Efficient algorithms for some special cases of the polynomial equivalence problem

Neeraj Kayal*

Abstract

We consider the following computational problem. Let $\mathbb{F}$ be a field. Given two n-variate polynomials $f(x_1, .., x_n)$ and $g(x_1, .., x_n)$ over the field $\mathbb{F}$, is there an invertible linear transformation of the variables which sends f to g? In other words, can we substitute a linear combination of the x_i's for each x_j appearing in f and obtain the polynomial g? This problem is known to be at least as difficult as the graph isomorphism problem even for homogeneous degree three polynomials. There is even a cryptographic authentication scheme (Patarin, 1996) based on the presumed *average-case* hardness of this problem.

Here we show that at least in certain (interesting) special cases there is a polynomial-time randomized algorithm for determining this equivalence, if it exists. Somewhat surprisingly, the algorithms that we present are efficient even if the input polynomials are given as arithmetic circuits. As an application, we show that if in the key generation phase of Patarin's authentication scheme, a *random multilinear* polynomial is used to generate the secret, then the scheme can be broken and the secret recovered in randomized polynomial-time.

1 Introduction

We consider the task of understanding polynomials upto invertible linear transformations of the variables. We will say that two n-variate polynomials $f(\mathbf{X})$ and $g(\mathbf{X})$ are equivalent (also sometimes called isomorphic), denoted $f \sim g$ if there exists an invertible linear transformation $A \in \mathbb{F}^{n \times n}$ such that $f(\mathbf{X}) = g(A \cdot \mathbf{X})$. Does the corresponding computational problem of determining whether two given polynomials are equivalent admit an efficient (polynomial-time) algorithm? A naive method for finding such a matrix A would involve solving an appropriate system of polynomial equations but this being a highly nonlinear task yields only an algorithm with exponential running time. No efficient algorithm is known even if the two polynomials are given verbosely as lists of coefficients and the task seems so difficult that there is even a cryptographic protocol based on the presumed average-case hardness of this problem. We wish to emphasize here that in cryptographic applications, the assumption is that even if the two polynomials have constant degree (say degree four) and are given verbosely as a list of coefficients the polynomial equivalence problem is *hard even in the average-case*. Somewhat surprisingly, we show that for certain special cases of this problem (these special cases we discuss and motivate below), the problem admits a polynomial time randomized algorithm even when the input polynomials are given succinctly as arithmetic circuits.

Motivation The following well-known lemma constructively classifies quadratic polynomials upto equivalence.

LEMMA 1.1. **(Structure of quadratic polynomials).** *Let $\mathbb{F}$ be an algebraically closed field of characteristic different from 2. For any homogeneous quadratic polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ there exists an invertible linear transformation $A \in \mathbb{F}^{n \times n}$ and a natural number $1 \leq r \leq n$ such that*

$$f(A \cdot \mathbf{X}) = x_1^2 + x_2^2 + \ldots + x_r^2.$$

Moreover, the linear transformation A involved in this equivalence can be computed efficiently. Furthermore, two quadratic forms are equivalent if and only if they have the same number r of variables in the above canonical representation.

This lemma allows us to understand many properties of a given quadratic polynomial. We give one example.

*Microsoft Research India.

EXAMPLE 1. **Formula size of a quadratic polynomial.** *Let Φ be an arithmetic formula. The size of the formula Φ, denoted $L(\Phi)$ is defined to be the number of multiplication gates in it. For a polynomial f, $L(f)$ is the size of the smallest formula computing f. Then for a homogeneous quadratic polynomial f, we have that*

$$L(f) = \left\lceil \frac{r}{2} \right\rceil,$$

where r is as given by Lemma 1.1.

