A short introduction to finite fields and their arithmetic

Pierre Karpman

April 8, 2021

1 Prime fields

This section introduces the simplest examples of finite fields, *i.e. prime fields*, and some usual and basic definitions and results about finite fields.

1.1 Fermat-based approach

We start by recalling Fermat's "little" theorem:

Theorem 1 (Fermat's little theorem (FLT)). *Let p be a prime, $a \in [\![1, p-1]\!]$, then $a^{p-1} \equiv 1 \mod p$.*

Proof. Following the strategy of Fermat, the proof is admitted. □

This immediately leads to the fundamental result:

Theorem 2. *The ring $\mathbb{Z}/n\mathbb{Z}$ is a field iff. n is prime.*

Proof. If $n > 1$ is not prime, then $\exists\, a, b \in [\![2, n-1]\!]$ s.t. $n = ab$ so a is a zero divisor modulo n and hence not invertible.

If n is prime, then by Theorem 1 every $a \in [\![1, n-1]\!]$ is invertible modulo n. □

Note that the above proof is essentially algorithmic in nature in that it provides a way to compute inverses modulo p and then to divide in $\mathbb{Z}/p\mathbb{Z}$, which is the only one of the basic operations $(+,-,\times,\div)$ that is non trivial. In all of the following we will use the notation $\mathbb{F}_p$ to denote the *prime field* $\mathbb{Z}/p\mathbb{Z}$, and $[n]x$ where n is an integer and x the element of an additive group to denote the *scalar multiplication* of x by n, *i.e.* the sum of n copies of x together, or $\sum_{i=1}^{n} x$.

Definition 3 (Characteristic). The *characteristic* of a ring $\mathbb{A}$, denoted char($\mathbb{A}$), is the least integer n s.t. $\forall\, x \in \mathbb{A}$, $[n]x = 0$, if it exists, and zero otherwise.

An equivalent definition for fields is to only consider the multiplicative identity 1 instead of an arbitrary x in the above statement; indeed since every non-zero $x \in \mathbb{A}$ is by definition equal to $x \times 1$, then by distributivity one has that $[n]x = [n](x \times 1) = x \times ([n]1)$ which is zero iff. $[n]1 = 0$.

Exercise 1. Show that char($\mathbb{F}_p$) = p.

Exercise 2 (Prime fields arithmetic). Let p be a prime. We write R the cost in elementary bit operations to reduce $x \in [\![-p^2, p^2]\!]$ modulo p (*i.e.* to compute the positive remainder of the division of x by p).

1. How many bits are necessary and sufficient to represent all the elements of $\mathbb{F}_p$?
2. Give an algorithm that computes the sum or the subtraction of two elements of $\mathbb{F}_p$. What is its cost in elementary bit operations?

3. Same question for the product, assuming that schoolbook integer multiplication is used.

4. Same question for computing the inverse of an element using the FLT. Give the cost first in function of the cost M of products in $\mathbb{F}_p$, then in elementary bit operations.

In the last question, one may rely on a "fast exponentiation" algorithm, for which we give an example in Fig. 1.

```
/* Input:
   - x, an element of a group G (with multiplicative notation)
   - b, an integer > 1
   Output:
   - x**b in G
*/
fastexp(x, b)
{
    res = 1;
    acc = x;
    while(b > 0)
    {
        if (b % 2 == 1)
            res = res * acc; // product in G
        acc = acc * acc;     // squaring in G
        b = b / 2;           // integer division
    }
    return res;
}
```

Figure 1: Fast exponentiation in a group.

While it is clear that the additive group of $\mathbb{F}_p$ is cyclic, this is maybe less so for its multiplicative group. It is in fact the case that the multiplicative group of any finite field —and prime fields in particular— is cyclic. We state this in this particular case and without proof as:

Theorem 4. *The multiplicative group $\mathbb{F}_p^\times$ of $\mathbb{F}_p$ is cyclic. Any of its generators is called a* primitive *element of $\mathbb{F}_p$.*

1.2 Euclide-based approach

Given the importance of finite fields and of the ring $\mathbb{Z}/n\mathbb{Z}$ in general, we wish to reprove some of the above results using Euclid's extended algorithm as a basis instead of the FLT.

Theorem 5 (Extended Greatest Common Divisor (XGCD))**.** *Let $a, b \in \mathbb{Z}\backslash\{0\}$, then $\exists\, u, v, d \in \mathbb{Z}$ s.t. $ua + vb = d = \gcd(a, b)$, where d is the* greatest common divisor *of a and b,* i.e. *the largest positive integer that divides both a and b.*

Proof. We first show that if u, v, d exist s.t. $ua + vb = d$ and d divides both a and b, then d is indeed $\gcd(a, b)$. Assume instead that $d' > d$ divides both $a = a'd'$ and $b = b'd'$, then dividing both sides by d' we get $ua' + vb' = d/d'$ where the left-hand side is integral and the right-hand side is not, which is a contradiction, so such a d' cannot exist.

It remains to show that such a triplet exists, which we do algorithmically thanks to Euclid's algorithm. We start by proving the "standard" non-extended algorithm that only computes d and address the computation of u and v next.

W.l.o.g. we assume that $a \geqslant b$ and use the fact that $d = \gcd(a, b) = \gcd(b, a \mod b)$, where $x \mod y$ denotes here the positive remainder of the division of x by y. Indeed one has that $a = a'd$ and $b = b'd$ for some a', b', hence letting $a = qb + r$ we have $r = d(a' - qb')$ so $d \,|\, r = a \mod b$; conversely if there were $d' > d$ that divided both r and b, then one would have $a = d'(qb'' + r'')$ for some b'', r'', so d' would also divide a which contradicts the fact that $d = \gcd(a, b)$. We then use this equality repeatedly to compute the sequence $r_0 := a, r_1 := b, r_2 := r_0 \mod r_1, \ldots, r_i := r_{i-2} \mod r_{i-1}, \ldots$. Since the r_i's are positive and decreasing there is an index $k + 1$ s.t. $r_{k+1} = 0$, which is equivalent to the fact that $r_k | r_{k-1}$ which itself implies that $\gcd(r_0, r_1) = \gcd(r_{k-1}, r_k) = r_k$.

