k)$ (these points being deduced from the coordinates of points of p^k-torsion in E and E'). Since the required degree p^k can be taken arbitrarily high, we need efficient algorithms to compute the Artin-Schreier tower, but also perform other tasks such as the interpolation; the paper [14] described some algorithms in this direction, with further developments in [18, 19].

4. BUILDING GENERAL TOWERS

Using cyclotomy. To construct a tower for an arbitrary ℓ, a first approach extends directly the construction used in the quadratic case above, using ideas from Kummer theory. We first review algorithms for a similar problem, the construction of irreducible polynomials.

Suppose that $\mathbb{F}_p$ contains an ℓth root of unity, for some prime ℓ; equivalently, ℓ divides $p-1$. In order to construct an irreducible polynomial of degree ℓ in $\mathbb{F}_p$, it is enough to find a non ℓth power, say α. Then, the polynomials $x^{\ell^i} - \alpha$ are all irreducible, as was the case for quadratic extensions.

When there is no such root of unity, we can adjoin one. We follow here the approach of [47, 48]: first, compute an irreducible factor P of the ℓth cyclotomic polynomial in $\mathbb{F}_p[x]$ (call r its degree), and replace our base field $\mathbb{F}_p$ by $\mathbb{K}_0 = \mathbb{F}_p(y_0)$, where $P(y_0) = 0$. Then, proceed as in the previous case: picking a random element α in $\mathbb{K}_0$ that is not an ℓth power, we build the extension $\mathbb{K}_1 = \mathbb{K}_0(y_1)$, with $y_1^\ell = \alpha$; we can build further levels similarly.

In order to deduce an irreducible polynomial over $\mathbb{F}_p$, we compute the trace $T_{\mathbb{K}_1/\mathbb{F}_{p^\ell}}(y_1)$; its minimal polynomial over $\mathbb{F}_p$ has degree ℓ [47]. Essentially the same idea is used by de Smit and Lenstra [21], using Gauss periods to descend from $\mathbb{K}_1$.

To build the tower itself, we continue the construction above $\mathbb{K}_0$ and descend over $\mathbb{F}_p$, as illustrated in Figure 2 (which is borrowed from [16]). That reference also gives a fast algorithm for lift up and push down, allowing us to perform all conversions at level i using $O^{\tilde{}}(\ell^{i+2})$ operations in $\mathbb{F}_p$ (for data of size ℓ^i).

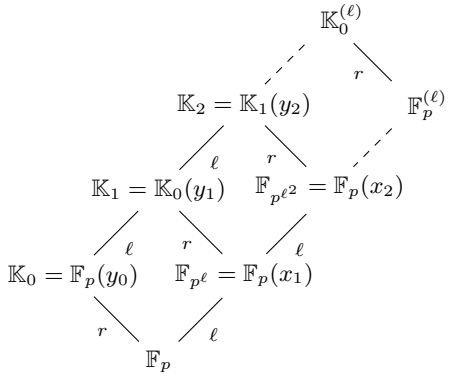

Figure 2: The ℓ-adic towers over $\mathbb{F}_p$ and $\mathbb{K}_0$.

Couveignes and Lercier's algorithm. A second workaround when roots of unity are missing is to use elliptic curves instead: this principle underlies some very well known constructions, such as Lenstra's ECM factoring algorithm [36], that is derived Pollard's $p-1$ method, as well as some other ones, such as Chudnovsky and Chudnovsky's "Fast Elliptic Number Theoretic Transform" [12], derived from the usual Fast Fourier Transform.

In the case at hand, Couveignes and Lercier [15] showed how to construct irreducible polynomials by taking fibers of isogenies (we already saw isogenies in the previous section). We will not repeat the construction here, but simply state that for well-chosen curves (with conditions on their cardinality, as often for such constructions), they showed that

some particular fibers indeed give irreducible polynomials of degrees of the form ℓ^i. Note the analogy with the previous case, where we were looking at fibers of the ℓth power map.

In [16], we showed how this idea carries over to defining a whole tower; that is, how to build all polynomials defining the tower and do lift up and push down in good complexity, $O^{\sim}(\ell^i)$ operations in degree ℓ^i.

5. COMPOSITA

Finally, we look at the question of building general extensions. The basic idea already appears in [9] and was used in [47]: if a and b are elements of respective degrees $L = \ell^i$ and $M = m^j$ over $\mathbb{F}_p$, with ℓ and m coprime, then both $a+b$ and ab have degree $\ell^i m^j$.