No generalization of the above example to higher degree polynomials is known. Indeed, no *explicit* family of *cubic* (i.e. degree three) polynomials is known which has superlinear formula-size complexity. One might naïvely hope that an appropriate generalization of Lemma 1.1 to cubic polynomials might shed some light on the formula size complexity of a cubic polynomial. That is, one wants a characterization of cubic polynomials upto equivalence. Despite intensive effort (cf. [MH74, Har75]), no 'explicit' characterization of cubic forms was obtained. In a recent work, Agrawal and Saxena [AS06] 'explained' this lack of progress: they showed that the well-studied but unresolved problem of graph isomorphism reduces to the problem of testing equivalence of cubic forms. A simpler proof of a slightly weaker version of their result is presented in example 2. This means that the polynomial equivalence problem is likely to be very challenging, even when the polynomials are given verbosely via a list of coefficients. In this work, we do not tackle the general polynomial equivalence problem, but rather some special cases of it which are motivated by the desire to present a given polynomial in an "easier way". The "easier" ways of presenting that we look at are motivated by the characterization of quadratic polynomials as given in Lemma 1.1 and example 1. Let us describe these special cases of polynomial equivalence.
The integer r of Lemma 1.1 is referred to in the literature as the rank of the quadratic form. Notice that upto equivalence, it is the smallest number of variables which the given polynomial f depends on. One then asks whether a given polynomial is equivalent to another polynomial which depends on a fewer number of variables. Now the canonical form for a quadratic polynomial is as a sum of squares of linear forms. The natural question for higher degree polynomials then is whether the given polynomial is a sum of appropriate powers of linear forms. i.e whether a given polynomial of degree d is equivalent to

$$x_1^d + x_2^d + \ldots + x_n^d.$$

It should be noted that unlike quadratic forms, not every polynomial of degree $d \geq 3$ can be presented in this fashion. We devise an efficient randomized algorithm for this special case of equivalence testing. We then consider some other classes of polynomials and do equivalence testing for those. In particular, we devise algorithms to test whether the given polynomial is equivalent to an elementary symmetric polynomial. The algorithms that we devise can be generalized quite a bit and these generalizations (which we call polynomial decomposition and polynomial multilinearization) are explained in section 7 of this article. Before we go on let us motivate our consideration of such special cases by obtaining a hardness result for polynomial equivalence.

EXAMPLE 2. *Graph Isomorphism many-one reduces to testing equivalence of cubic polynomials.*

A proof is given in the appendix.

2 Previous work and our results

The mathematical subject of Geometric Invariant Theory is concerned with the properties of polynomial functions which is independent of the choice basis - in other words properties of polynomials which are invariant under linear transformations of the variables. This subject was at the forefront of nineteenth-century mathematics during which time Hilbert and others completely characterized binary forms (i.e. homogeneous polynomials in two variables) upto equivalence. Algorithmically, the problem has received much less attention - an exponential-time algorithm (better than the trivial method though) is given for example in [FP06, PGC98]. It has also been noted that the problem is in NP $\cap$ coAM and its not known to be in NP $\cap$ coNP. In his thesis [Sax06], Saxena notes that the work of Harrison [Har75] can be used to solve certain special cases of polynomial equivalence but the time complexity deteriorates exponentially with the degree. In particular, the techniques of Harrison imply that one can *deterministically* test whether a given polynomial is equivalent to $x_1^d + \ldots + x_n^d$ but the time taken is exponential in the degree d. Here we give a *randomized* algorithm with running time polynomial in n, d and the

size of the input circuit. We also present a new randomized polynomial time algorithm to test whether a given polynomial is equivalent to an elementary symmetric polynomial. Our algorithms generalize somewhat and we obtain efficient randomized algorithms for polynomial decomposition and multilinearization. See theorems 7.2 and 7.1 for the precise statements of these generalizations and the degenerate cases which need to be excluded.

Finally, we mention that there is a cryptographic authentication scheme due to Patarin [Pat96] based on the presumed *average-case* hardness of the polynomial equivalence problem. We describe Patarin's authentication scheme and then show how the algorithms above can be extended to break this scheme in randomized polynomial-time if a *random multilinear* polynomial is chosen during the key-generation phase.

Organization The rest of this article is organized as follows. We fix some notation and terminology in section 3. We then develop some preliminary subroutines in section 4. Thereafter we consider polynomial equivalence in sections 5, 6 and 7. We then discuss Patarin's authentication scheme and discuss how it can be broken in randomized polynomial time if a random multilinear polynomial is chosen during the key-generation phase. We conclude by posing some new problems.