We now conclude by showing how to compute u and v. Let $\boldsymbol{T}_1 = \begin{pmatrix} 0 & 1 \\ 1 & -r_0 \div r_1 \end{pmatrix}$, where $x \div y$ denotes here the quotient in the division of x by y, then we have that $\boldsymbol{T}_1 \begin{pmatrix} r_0 & r_1 \end{pmatrix}^t = \begin{pmatrix} r_1 & r_2 \end{pmatrix}^t$. By defining $\boldsymbol{T}_2, \ldots, \boldsymbol{T}_k$ similarly and letting $\boldsymbol{M} = \boldsymbol{T}_k \cdots \boldsymbol{T}_1$ it follows that $\boldsymbol{M} \begin{pmatrix} r_0 & r_1 \end{pmatrix}^t = \begin{pmatrix} r_k & 0 \end{pmatrix}^t$ and the first row of $\boldsymbol{M}$ gives the desired relation. □

Remark 6. The above algorithm (and the notion of GCD) is not restricted to integers and may also for instance be applied to polynomials. There also exist many variants of Euclid's algorithm, some being asymptotically faster than the one presented above.

We may now reprove the second part of Theorem 2 by observing that one may recover an inverse of any $a \in [\![1, p - 1]\!]$ modulo p from $ua + vp = \gcd(a, p) = 1$ as (the possibly reduced) u, since $ua \equiv 1 \mod p$. It is also worth noting that this approach is not restricted to finite fields as it allows to compute the inverse of any element of (say) an arbitrary ring $\mathbb{Z}/n\mathbb{Z}$ as long as it is coprime with n, those elements being indeed exactly the invertible ones.

2 Extensions

In the previous section we have seen how to build and compute in finite fields $\mathbb{F}_p$ with a prime number of elements. We now wish to consider *extensions* of such fields whose number of elements is not necessarily prime any more.

2.1 Cardinality, uniqueness

Before building extension fields, we will state (and partially prove) two important results regarding finite fields in general.

Theorem 7 (Cardinality)**.** *Every finite-field $\mathbb{F}$ has cardinality $q = p^k$ where p is prime and k is a non-zero positive integer.*

Proof. We first show that $\operatorname{char}(\mathbb{F})$ is prime. Since q is finite $\exists\, n \in \mathbb{N}\backslash\{0\}$ s.t. $[n]1 = 0$, so $c := \operatorname{char}(\mathbb{F}) > 0$. Now assume by contradiction that $c = ab$ for some $a, b \in [\![2, c - 1]\!]$. We then have $[ab]1 = 0 = [a]([b]1)$; let $\beta = [b]1 \neq 0$, since we are in a field this latter element must have a multiplicative inverse β^{-1} and it follows that $([a]\beta)\beta^{-1} = [a](\beta\beta^{-1}) = [a]1 = 0$ with $a < c$, which is a contradiction. So c must be a prime p.

To prove the remainder of the statement we will show that $\mathbb{F}$ must have the structure of a finite-dimensional $\mathbb{F}_p$-vector space, which will allow us to conclude.

First let us remark that equiped with the same laws as $\mathbb{F}$, $\mathcal{F}_1 := \{1, [2]1, \ldots, [p-1]1\} \cong \mathbb{F}_p$: indeed it is clear that the $[i]1$, $1 \leqslant i < p$ are distinct, and that $[a]1 \times [b]1 = [ab]1 = [ab$

mod p]1 follows from distributivity and the fact that $\mathrm{char}(\mathbb{F}) = p$. We thereafter identify the *subfield* $\mathcal{F}_1$ of $\mathbb{F}$ with $\mathbb{F}_p$.

Now either $\mathbb{F}_p = \mathbb{F}$ and we are done, or there exists $\beta_2 \in \mathbb{F}$ that is $\mathbb{F}_p$-linearly-independent from $\beta_1 := 1$, *i.e.* that cannot be written as $\lambda_1\beta_1$ for some $\lambda_1 \in \mathbb{F}_p$. In that latter case we consider the set $\mathcal{F}_2$ of linear combinations $\{\lambda_1\beta_1 + \lambda_2\beta_2\}$ with $\lambda_1, \lambda_2 \in \mathbb{F}_p$, which is of size p^2;[*] now either $\mathcal{F}_2 = \mathbb{F}$ and we are done, or we again pick $\beta_3 \in \mathbb{F}$ not of the form $\lambda_1\beta_1 + \lambda_2\beta_2$ to build $\mathcal{F}_3$ of size p^3. It is clear that we can iterate this process until $\mathcal{F}_k = \mathbb{F}$, which must be true for some k since q is finite. At that point one has that every element of $\mathbb{F}$ can be written as the $\mathbb{F}_p$-linear combination of $\{\beta_1, \dots, \beta_k\}$, that two elements can be added "component-wise" in this representation, and that one can rescale an element by an arbitrary scalar in $\lambda \in \mathbb{F}_p$ by multiplying every component; we leave the proof that $\mathbb{F}$ satisfies the remaining axioms of an $\mathbb{F}_p$-vector space to the reader. $\square$

Remark 8. The set $\{\beta_1, \dots, \beta_k\}$ as above is said to form an $\mathbb{F}_p$*-basis* of $\mathbb{F}$, or *a basis of* $\mathbb{F}$ *over* $\mathbb{F}_p$. Note that in general such a basis is not unique.

We only state and do not prove the following fundamental result.

Theorem 9 (Unicity)**.** *Let $\mathbb{F}$ be a finite field with q elements, then every other field with q elements (if any) is isomorphic to $\mathbb{F}$.*

From now on we will denote the unique (up to isomorphism) field with q elements by $\mathbb{F}_q$.

Remark 10. Let $\mathbb{F}$, $\mathbb{F}'$ be two finite fields with q elements, although from Theorem 9 one has that $\mathbb{F} \cong \mathbb{F}'$, $\mathbb{F}$ and $\mathbb{F}'$ may still have different *representations* for their elements and their arithmetic. From a practical point of view this means that even if $\mathbb{F}$ and $\mathbb{F}'$ fundamentally posess the same structure, it might be easier to implement operations in $\mathbb{F}$ than in $\mathbb{F}'$.

2.2 Building extensions from polynomials

The basic idea used to build *extensions* of a prime field such as $\mathbb{F}_2$ is to observe that one may easily define polynomials over any field; for instance polynomials in one indeterminate X form the ring $\mathbb{F}_2[X]$.

Exercise 3. Let $P = X^2 + X + 1, Q = X^4 + X + 1 \in \mathbb{F}_2[X]$, compute $P + P$, $P + Q$, PQ.

There are two obstacles that prevent $\mathbb{F}_2[X]$ from being a finite field: it is infinite, and not all of its non-zero elements are invertible. One may remark that these are also what prevents $\mathbb{Z}$ from being a finite field and thus may try to adapt the strategy yielding the finite fields $\mathbb{Z}/p\mathbb{Z}$ to polynomials.