This result was first put to use in order to compute irreducible polynomials — namely, starting from the minimal polynomials of a and b, say F and G, it is enough to compute the minimal polynomials of either $a + b$ or ab. These polynomials are often called the *composed sum* and *composed product* of F and G; they are written

$$F \oplus G = \prod_{\alpha,\beta} \big(x - (\alpha + \beta)\big) \quad \text{and} \quad F \otimes G = \prod_{\alpha,\beta} (x - \alpha\beta),$$

the products running over all the roots α of F and β of G, and have degree $D = LM$. The construction of the composed product is often credited to Selmer [46].

The polynomials $F \oplus G$ and $F \otimes G$ can be computed as resultants, namely as

$$\begin{aligned} (F \oplus G)(x) &= \mathrm{res}_y(F(x-y), G(y)) \\ (F \otimes G)(x) &= \mathrm{res}_y(y^m F(x/y), G(y)), \end{aligned} \tag{2}$$

but it is unknown how to obtain a quasi-linear running time from these expressions. Faster algorithms were given in [6], using conversions to and from Newton sums for all these polynomials, for fields of large enough characteristics; for general cases, see the discussions in [7, 5].

The last operation we consider is embedding, and more generally change of basis. Indeed, the discussion above implies the existence of isomorphisms

$$\begin{array}{ccc} \mathbb{F}_p[x,y]/\langle F, G\rangle & \to & \mathbb{F}_p[z]/\langle F \oplus G\rangle \\ x + y & \mapsto & z \end{array}$$

and

$$\begin{array}{ccc} \mathbb{F}_p[x,y]/\langle F, G\rangle & \to & \mathbb{F}_p[z]/\langle F \otimes G\rangle \\ xy & \mapsto & z. \end{array}$$

As of now, we still do not know how to perform these operations efficiently. Of course, it is possible to use the extension of Kedlaya and Umans' algorithm mentioned before; this leads to algorithms with almost linear running time in a boolean model, but as we said above, they are difficult to put into practice. In an algebraic model, the paper [17] give two different algorithms, but none of them is quasi-linear.

6. REFERENCES

[1] L. M. Adleman and H. W. Lenstra. Finding irreducible polynomials over finite fields. In *STOC'86*, pages 350–355, New York, NY, USA, 1986. ACM.

[2] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *J. Symb. Comput.*, 28(1,2):45–124, 1999.

[3] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography*. Cambridge University Press, New York, NY, USA, 1999.

[4] W. Bosma, J. Cannon, and A. Steel. Lattices of compatibly embedded finite fields. *J. Symb. Comput.*, 24(3-4):351–369, 1997.

[5] A. Bostan, M. F. I. Chowdhury, J. van der Hoeven, and É. Schost. Homotopy techniques for multiplication modulo triangular sets. *J. Symb. Comput.*, 46(12):1378-1402, 2011.

[6] A. Bostan, P. Flajolet, B. Salvy, and É. Schost. Fast computation of special resultants. *J. Symb. Comput.*, 41(1):1–29, 2006.

[7] A. Bostan, L. González-Vega, H. Perdry, and É. Schost. From Newton sums to coefficients: complexity issues in characteristic p. In *MEGA'05*, 2005.

[8] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *ISSAC'03*, pages 37–44. ACM, 2003.

[9] J. V. Brawley and L. Carlitz. Irreducibles and the composed product for polynomials over a finite field. *Discrete Math.*, 65(2):115–139, 1987.

[10] R. P. Brent and H.-T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25(4):581–595, 1978.

[11] D. G. Cantor. On arithmetical algorithms over finite fields. *J. Combin. Theory Ser. A*, 50(2):285–300, 1989.

[12] D. V. Chudnovsky and G. V. Chudnovsky. Computational problems in arithmetic of linear differential equations. Some Diophantine applications. In *Number theory (New York, 1985/1988)*, volume 1383 of *Lecture Notes in Math.*, pages 12–49. Springer, Berlin, 1989.

[13] J.-M. Couveignes. Computing ℓ-isogenies using the p-torsion. In *ANTS-II*, pages 59–65, London, UK, 1996. Springer-Verlag.

[14] J.-M. Couveignes. Isomorphisms between Artin-Schreier towers. *Math. Comp.*, 69(232):1625–1631, 2000.

[15] J.-M. Couveignes and R. Lercier. Fast construction of irreducible polynomials over finite fields. *Israel J. Math.*, 194(1):77–105, 2013.

[16] L. De Feo, J. Doliskani, and É. Schost. Fast algorithms for ℓ-adic towers over finite fields. In *ISSAC'13*, pages 165–172. ACM, 2013.

[17] L. De Feo, J. Doliskani, and É. Schost. Fast arithmetic for the algebraic closure of finite fields. In *ISSAC '14*, pages 122–129. ACM, 2014.