3 Notation

We will abbreviate the vector of indeterminates $(x_1, x_2, \ldots, x_n)$ by $\mathbf{X}$. The set $\{1, 2, \ldots, n\}$ will be abbreviated as $[n]$. We will be consider polynomials in n variables over some field $\mathbb{F}$. For convenience we will henceforth assume that $\mathbb{F} = \mathbb{C}$ although any field of characteristic larger than say d^2 is good enough. A polynomial of degree one is called an affine form. Affine forms whose constant term is zero are called linear forms. We will say that a given polynomial $f(\mathbf{X})$ is a *low-degree* polynomial if its degree is bounded above by a polynomial in the size of the arithmetic circuit computing $f(\mathbf{X})$.
For a linear transformation

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \in \mathbb{F}^{n \times n},$$

we shall denote by $A \cdot \mathbf{X}$ the tuple of polynomials

$$(a_{11}x_1 + \ldots + a_{1n}x_n, \quad \ldots \quad , a_{n1}x_1 + \ldots + a_{nn}x_n).$$

Thus, for a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, $f(A \cdot \mathbf{X})$ denotes the polynomial obtained by making the linear transformation A on the variables in f.
Derivatives. We also set up a compact notation for partial derivatives. Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. Then we shall use the following shorthand:

$$\partial_i f \stackrel{\text{def}}{=} \frac{\partial f}{\partial x_i}.$$

$\partial^k f$ shall denote the set of k-th order partial derivatives of f. Thus $\partial^1 f$, abbreviated as ∂f, shall equal

$$\left\{ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right\}.$$

$\partial^2 f$ is the set

$$\left\{ \frac{\partial^2 f}{\partial x_i \cdot \partial x_j} : 1 \leq i \leq j \leq n \right\},$$

and so on.

4 Preliminaries

Here we develop some useful subroutines. Although some of this is new, the observations here follow fairly easily from known results and techniques.

4.1 Linear dependencies among polynomials In this section we isolate and study a subproblem which is common to many of the problems studied here. We call it the problem of computing linear dependencies among polynomials and denote it by POLYDEP.

DEFINITION 3. *Let* $\mathbf{f}(\mathbf{X}) \stackrel{\text{def}}{=} (f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_m(\mathbf{X})) \in (\mathbb{F}[\mathbf{X}])^m$ *be a vector of polynomials over a field* $\mathbb{F}$*. The set of* $\mathbb{F}$-linear dependencies *in* $\mathbf{f}$*, denoted* $\mathbf{f}^\perp$*, is the set of all vectors* $\mathbf{v} \in \mathbb{F}^m$ *whose inner product with* $\mathbf{f}$ *is the zero polynomial, i.e.,*

$$\mathbf{f}^\perp \stackrel{\text{def}}{=} \left\{(a_1, \ldots, a_m) \in \mathbb{F}^m : a_1 f_1(\mathbf{X}) + \ldots + a_m f_m(\mathbf{X}) = 0\right\}$$

If $\mathbf{f}^\perp$ *contains a nonzero vector, then the* f_i*'s are said to be* $\mathbb{F}$*-linearly dependent.*

The set $\mathbf{f}^\perp$ is clearly a linear subspace of $\mathbb{F}^m$. In many of our applications, we will want to efficiently compute a basis of $\mathbf{f}^\perp$ for a given tuple $\mathbf{f} = (f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))$ of polynomials. Let us capture this as a computational problem.

DEFINITION 4. *The problem of computing linear dependencies between polynomials, denoted POLYDEP, is defined to be the following computational problem: given as input* m *arithmetic circuits computing polynomials* $f_1(\mathbf{X}), \ldots, f_m(\mathbf{X})$ *respectively, output a basis for the subspace* $\mathbf{f}^\perp = (f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))^\perp \subseteq \mathbb{F}^m$*.*

POLYDEP admits an efficient randomized algorithm. This randomized algorithm will form a basic building block of our algorithms

LEMMA 4.1. *Given a vector of* m *polynomials*

$$\mathbf{f} = (f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_m(\mathbf{X}))$$

in which every f_i *is as usual specified by a circuit, we can compute a basis for the space* $\mathbf{f}^\perp$ *in randomized polynomial time.*

See appendix section A.1 for a proof.