Fact 11. *Let $P \in \mathbb{F}_q[X]$ be a polynomial of degree $d \geqslant 1$, then $\mathbb{F}_q[X]/\langle P\rangle$ is a finite ring of cardinality q^d.*

Exercise 4. Let $P = X^2 + X + 1, Q = X^4 \in \mathbb{F}_2[X]$, compute the Euclidean division of Q by P. Let $R = X, S = X + 1 \in \mathbb{F}_2[X]$, compute (the class of) RS in $\mathbb{F}_2[X]/\langle P\rangle$, that is, the unique polynomial of degree less than two equal to the remainder of the division of RS by P. Same question where the product is computed in $\mathbb{F}_2[X]/\langle X^2 + X\rangle$; is this latter ring a field?

The last exercise shows that similarly to the fact that $\mathbb{Z}/n\mathbb{Z}$ is a field iff. n is prime, some constraint on P is also necessary for $\mathbb{F}_q[X]/\langle P\rangle$ to possibly be one. This constraint is in fact the same as in the integral case, that is we require P not to have any non-trivial factorisation over $\mathbb{F}_q$.

[*]This is because by definition of β_2, $\{\lambda_1\beta_1\} \cap \{\lambda_2\beta_2\} = 0$.

Definition 12 (Irreducible polynomial)**.** Let P be a polynomial of degree $d \geqslant 1$ with coefficients in some field $\mathbb{K}$, we say that P is *irreducible over* $\mathbb{K}' \supseteq \mathbb{K}$ iff. it cannot be written as the product of two non-constant polynomials with coefficients in $\mathbb{K}'$.

Exercise 5. Is $X^2 + 1$ irreducible over $\mathbb{R}$? What about over $\mathbb{C}$? And over $\mathbb{F}_2$?

We now state the "equivalent" of Theorem 2 for polynomials over finite fields.

Theorem 13. *Let $P \in \mathbb{F}_q[X]$ be a monic polynomial of degree $d \geqslant 1$, then $\mathbb{F}_q[X]/\langle P\rangle$ is a finite field with q^d elements iff. P is irreducible over $\mathbb{F}_q$.*

Proof. If P is not irreducible, then its factors divide zero in $\mathbb{F}_q[X]/\langle P\rangle$ and thus do not admit any inverse in this ring. In the converse case, all of the $q^d - 1$ non-zero polynomials P' of $\mathbb{F}_q[X]$ of degree less than d are coprime with P and thus using Theorem 5 for polynomials one may compute P'^{-1} s.t. $P'P'^{-1} + QP = 1$. □

In the above, we say that a field $\mathbb{F}_{q^d} := \mathbb{F}_q[X]/\langle P\rangle$ is an *extension* of $\mathbb{F}_q$ *of degree* d. One may remark that by construction $\mathbb{F}_{q^d}$ has the structure of a d-dimensional $\mathbb{F}_q$-vector space and that its characteristic is also equal to the one of $\mathbb{F}_q$. More generally given two finite fields $\mathbb{F}$ and $\mathbb{F}'$, we may write $\mathbb{F}'/\mathbb{F}$ to denote the fact that $\mathbb{F}'$ is an extension of $\mathbb{F}$ — conversely this means that $\mathbb{F}$ is a *subfield* of $\mathbb{F}$ — and $[\mathbb{F}' : \mathbb{F}]$ to denote the degree of the extension.

We admit the following result.

Theorem 14. *Let $\mathbb{F}$ be an arbitrary finite field, then for every $d \geqslant 1$ there exists at least one irreducible polynomial of degree d in $\mathbb{F}[X]$.*

To this we add that it is computationally "efficient" to test if a given polynomial is irreducible, and that irreducible polynomials form a dense subset of all the polynomials. It then follows that there exists an efficient randomised procedure that returns an irreducible polynomial of arbitrary degree over any finite field.

We also have the immediate corollary.

Corollary 15. *There is a finite field with q elements iff. $q = p^k$, where p is prime and k is a non-zero positive integer.*

Proof. One direction is given by Theorem 7, the other by Theorems 13 and 14. □

Finally since it is efficient to find an irreducible polynomial of arbitrary degree, it is similarly efficient to compute a *representation* of an arbitrary finite field. Since there are also in general several irreducible polynomials of the same degree, one may define several fields with the same number of elements as, say, $\mathbb{F}_q[X]/\langle P\rangle$, $\mathbb{F}_q[X]/\langle Q\rangle$ with distinct P and Q both of degree d and irreducible over $\mathbb{F}_q$. However by Theorem 9 those two fields would in fact be isomorphic (written $\mathbb{F}_q[X]/\langle P\rangle \cong \mathbb{F}_q[X]/\langle Q\rangle$).

We now give a few examples in the binary case.

Example 16. One may check that $P := X^2 + X + 1$ is irreducible over $\mathbb{F}_2$. Consequently $\mathbb{F}_2[X]/\langle P\rangle$ is a degree-2 extension over $\mathbb{F}_2$, *i.e.* a finite field with 4 elements. Those elements may be represented as polynomials as 0, 1 (the elements of the subfield $\mathbb{F}_2$), X and $X + 1$. The addition between the elements is the addition for polynomials and the multiplication is done modulo P, *i.e.* $X \times X = X + 1$, $X(X + 1) = 1$, $(X + 1)(X + 1) = X$.

Example 17. One may check that $P := X^4 + X + 1$ and $Q := X^4 + X^3 + 1$ are both irreducible over $\mathbb{F}_2$, so one may represent the field $\mathbb{F}_{2^4}$ both as $\mathbb{F}_2[X]/\langle P\rangle$ or $\mathbb{F}_2[X]/\langle Q\rangle$.

Example 18. Let P be as in Example 16, in the ring $(\mathbb{F}_2[X]/\langle P\rangle)[Y]$, the polynomial Y^2+Y+1 is *not* irreducible over $\mathbb{F}_2[X]/\langle P\rangle$ since it factors as $(Y+X)(Y+X+1) = Y^2+XY+Y+XY+X^2+X = Y^2+Y+X^2+X = Y^2+Y+1$. One may however check that $Q := Y^2+XY+1$ is irreducible over the same field, so one may build a degree-2 field extension $(\mathbb{F}_2[X]/\langle P\rangle)[Y]/\langle Q\rangle$. Taken together, the extensions over $\mathbb{F}_2$ and $\mathbb{F}_2[X]/\langle P\rangle$ form an *extension tower* of two extensions of degree 2, which yields an extension of degree $4 = 2\times 2$ over the base field $\mathbb{F}_2$, *i.e.* a representation of $\mathbb{F}_{2^4}$; in other words, $\mathbb{F}_{2^4} \cong \mathbb{F}_2[X,Y]/\langle P,Q\rangle$.