[18] L. De Feo and É. Schost. Fast arithmetics in Artin-Schreier towers over finite fields. *J. Symb. Comput.*, 47(7):771–792, 2012.

[19] L. De Feo. Fast algorithms for computing isogenies between ordinary elliptic curves in small characteristic. *Journal of Number Theory*, 131(5):873–893, May 2011.

[20] B. de Smit and H. W. Lenstra. Standard models for finite fields: the definition, 2008.

[21] B. de Smit and H. W. Lenstra. Standard models of finite fields. In G. Mullen and D. Panario, editors, *Handbook of Finite Fields*. CRC Press, 2013.

[22] J. Doliskani and É. Schost. Computing in degree

2^k-extensions of finite fields of odd characteristic. *Des. Codes Cryptogr.*, to appear.

[23] J. von zur Gathen. Irreducible polynomials over finite fields. In *Foundations of Software Technology and Theoretical Computer Science*, volume 241 of *Lecture Notes in Computer Science*, pages 252–262. Springer Berlin Heidelberg, 1986.

[24] J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Comput. Complexity*, 2:187–224, 1992.

[25] P. Gaudry and É. Schost. Point-counting in genus 2 over prime fields. *J. Symb. Comput.*, 47(4):368âĂŞ400, 2012.

[26] X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *Journal of Complexity*, 14(2):257–299, 1998.

[27] M. Kalkbrener. A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. *J. Symb. Comput.*, 15:143–167, 1993.

[28] E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. *Math. Comp.*, 67(223):1179–1197, 1998.

[29] E. Kaltofen and V. Shoup. Fast polynomial factorization over high algebraic extensions of finite fields. In *ISSAC'97*, pages 184–188. ACM, 1997.

[30] K. S. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. *SICOMP*, 40(6):1767–1802, 2011.

[31] S. Lang. *Algebra*. Springer, 3rd edition, January 2002.

[32] D. Lazard. A new method for solvong algebraic systems of positive dimension. *Disc. Appl. Math.*, 33:147–160, 1991.

[33] D. Lazard. Solving zero-dimensional algebraic systems. *J. Symb. Comput.*, 13:147–160, 1992.

[34] F. Le Gall. Powers of tensors and fast matrix multiplication. In *ISSAC'14*, pages 296–303. ACM, 2014.

[35] R. Lebreton. Relaxed Hensel lifting of triangular sets. *J. Symb. Comput.*, 68, Part 2:230–258, 2015.

[36] H. W. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.

[37] X. Li, M. Moreno Maza, and É. Schost. Fast arithmetic for triangular sets: from theory to practice. In *ISSAC'07*, pages 269–276. ACM, 2007.

[38] M. Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report TR 4/99, NAG Ltd, Oxford, UK, 1999. `http://www.csd.uwo.ca/ moreno/`.

[39] C. Pascal and É. Schost. Change of order for bivariate triangular sets. In *ISSAC'06*, pages 277–284. ACM, 2006.

[40] A. Poteaux and É. Schost. Modular composition modulo triangular sets and applications. *Comput. Complexity*, 22(3):463–516, 2013.

[41] A. Poteaux and É. Schost. On the complexity of computing with zero-dimensional triangular sets. *J. Symb. Comput.*, 50:110–138, 2013.

[42] J. F. Ritt. *Differential Algebra*. Dover Publications, 1966.

[43] A. Rostovtsev and A. Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145, April 2006.

[44] A. Scheerhorn. Trace- and norm-compatible extensions of finite fields. *Appl. Algebra Engrg. Comm. Comput.*, 3(3):199–209, 1992.

[45] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44(170):483–494, 1985.

[46] E. S. Selmer. *Linear recurrence relations over finite fields*. Department of Mathematics, University of Bergen, 1966.

[47] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Math. Comp.*, 54:435–447, 1990.

[48] V. Shoup. Fast construction of irreducible polynomials over finite fields. *J. Symb. Comput.*, 17(5):371–391, 1994.

[49] V. Shoup. A new polynomial factorization algorithm and its implementation. *J. Symb. Comput.*, 20(4):363–397, 1995.

[50] I. E. Shparlinski. Finding irreducible and primitive polynomials. *Appl. Algebra Engrg. Comm. Comput.*, 4(4):263–268, 1993.

[51] E. Teske. An elliptic curve trapdoor system. *Journal of Cryptology*, 19(1):115–133, January 2006.

[52] W. T. Wu. On zeros of algebraic equations — an application of Ritt principle. *Kexue Tongbao*, 31:1–5, 1986.