4.2 Minimizing Variables In this section we study how to eliminate redundant variables from a polynomial. In the subsequent sections we will assume that the input polynomial has no redundant variables. The problem of minimizing the number of variables in a polynomial upto equivalence was considered earlier by Carlini [Car06]. An efficient algorithm for verbosely represented polynomial (i.e. polynomials given via a list of coefficients) was devised by Carlini [Car06] and implemented in the computer algebra system **CoCoA**. We observe here that this problem admits an efficient randomized algorithm even when the polynomial is given as an arithmetic circuit.

We adopt the following definition from Carlini [Car06]

DEFINITION 5. *Let* $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ *be a polynomial. We will say that* $f(\mathbf{X})$ *is independent of a variable* x_i *if no monomial of* $f(\mathbf{X})$ *contains* x_i*. We will say that* the number of essential variables *in* $f(\mathbf{X})$ *is* t *if we can make an invertible linear* $A \in \mathbb{F}^{(n\times n)*}$ *transformation on the variables such that* $f(A \cdot \mathbf{X})$ *depends on only* t *variables* $x_1, \ldots, x_t$*. The remaining* $(n-t)$ *variables* $x_{t+1}, \ldots, x_n$ *are said to be* redundant *variables. We will say that* $f(\mathbf{X})$ *is* regular *if it has no redundant variables.*

We have:

THEOREM 4.1. *Given a polynomial* $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ *with* m *essential variables, we can compute in randomized polynomial time an invertible linear transformation* $A \in \mathbb{F}^{(n\times n)*}$ *such that* $f(A \cdot \mathbf{X})$ *depends on the first* m *variables only.*

The proof is given in appendix B.

5 Equivalence to sums of d-th powers

Consider the following problem: given a homogeneous polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d, does there exist a linear transformation $A \in \mathbb{F}^{n\times n}$ and constants $a_1, \ldots, a_n \in \mathbb{F}$ such that

$$f(A \cdot \mathbf{X}) = a_1 \cdot x_1^d + a_2 \cdot x_2^d + \ldots + a_n \cdot x_n^d.$$

Equivalently, the problem can be restated as follows: given a homogeneous polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d, determine n independent linear forms $\ell_1, \ldots, \ell_n \in \mathbb{F}[\mathbf{X}]$ and constants $a_1, \ldots, a_n \in \mathbb{F}$ such that

$$f(\mathbf{X}) = a_1 \cdot \ell_1(\mathbf{X})^d + \ldots + a_n \cdot \ell_n(\mathbf{X})^d.$$

We will devise a randomized polynomial-time algorithm that given $f(\mathbf{X})$, computes the constants and the set of linear forms $\ell_1(\mathbf{X}), \ldots, \ell_n(\mathbf{X})$. The key idea involved in this is the Hessian matrix.

DEFINITION 6. *For a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, the Hessian Matrix $H_f(\mathbf{X}) \in (F[\mathbf{X}])^{n\times n}$ is defined as follows.*

$$H_f(\mathbf{X}) \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \cdot \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \cdot \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \cdot \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \cdot \partial x_n} \end{bmatrix}$$

The most interesting property of the hessian matrix of a polynomial is the effect that a linear transformation of the variables has on it.