Example 19. Let P, Q be as in Example 17; we wish to explicit an isomorphism φ between $\mathbb{F} := \mathbb{F}_2[X]/\langle P\rangle$ and $\mathbb{F}' := \mathbb{F}_2[Y]/\langle Q\rangle$[†] whose existence is guaranteed by Theorem 9. That is we want to determine $\varphi : \mathbb{F}\to\mathbb{F}'$ s.t. $\forall\, a,b\in\mathbb{F}$ one has $\varphi(a+b) = \varphi(a)+\varphi(b)$ and $\varphi(ab) = \varphi(a)\,\varphi(b)$.

From the definition, it is clear that we need to have $\varphi(0) = 0$ and $\varphi(1) = 1$, where the right-hand side of both equalities "live in $\mathbb{F}'$". To determine the image of φ for the remaining elements we may try to use the fact that the mulitplicative group of a finite field is cyclic. Observing that $\langle X\rangle = \mathbb{F}^\times$ and $\langle Y\rangle = \mathbb{F}'^\times$, we could try to extend φ by taking $\alpha = X$, $\beta = Y$, and $\varphi(\alpha^i) = \beta^i$ for all i. For any non-zero a,b one gets $\varphi(ab) = \varphi(\alpha^i\alpha^j) = \varphi(\alpha^{i+j}) = \beta^{i+j} = \beta^i\beta^j = \varphi(a)\,\varphi(b)$ for some i,j. Defined thusly φ would indeed be a group homorphism $\mathbb{F}^\times\to\mathbb{F}'^\times$, however it would *not* be one for the additive group; this is not surprising since here φ would essentially be the "identity" mapping, and $\mathbb{F}$ and $\mathbb{F}'$ *do* use a different representation: in particular one requires $\varphi(X^4+X+1) = \varphi(X^4)+\varphi(X)+\varphi(1) = 0$ and taking $\varphi(X) = Y$ yields $Y^4+Y+1 = Y^3+Y \neq 0$. More generally, if we define the *minimal polynomial* $\min_{\mathbb{K}'}(a)$ over $\mathbb{K}'$ of an element $a\in\mathbb{K}$ to be the monic polynomial $\sum_{i=0}^{d}\pi_iZ^i\in\mathbb{K}'[Z]$ of least degree that annihilates a (i.e. s.t. $\sum_{i=0}^{d}\pi_ia^i = 0$) we have here the necessary condition that $\forall a, \min_{\mathbb{F}_2}(\varphi(a)) = \min_{\mathbb{F}_2}(a)$.

To show that adding this condition is also sufficient we first remark that if $\deg(\min(a)) = k$[‡] then k is also the rank of $\langle a^i\rangle_{i\in\mathbb{N}}$ as an $\mathbb{F}_2$-vector space. In our particular case the dimension of $\mathbb{F}$ over $\mathbb{F}_2$ is 4, so we have that if $k=4$ then every element of $\mathbb{F}$ can be written as an $\mathbb{F}_2$-linear combination of $1 = a^0$, a, a^2 and a^3. Suppose now that we have $\varphi(\alpha) = \beta$ with primitive α,β and where $\min(\alpha) = \min(\beta)$ a polynomial of degree 4, then $\langle\alpha^i\rangle_{0\leqslant i<4}$ (resp. $\langle\beta^i\rangle_{0\leqslant i<4}$) is a basis of $\mathbb{F}$ (resp. $\mathbb{F}'$) over $\mathbb{F}_2$. Moreover writing $\sum_{i=0}^{4}\pi_iZ^i$ for $\min(\alpha)$ one has that $\alpha^4 = \sum_{i=0}^{3}\pi_i\alpha^i$ and $\beta^4 = \sum_{i=0}^{3}\pi_i\beta^i$; $\alpha^5 = \sum_{i=0}^{3}\pi_i\alpha^{i+1} = \sum_{i=0}^{2}\pi_i\alpha^{i+1} + \pi_3\sum_{i=0}^{3}\pi_i\alpha^i$ and $\beta^5 = \sum_{i=0}^{3}\pi_i\beta^{i+1} = \sum_{i=0}^{2}\pi_i\beta^{i+1} + \pi_3\sum_{i=0}^{3}\pi_i\beta^i$; more generally if $a = \alpha^j = \sum_{i=0}^{3}a_i\alpha^i$ then $\varphi(a) = \beta^j = \sum_{i=0}^{3}a_i\beta^i$ and so φ is a homomorphism for the addition.

To conclude the construction of φ, it is now sufficient to find a suitable primitive element $\beta\in\mathbb{F}'$ whose minimal polynomial equals $\min(\alpha)$ for some primitive $\alpha\in\mathbb{F}$. In our example one can check that $\min(Y^7) = Z^4+Z+1$ and since $7\nmid 15$ its multiplicative order is 15; we then finally define φ as $X\mapsto Y^7$, and all the other points are obtained by using the fact that φ is an isomorphism.

The above Example 19 goes a fair way to actually proving Theorem 9, as a full proof would only need to show in the last step that a suitable primitive element with a prescribed minimal polynomial always exists. We do not provide such a proof but sketch the main steps and arguments.

Consider $\mathbb{F}/\mathbb{F}_q$, $[\mathbb{F}:\mathbb{F}_q] = d$, then every element of $\mathbb{F}$ is a root of $P := X^{q^d}-X$, *i.e.* P *splits* over $\mathbb{F}$. On the other hand one may show that over $\mathbb{F}_q$ the decomposition of P into irreducible factors contains *all* the monic irreducible polynomials of degree dividing d and it follows that those all split over $\mathbb{F}$. To build an isomorphism between $\mathbb{F}$ and another field $\mathbb{F}'$ of the same cardinality, it is sufficient to map a primitive element of $\mathbb{F}$ to one of

[†] We use two names for the indeterminate so as to make it simpler to determine in which representation an element is living.

[‡] We drop the index $\mathbb{F}_2$ from "$\min_{\mathbb{F}_2}$" in the following for the sake of conciseness.

$\mathbb{F}'$ with the same minimal polynomial Q, which is akin to finding a root of Q in $\mathbb{F}'$. One concludes by observing that the minimal polynomial is always monic and irreducible, and always of degree d if the associated element is primitive. Furthermore, the primitivity of an element is entirely determined by its minimal polynomial in the sense that all the primitive elements are exactly the roots of certain *primitive* irreducible polynomials of maximum degree d. Note however that not all irreducible polynomials are primitive, which means that in general not all elements with minimal polynomial of degree d are primitive.

We now give an alternative description of the process used to build extension fields in Theorem 13: one may build a degree-d extension $\mathbb{F}_{q^d}$ of $\mathbb{F}_q$ by adding a root α of a degree-d polynomial P that is irreducible in $\mathbb{F}_q$. If we write $\mathbb{F}_q[\alpha]$ for such a construction, then α corresponds to (the class of) X in $\mathbb{F}_q[X]/\langle P\rangle$; indeed in the latter case the minimal polynomial of X is P, *i.e.* it is one of its roots.

In the case of finite fields the choice of the root used to build the extension $\mathbb{F}_q[\alpha]$ does not matter in the sense that adding *one* root of P to $\mathbb{F}_q$ is equivalent to adding *all* of them; we say that $\mathbb{F}_q[\alpha]$ is the *splitting field* of P.

Example 20. Let $P := X^2 + 1$ and denote one of its roots by i, then the field $\mathbb{C}$ of complex numbers can be built as $\mathbb{R}[i]$, or equivalently $\mathbb{R}[X]/\langle P\rangle$.

In the above Example 20, the resulting field $\mathbb{C}$ is *algebraically closed*, or equivalently it is the *algebraic closure* $\overline{\mathbb{R}}$ of $\mathbb{R}$: every polynomial of $\mathbb{C}[X]$ can be decomposed into linear factors; in particular this means that one cannot build further extensions of $\mathbb{C}$ by adding more roots of irreducible polynomials. In the case of finite fields, we know from Theorem 14 that the algebraic closure $\overline{\mathbb{F}_q}$ of any finite field $\mathbb{F}_q$ is not finite. A more direct (albeit non constructive) proof of this fact is that assuming that $\overline{\mathbb{F}_q}$ is finite, then $1 + \prod_{a\in\overline{\mathbb{F}_q}}(X - a)$ would have no root in $\overline{\mathbb{F}_q}$, which is a contradiction.

3 Arithmetic in binary fields

Our goal is now to discuss some implementation aspects for extension fields, and binary fields in particular. We will mostly focus on computing products, since additions are trivial and inversion can be implemented with an exponentiation (which while sometimes suboptimal will be considered sufficient in our case).

3.1 Data representation and basic operations

Before describing algorithms and their implementation, we need to decide on a suitable way to represent our field elements. Since we know that a binary field $\mathbb{F}_{2^n}$ has the structure of a vector space $\mathbb{F}_2^n$ and that elements of $\mathbb{F}_2$ can be represented by 0 and 1, a natural choice is to represent $x \in \mathbb{F}_{2^n}$ as a vector of $\mathbb{F}_2^n$, itself represented by a binary string of length n. In a programming language, a binary string is often conveniently manipulated as the (unsigned) integer that it represents; that is, one uses the canonical embedding $\{0,1\}^* \hookrightarrow \mathbb{N}$, $b_k \cdots b_1 b_0 \mapsto \sum_{i=0}^{k} b_i 2^i$. We will use a similar convention in our case and often denote elements of $\mathbb{F}_{2^n}$ by an integer $[\![0, 2^n - 1]\!]$; one must be cautious however not to interpret this as the (wrong) fact that $\mathbb{F}_{2^n} \cong \mathbb{Z}/2^n\mathbb{Z}$. More generally, this convention will be used for any element of an $\mathbb{F}_2$-vector space, not necessarily a finite field.

Example 21. Let $\mathbb{F}_{2^4} \cong \mathbb{F}_2[X]/\langle X^4 + X + 1\rangle$, then the element $X^3 + X + 1$ is represented by the binary string 1011, written `0xB`. The irreducible polynomial $X^4 + X + 1$ used in this representation of $\mathbb{F}_{2^4}$ is represented by the binary string 10011, written `0x13`; since this defining polynomial is monic, if its degree is known from the context, a compressed representation as 0011, written `0x3` may be used instead.

The representation of vectors of $\mathbb{F}_2^n$ as unsigned integers goes further than being a compact encoding; basic operations in this representation are also readily available in many instruction sets, and accessible through any decent programming language. We illustrate this in `C` in the case of elements represented in a single 64-bit machine word, but extension to larger or smaller inputs are straightforward.

— The addition of two vectors coincides with the bitwise XOR: `u ^ v;`.

— The pointwise product coincides with the bitwise AND: `u & v;`.

— The Hamming weight of a vector (*i.e.* the number of coordinates where it is non-zero) coincides with the population count. On `x86` processors with the `POPCNT` instruction set extension, this can be computed using the `popcnt` instruction, accessible for instance through the intrinsic `_mm_popcnt_u64`.

— In the more specific case of polynomials of $\mathbb{F}_2[X]/\langle X^{64}+1\rangle$, multiplication by X^i coincides with the left circular shift (or rotation) on 64-bit words.

— In the more specific case of polynomials of $\mathbb{F}_2[X]$, the product coincides with the carryless multiplication. On `x86` processors with the `PCLMULQDQ` instruction set extension, this can be computed using the `pclmulqdq` instruction, accessible for instance through the intrinsic `_mm_clmulepi64_si128`. Also, the quotient (resp. remainder) in the division by X^i coincides with the right logical shift: `p >> i;` (resp. bitwise AND with the string whose i lowest bits are 1: `p & ((1 << i) - 1)`).

3.2 Product in modular polynomial rings: an `xtime` approach

For a field $\mathbb{F}_{2^n}$ represented as $\mathbb{F}_2[X]/\langle P\rangle$ with P some irreducible polynomial of degree n, the product of two field elements may be implemented by exactly using this representation, *i.e.* from the product of two polynomials modulo P. Note also that the same would be true even if P were not irreducible, in the same way that arithmetic in $\mathbb{Z}/p\mathbb{Z}$ is in large part a special case of the one in $\mathbb{Z}/n\mathbb{Z}$. In this section we describe how to implement the product by using an "`xtime`" primitive operation; we will describe an alternative approach in Section 3.4 that uses a different primitive operation.