LEMMA 5.1. *Let $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ be an n-variate polynomial and $A \in \mathbb{F}^{n\times n}$ be a linear transformation. Let $F(\mathbf{X}) \stackrel{\text{def}}{=} f(A \cdot \mathbf{X})$. Then,*

$$H_F(\mathbf{X}) = A^T \cdot H_f(A \cdot \mathbf{X}) \cdot A.$$

In particular,

$$\textsc{Det}(H_F(\mathbf{X})) = \textsc{Det}(A)^2 \cdot \textsc{Det}(H_f(A \cdot \mathbf{X}))$$

Proof. By the chain rule for differentiation we have for all $1 \le i \le n$:

$$\frac{\partial F}{\partial x_i} = \sum_{k=1}^{n} a_{ki} \cdot \frac{\partial f}{\partial x_k}(A \cdot \mathbf{X})$$

Therefore for all $1 \le i, j \le n$:

$$\begin{aligned} \frac{\partial^2 F}{\partial x_i \cdot \partial x_j} &= \sum_{k=1}^{n} a_{ki} \cdot (\sum_{\ell=1}^{n} a_{\ell j} \frac{\partial^2 f}{\partial x_k \cdot \partial x_l}(A \cdot \mathbf{X})) \\ &= \sum_{k\in[n],\ell\in[n]} a_{ki} \cdot \frac{\partial^2 f}{\partial x_k \cdot \partial x_l}(A \cdot \mathbf{X}) \cdot a_{\ell j} \end{aligned}$$

Putting these equations into matrix form immediate gives us the lemma.

Now consider a homogeneous polynomial $f(\mathbf{X})$ of degree $d \ge 3$ which has the property that there exists a linear transformation A of the variables such that

$$f(A \cdot \mathbf{X}) = x_1^d + x_2^d + \ldots + x_n^d.$$

Set $F(\mathbf{X}) \stackrel{\text{def}}{=} x_1^d + x_2^d + \ldots + x_n^d$. Observe that

$$\frac{\partial^2 F}{\partial x_i \cdot \partial x_j} = \begin{cases} 0 & \text{if } i \ne j, \\ d(d-1)x_i^{d-2} & \text{if } i = j. \end{cases}$$

Thus the matrix $H_F(\mathbf{X})$ is a diagonal matrix so that we have

$$\textsc{Det}(H_F(\mathbf{X})) = d(d-1) \cdot \prod_{i=1}^{n} x_i^{d-2}.$$

By the Lemma 5.1 above we get that

$$\textsc{Det}(H_f(\mathbf{X})) = d(d-1) \cdot \textsc{Det}(A)^{-2} \cdot \prod_{i=1}^{n} \ell_i(\mathbf{X})^{d-2},$$

where the $\ell_i(\mathbf{X})$'s are linear forms corresponding to the different rows of the matrix A^{-1}. Let us record this as a lemma.

Lemma 5.2. *For a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d, if*

$$f(\mathbf{X}) = \sum_{i=1}^{n} a_i \cdot \ell_i(\mathbf{X})^d,$$

where $\ell_1(\mathbf{X}), \ldots \ell_n(\mathbf{X})$ are independent linear forms then

$$\textsc{Det}(H_f(\mathbf{X})) = c \cdot \prod_{i=1}^{n} \ell_i(\mathbf{X})^{d-2},$$

where $c \in \mathbb{F}$ is a nonzero constant.

Using Lemma 5.2 and applying unique factorization of polynomials to $\textsc{Det}(H_f(\mathbf{X}))$, we immediately get the following corollary.

Corollary 5.1. *If $\sum_{i\in[n]} x_i^3 = \sum_{i\in[n]} \ell_i(\mathbf{X})^3$, where the ℓ_i's are independent linear forms then there exists a permutation $\pi \in S_n$ such that $\ell_i = \omega^j \cdot x_{\pi(i)}$, where ω is a primitive third root of unity.*

Lemma 5.2 combined with Kaltofen's algorithm for polynomial factoring [Kal89] can be used to devise a randomized polynomial-time algorithm for our problem. We now give the details of this algorithm.

5.1 The algorithm for equivalence to sums of powers We have seen that the determinant of the hessian of a polynomial $f \sim x_1^d + x_2^d + \ldots + x_n^d$ factors into product of powers of the appropriate linear forms. We now give the details as to how this factorization can be used to determine this equivalence.
Input. An n-variate polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d.

Output. A set of independent linear forms $\ell_1(\mathbf{X}), \ldots, \ell_n(\mathbf{X})$ and constants $a_1, \ldots, a_n$ such that

$$f(\mathbf{X}) = a_1 \cdot \ell_1(\mathbf{X})^d + \ldots + a_n \cdot \ell_n(\mathbf{X})^d,$$

if such a set of ℓ_i's exist.