We start with the useful subcase where one of the operands in the product is the monomial X, resulting in an elementary operation customarily called `xtime`. We take $Q \in \mathbb{F}_2[X]$ of degree at most $n-1$ and wish to compute $Q \times X \mod P$, *i.e.* the unique polynomial U s.t. $\deg(U) < n$, $Q \times X = V \times P + U$. This can be done easily by observing that: 1) if $\deg(Q \times X) < n$ then $U = Q \times X$; 2) if $\deg(Q \times X) \geqslant n$ then it is exactly n so $\deg(V) = 0$; but V is also non-zero and since we are working in $\mathbb{F}_2[X]$ it must be 1, so then $U = Q \times X - P$.

If polynomials are represented as in Section 3.1, the above operation can be implemented efficiently as in Fig. 2.

If the underlying architecture uses two's complement representation for signed integers, the branchless alternative of Fig. 3 may be preferred.

Remark 22. The above `xtime` operation is sometimes described to correspond to the clocking of a *Linear-Feedback Shift Register* (LFSR) in *Galois configuration.* This is following the view that in `xtime`, one *shifts* bits in a register and *linearly* updates the state of the register with the leftmost, thrown-out bit. We will not explore this alternative description in more details, but only mention that LFSR are frequently-used components in hardware systems due to their efficient implementation as circuits.

We are now ready to address the general case, using `xtime` as a subroutine. This essentially follows from distributivity, and using a process similar to the one of Fig. 1.

```c
/* Input:
   - Q, a binary polynomial of degree < 4
   Output:
   - Q*X mod X**4+X+1
*/
uint8_t xtime(uint8_t Q)
{
    if (Q < 8) // the polynomial represented by Q has deg < 3
        return Q << 1;
    else
        return (Q << 1) ^ 0x13;
}
```

Figure 2: Multiplication of Q by X modulo $X^4 + X + 1$.

```c
/* Input:
   - Q, a binary polynomial of degree < 4
   Output:
   - Q*X mod X**4+X+1
*/
uint8_t xtime_bl(uint8_t Q)
{
    uint8_t m = -(Q >> 3); // ~0 if Q >> 3 == 1, 0 otherwise
    return (Q << 1) ^ (m & 0x3);
}
```

Figure 3: Multiplication of Q by X modulo $X^4 + X + 1$, branchless.

The idea is to remark that one can write the product $Q \times R \mod P$ as $Q\sum_{i=0}^{n-1} R_i X^i \mod P$, where $R = \sum_{i=0}^{n-1} R_i X^i$. Then since the R_i's belong to $\{0, 1\}$ one only needs to compute $Q \times X^i$ up to $i = \deg(R)$ using repeated calls to `xtime` and sum those terms for which $R_i = 1$. This is detailed in Fig. 4.

3.3 Multiplication by a constant

One may sometimes be interested solely in the multiplication by a fixed (but arbitrary) constant $\alpha \in \mathbb{F}_{2^n} \cong \mathbb{F}_2[X]/\langle P \rangle$ rather than by an arbitrary element. Although it is of course possible to still use a general multiplication algorithm to do so, this specialisation opens the way to a more dedicated approach. The idea is to observe that multiplication by a constant is a linear operation, and since $\mathbb{F}_{2^n}$ has the structure of an $\mathbb{F}_2$-vector space, for every α there must exist a matrix $\boldsymbol{M}_\alpha \in \mathbb{F}_2^{n\times n}$ implementing $x \mapsto x\alpha$ as $\boldsymbol{x} \mapsto \boldsymbol{x}\boldsymbol{M}_\alpha$, writing $\boldsymbol{x}$ for x to emphasise its structure as a vector and using the right product for $\boldsymbol{M}_\alpha$ (making $\boldsymbol{x}$ a *row* vector).

We first describe $\boldsymbol{M}_X$ before generalising to arbitrary constants. In this case, $\boldsymbol{M}_X$ in fact implements `xtime`; taking the (somewhat arbitrary) convention that the polynomial $\sum_{i=0}^{n-1} Q_i X^i$ be represented as the vector $\begin{pmatrix} Q_0 & \cdots & Q_{n-1}\end{pmatrix}$, then one has:

$$\boldsymbol{M}_X = \begin{pmatrix} \boldsymbol{0}_{n-1} & \boldsymbol{I}_{n-1} \\ \multicolumn{2}{c}{\boldsymbol{p}} \end{pmatrix},$$

with $\boldsymbol{0}_{n-1} \in \mathbb{F}_2^{(n-1)\times 1}$ the all-zero column vector in dimension $n-1$, $\boldsymbol{I}_{n-1} \in \mathbb{F}_2^{(n-1)\times(n-1)}$ the identity matrix, and $\boldsymbol{p} = \begin{pmatrix} P_0 & \cdots & P_{n-1}\end{pmatrix}$ an encoding of the defining polynomial

```c
/* Input:
   - Q, R binary polynomials of degree < 4
   Output:
   - Q*R mod X**4+X+1
*/
uint8_t gf16mul(uint8_t Q, uint8_t R)
{
    uint8_t res = 0;
    uint8_t acc = Q;
    while (R > 0)
    {
        if (R & 1)
            res ^= acc;
        acc = xtime(acc);
        R >>= 1;
    }
    return res;
}
```

Figure 4: Multiplication of Q by R modulo $X^4 + X + 1$.

P minus its degree-n coefficient. This matrix is in fact the *companion matrix* of P,[§] a canonical representative of all matrices with minimal polynomial equal to P.

Example 23. In the field $\mathbb{F}_{2^4} \cong \mathbb{F}_2[X]/\langle X^4 + X + 1\rangle$, one has:

$$\boldsymbol{M}_X = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Now to define $\boldsymbol{M}_\alpha$ for an arbitrary element α one uses that the minimal polynomial of X is of degree n, which means that $\boldsymbol{M}_\alpha \in \langle \boldsymbol{M}_{X^i} = \boldsymbol{M}_X^i \rangle_{0 \leqslant i < n}$. Concretely, writing $\alpha = \sum_{i=0}^{n-1} \alpha_i X^i$, one simply has $\boldsymbol{M}_\alpha = \sum_{i=0}^{n-1} \alpha_i \boldsymbol{M}_{X^i}$.

Example 24. Continuing Example 23, take $\alpha = X^3 + X + 1$, then:

$$\boldsymbol{M}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{M}_{X^3} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad \boldsymbol{M}_\alpha = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

From an implementation perspective, the vector-matrix product $\boldsymbol{x}\boldsymbol{M}_\alpha$ can be efficiently implemented in the binary case by using a *broadcast* approach. The idea is to remark that the result is the sum of the rows of $\boldsymbol{M}_\alpha$ for which the corresponding coordinate in $\boldsymbol{x}$ is non-zero, and this can be computed efficiently if $\boldsymbol{M}_\alpha$ is stored in row-major format. We illustrate this in Fig. 5, in dimension 64.

We conclude with the remark that the matrix $\boldsymbol{M}_\alpha$ of the multiplication by α may also be used to reduce the computation of inverses in $\mathbb{F}_{2^n}$ to linear algebra in $\mathbb{F}_2^{n \times n}$. Indeed it is clear that $\boldsymbol{M}_\alpha^{-1} = \boldsymbol{M}_{\alpha^{-1}}$, and the only potential difficulty in using this approach as a general inversion algorithm would be to recover α^{-1} from $\boldsymbol{M}_\alpha^{-1}$ (or in fact α from $\boldsymbol{M}_\alpha$

[§]Or rather, one of the four possible definitions for the companion matrix, since this particular one reflects our ordering for the coefficients of polynomials and the fact that we multiply on the right.

```c
/* Input:
   - x, a binary vector of dimension 64
   - M, a binary 64*64 matrix in row-major format
   Output:
   - x*M
*/
uint64_t mmul64(uint64_t x, uint64_t M[64])
{
    uint64_t res = 0;
    for (int i = 0; i < 64; i++)
    {
        uint64_t m = -(x & 1);
        res ^= m & M[i];
        x  >>= 1;
    }

    return res;
}
```

Figure 5: Vector-matrix multiplication in $\mathbb{F}_2^{64\times 64}$.

in general). However this latter task is made rather easy by observing that due to the special structure of $\boldsymbol{M}_X$, the only non-zero coefficient in the first row of $\boldsymbol{M}_{X^i}$ is in the i^{th} coordinate (with indices starting from zero); it is thus straightforward to recover an expression of α in the basis $\langle X^i\rangle_{0\leqslant i<n}$. In fact this observation tells us that if what we wish to compute is α^{-1} rather than its full multiplication matrix, it is in fact enough to compute only the first row of $\boldsymbol{M}_\alpha^{-1}$.

Example 25. Continuing Example 24, we compute

$$\boldsymbol{M}_{\alpha^{-1}} = \boldsymbol{M}_\alpha^{-1} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix},$$

whose first row reveals that $\alpha^{-1} = X^2 + 1$.

3.4 Product in modular polynomial rings: a carryless approach

It is clear that to compute a product in a modular ring, one may compose the computation of the product in the corresponding “full” (*i.e.* not modular) ring with a reduction step. The goal of this section is to describe such a reduction algorithm for polynomials (*i.e.* which on input P, Q computes the unique polynomial R s.t. $\deg(R) < \deg(P)$ and $R \equiv Q \mod P$) that uses full (*i.e.* not modular) polynomial multiplication and addition as primitive operations; one also needs to compute quotients and remainders of divisions by monomials X^d, but those are easy since they only consist in keeping the monomials of degree more and less than d respectively, as already remarked in Section 3.1. This reduction algorithm is then especially useful if one is able to compute products efficently, which is for instance the case in $\mathbb{F}_2[X]$ if an instruction such as `PCLMULQDQ` is available; in that case both the full product and the reduction will essentially rely on this instruction. We will also see that, justifying Remark 10, the cost of this reduction algorithm will depend on the “shape” of the reduction polynomial P: this means that for a fixed field (up to isomorphism), the implementation of arithmetic using this approach will be more efficient in some representations than in others.

The algorithm is defined in Fig. 6. To prove its cost and its correctness, we will use an invariant on $\mathtt{A}+\mathtt{B}$ and a variant on $\mathtt{B}$. Namely, we have that $\mathtt{A}+\mathtt{B}$ is always congruent to Q modulo P, and that the degree of $\mathtt{B}$ decreases by at least $d-d_u \geqslant 1$ at every iteration (where $d_u = \deg(\mathtt{U})$) until its degree becomes less than d, at which point the algorithm terminates in the next iteration at the latest. It follows that the algorithm returns the correct result after at most $(\deg(Q) - d)/(d - d_u)$ iterations, and it will then be more efficient when d_u is small. Now to prove the invariant $\mathtt{A} + \mathtt{B} \equiv Q \mod P$ we observe that this is initially true; also $\deg(\mathtt{A}) < d$ since $\mathtt{A}$ is initially zero and one only adds to it polynomials of degree less than d. Then the previous equivalence can be written as $\mathtt{A} + \mathtt{B}_{lo} + \mathtt{B}_{hi} \equiv Q \mod P$, where $\mathtt{B}_{lo}$ (resp. $\mathtt{B}_{hi}$) denotes the terms of $\mathtt{B}$ of degree at most $d-1$ (resp. degree at least d), *i.e.* $\mathtt{B}_{lo}$ = `B % X**d`, $\mathtt{B}_{hi}$ = `X**d(B / X**d)`, and the conclusion follows from the fact that $\mathtt{U} \equiv$ `X**d` $\mod P$. Finally the variant on `B` comes from the fact that the degree of `B / X**d` is at least d less than the one of `B` when $\deg(\mathtt{B}) \geqslant d$ and zero otherwise, and that the one of `U` is d_u by definition.

```
 1    /* Input:
 2       - P, Q; deg(Q) = d
 3       Output:
 4       - R ~ Q mod P, deg(R) < d
 5    */
 6    REDBL(P, Q)
 7    {
 8        A = 0;
 9        B = Q;
10        U = X**d % P;
11        while (B > 0)
12        {
13            A += B % X**d;
14            B  = (B / X**d)*U;
15        }
16        return A;
17    }
```

Figure 6: Reduction of Q modulo P, using a Barrett-like approach.