The Algorithm.

1. Compute an arithmetic circuit $C(\mathbf{X})$ which computes $\textsc{Det}(H_f(\mathbf{X}))$.
2. Use Kaltofen's factorization algorithm [Kal89] to factor $C(\mathbf{X})$ in random polynomial time. If it is not the case that
$$C(\mathbf{X}) = \prod_{i=1}^{n} \ell_i(\mathbf{X})^{d-2},$$
where each $\ell_i(\mathbf{X})$ is a linear form then output No such forms. else (by solving a system of linear equations) compute constants $a_1, \ldots, a_n$ such that
$$f(\mathbf{X}) = \sum_{i=1}^{n} a_i \cdot \ell_i(\mathbf{X})^d.$$
If no such constants a_i's exist then output No such forms. Else output $(\ell_1(\mathbf{X}), \ldots, \ell_n(\mathbf{X})), (a_1, \ldots, a_n)$.

6 Equivalence to an elementary symmetric polynomial

The problem that we now tackle is the following — given an arithmetic circuit which computes an n-variate homogeneous polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, is there an invertible linear transformation A such that $f(A \cdot \mathbf{X})$ is the elementary symmetric polynomial of degree d? Recall that the elementary symmetric polynomial of degree d [1] is

$$\mathrm{SYM}_n^d \stackrel{\text{def}}{=} \sum_{S \subseteq [n], |S|=d} \prod_{i \in S} x_i.$$

Basic idea. We now give the basic idea underlying the algorithm. The key thing is to look at the second order partial derivatives $\partial^2(f)$ of the given polynomial f. The dimension $\partial^2(f)$ is a quantity which is invariant under a linear change of variables. For a random polynomial of degree $d \geq 4$, all the second order partial derivatives are linearly independent so that the dimension of $\partial^2(f)$ is the maximum possible, namely $\binom{n+1}{2}$. The nice thing about SYM_n^d is that its a multilinear polynomial and therefore we have

$$\partial_i^2 \mathrm{SYM}_n^d = 0, \quad \text{for all } i \in [n]. \tag{6.1}$$

Since these second order derivatives of f vanish, the dimension of $\partial^2(f)$ is at most

$$\binom{n+1}{2} - n = \binom{n}{2}.$$

More interestingly, these are essentially the only second-order partial derivatives of SYM_n^d which vanish. The following lemma shows that most of the these second order partial derivatives are linearly independent.

LEMMA 6.1. *For $d \geq 4$, we have*

$$\dim\left(\partial^2(SYM_n^d)\right) = \binom{n}{2}.$$

Proof. See [KN97, pp.22–23].

This means that if f is equivalent to SYM_n^d then $\partial^2(f)$ has dimension $\binom{n}{2}$. Indeed our method shows that for any polynomial $f \in \mathbb{F}(\mathbf{X})$ which has the property that $\partial^2(f)$ has dimension $\binom{n}{2}$, we can efficiently determine whether f is equivalent to a multilinear polynomial. When this happens, we also find an invertible matrix A such that $f(A \cdot \mathbf{X})$ is multilinear. Now let

$$g(\mathbf{X}) \stackrel{\text{def}}{=} f(A \cdot \mathbf{X})$$

be multilinear. It will also follow from our proof that this multilinear polynomial $g(\mathbf{X})$ is equivalent to an elementary symmetric polynomial if and only if there is a diagonal matrix B such that

$$g(B \cdot \mathbf{X}) = \mathrm{SYM}_n^d.$$

It is then a relatively easy exercise to determine whether such a diagonal matrix B exists or not.

PROPOSITION 6.1. *Let $g(\mathbf{X})$ be a homogeneous multilinear polynomial of degree d. Assume that there exist $\lambda_1, \ldots, \lambda_n \in \mathbb{F}$ such that*

$$g(\lambda_1 x_1, \ldots, \lambda_n x_n) = SYM_n^d(\mathbf{X}).$$

Then once can efficiently compute the λ_i's.