We now give in Fig. 7 an example of an implementation of the above algorithm in `C`, using `PCLMULQDQ` accessed from the `_mm_clmulepi64_si128` intrinsic. More precisely, this is a full implementation of the multiplication in $\mathbb{F}_{2^{64}} \cong \mathbb{F}_2[X]/\langle X^{64} + X^4 + X^3 + X + 1\rangle$ that uses the above for the reduction step. In some more details, lines 8 and 9 set the low 64 bits of two 128-bit registers to the binary representation of the operands Q and R respectively, while line 10 does it for the representation of X^{64} modulo the chosen irreducible polynomial, *i.e.* $X^4 + X^3 + X + 1$. Line 12 assigns to `b` the result of the carryless (*i.e.* "binary polynomial") multiplication of `q` and `r`. This is a polynomial of degree at most 126, which then fits into 128 bits. The last operand `0x00` means here that the two inputs of degree at most 63 are to be found in the 64 low bits of the first two operands. Line 15 performs a similar computation between `b` and `u`, but this time with last operand `0x01`, meaning that the first polynomial is defined by the 64 *high* bits of `b`, *i.e.* the terms of degree above 64 then divided by X^{64}; that is, this line corresponds to the computation of line 14 in Fig. 6. Note that at this point, the polynomial represented by `b` has degree at most $66 = 126 - (64 - 4)$. Line 16 adds the full result to `a` that already contains the previous value of b (however only the 64 low bits of this register, corresponding to the terms of degree less than 64, will eventually be used). Line 17 proceeds as line 15; at this point the polynomial represented by b has degree at most 6, so we know that another iteration would result in the zero polynomial and there would be nothing more to add to a. This means that the fully-reduced result is to be found in the 64 low bits of a after line 18, which are then returned in line 20. We conclude with two remarks: first the implementation in Fig. 7 runs in "constant time", in the sense that the amount of work does not depend on the input (in particular it always does the maximum work needed to fully reduce the product, which may be more than what is sometimes needed); this may be useful in contexts when one does not want the computation time to leak information about the operands. Second, we again emphasize that this running time was kept low by using an irreducible polynomial s.t. d_U is particularly small.

```
 1  /* Input:
 2     - Q, R binary polynomials of degree < 64
 3     Output:
 4     - Q*R mod X**64 + X**4 + X**3 + X + 1
 5   */
 6  uint64_t gf2_64mul(uint64_t Q, uint64_t R)
 7  {
 8      __m128i q = _mm_set_epi64x(0, Q);
 9      __m128i r = _mm_set_epi64x(0, R);
10      __m128i u = _mm_set_epi64x(0, 0x1B);
11      __m128i a;
12      __m128i b = _mm_clmulepi64_si128(q, r, 0x00);
13
14      a = b;
15      b = _mm_clmulepi64_si128(b, u, 0x01);
16      a = _mm_xor_si128(a, b);
17      b = _mm_clmulepi64_si128(b, u, 0x01);
18      a = _mm_xor_si128(a, b);
19
20      return _mm_extract_epi64(a, 0);
21  }
```

Figure 7: Multiplication of Q by R modulo $X^{64} + X^4 + X^3 + X + 1$.

3.5 Arithmetic in a recursive representation

In this section we wish to design a multiplication (and addition) algorithm for elements of $\mathbb{F}_{2^{2^n}}$ built from a recursive *Artin-Schreier extension tower* as:

$$\mathbb{F}_{2^{2^n}} \cong \mathbb{F}_2[X_1, \ldots, X_n]/\langle X_i^2 + X_i + \prod_{j<i} X_j\rangle_{1\leqslant i\leqslant n}$$

The algorithm will itself be recursive, which means that we will reduce the multiplication in $\mathbb{F}_{2^{2^n}}$ to multiplication in $\mathbb{F}_{2^{2^{n-1}}}$, etc..

It can be shown that for all $n \geqslant 1$ the *Artin-Schreier polynomial* $X_n^2 + X_n + \prod_{j<n} X_j$ is irreducible in $\mathbb{F}_2[X_1, \ldots, X_n]/\langle X_i^2 + X_i + \prod_{j<i} X_j\rangle_{1\leqslant i\leqslant n-1}$ (where we take the convention that for $n = 1$ the empty product equals 1), so one can build an extension of degree 2 of $\mathbb{F}_{2^{2^{n-1}}} \cong \mathbb{F}_2[X_1, \ldots, X_{n-1}]/\langle X_i^2 + X_i + \prod_{j<i} X_j\rangle_{1\leqslant i\leqslant n-1}$ by adding one indeterminate X_n

and the corresponding polynomial $X_n^2 + X_n + \prod_{j<n} X_j$ to the generators of the quotienting ideal.

Exercise 6. Design an addition and multiplication algorithm for the above extension tower, and analyse their cost. More precisely, answer to the following:

1. How can you concisely represent elements of $\mathbb{F}_2[X_1, \ldots, X_n]$ as vectors of $\mathbb{F}_2^{2^n}$?
2. Give the vector corresponding to $X_1 + X_2 + X_1X_3 + X_2X_3$ when $n = 3$. Same question for $n = 4$.
3. How can you add together two elements of $\mathbb{F}_{2^{2^n}}$ using this embedding? Is this easy to implement on a typical CPU, when vectors are mapped to binary strings?
4. Show how to compute the multiplication of two elements $P, Q \in \mathbb{F}_{2^{2^n}}$ from four[¶] multiplications and one *Nim transform* in $\mathbb{F}_{2^{2^{n-1}}}$ by writing them as $P = P_0 + X_nP_1$, $Q = Q_0 + X_nQ_1$, $P_0, P_1, Q_0, Q_1 \in \mathbb{F}_{2^{2^{n-1}}}$, where the Nim transform in $\mathbb{F}_{2^{2^n}}$ is the linear mapping $\mathrm{NT} : \mathbb{F}_{2^{2^n}} \cong \mathbb{F}_2[X_1, \ldots, X_n]/\langle X_i^2 + X_i + \prod_{j<i} X_j\rangle_{1 \leqslant i \leqslant n} \to \mathbb{F}_{2^{2^n}}$, $P \mapsto P \cdot X_1 \cdots X_n$.
5. Show how to recursively compute the Nim transform in $\mathbb{F}_{2^{2^n}}$ from Nim transforms in $\mathbb{F}_{2^{2^{n-1}}}$.
6. Using the same embedding into $\mathbb{F}_2^{2^n}$ as above, what is the (recursive) expression of the Nim transform as a matrix? (That is, express the matrix $\boldsymbol{A}_n$ of the Nim transform in $\mathbb{F}_{2^{2^n}}$ as a block matrix in function of $\boldsymbol{A}_{n-1}$, where $\boldsymbol{A}_0 := \begin{bmatrix}0\end{bmatrix}$.)
7. What is the cost of this multiplication algorithm in $\mathbb{F}_{2^{2^n}}$ (using either the schoolbook or the Karatsuba algorithm in the above)? How does this compare with the addition? *Hint:* Use the "Master theorem" to analyse the recursivity (https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms)).

Remark 26. Artin-Schreier extension towers play an important role (among others) in additive Fast Fourier Transform algorithms (Cantor, 1989, etc.), especially useful in characteristic two. Conway also used the above tower to define "Nim arithmetic" over the integers (and beyond); notably this allows to endow $\mathbb{N}$ with a field structure.

[¶]Or three when using Karatsuba's algorithm.