Sketch of Proof : The λ_i's can be obtained by solving an appropriate set of linear equations in $(\log \lambda_i)$'s. Here is how we obtain the appropriate set of linear equations to solve. We have

$$\begin{aligned} g(1, \ldots, 1, 0, \ldots, 0) &= \mathrm{SYM}_n^d(\lambda_1^{-1}, \ldots, \lambda_d^{-1}, 0, \ldots, 0) \\ &= \lambda_1^{-1} \cdot \lambda_2^{-1} \cdot \ldots \lambda_d^{-1} \end{aligned}$$

[1] SYM_n^d can also be defined as the unique (upto scalar multiples) homogeneous *multilinear* polynomial of degree d in n variables, which is invariant under every permutation of the variables.

Taking logarithm on both sides, we get one such equation:

$$\log \lambda_1 + \ldots + \log \lambda_d = -\log g(1, \ldots, 1, 0, \ldots, 0)$$

Choosing various subsets of size d we can obtain in this manner an adequate number of linearly independent equations. Solving this system of linear equations gives us the logarithm of the λ_i's. □

In the rest of this section, we will assume that $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ is a polynomial that satisfies $\dim(\partial^2(f)) = \binom{n}{2}$. We will tackle the problem of finding an invertible matrix A such that $f(A \cdot \mathbf{X})$ is multilinear, if such an A exists. We will first observe that our problem boils down to finding an "nicer" basis for a given space of matrices. By a "nicer" basis, we will mean a basis consisting of rank one matrices. We then devise an efficient randomized algorithm for the latter problem.

6.1 Reduction to finding a good basis for a space of matrices. We first consider linear transformations of the variables of a polynomial which make the polynomial multilinear. Let

$$\begin{aligned} g(\mathbf{X}) &= f(A \cdot \mathbf{X}) \\ &= f\left(\sum_j a_{1j}x_j, \sum_j a_{2j}x_j, \ldots, \sum_j a_{nj}x_j\right) \end{aligned}$$

be the polynomial obtained by applying the transformation A to the variables in f. Then $\partial_i^2 g = 0$ if and only if

$$(a_{1i}\partial_1 + a_{2i}\partial_2 + \ldots + a_{ni}\partial_n)^2 f = 0.$$

Therefore, if $g(\mathbf{X})$ is multilinear then every column vector of A satisfies

$$(a_1\partial_1 + a_2\partial_2 + \ldots + a_n\partial_n)^2 f = 0,$$

and these n vectors are linearly independent since A is invertible.
We will apply the above observation algorithmically as follows. Given f, we first compute the set

$$\partial^2 f \stackrel{\text{def}}{=} \{\frac{\partial^2 f}{\partial i \cdot \partial j} : i \neq j\}$$

and then using the randomized algorithm for POLYDEP, we obtain a basis for the set of all quadratic differential operators $D(\partial_1, \ldots, \partial_n)$ such that $Df = 0$. Since $\dim(\partial^2(f)) = \binom{n}{2}$ we have $\dim(D(\partial_1, \ldots, \partial_n)) = n$. By the observation above our problem boils down to finding a basis for $D(\partial_1, \ldots, \partial_n)$ such that every quadratic operator in the basis has the following form:

$$(a_1\partial_1 + a_2\partial_2 + \ldots + a_n\partial_n)^2 f = 0.$$

Towards this end, we associate every n-variate quadratic operator D with an $n \times n$ symmetric matrix $\hat{D}$ in the following natural way. Let $D \in \mathbb{F}[\partial_1, \ldots, \partial_n]$ be a quadratic polynomial, where

$$D = \sum_{i \in [n]} \alpha_i \partial_i^2 + \sum_{1 \leq i < j \leq n} \beta_{ij} \partial_i \partial_j.$$

The matrix $\hat{D}$ associated with this operator D is the following:

$$\hat{D} \stackrel{\text{def}}{=} \begin{pmatrix} \alpha_1 & \frac{1}{2}\beta_{12} & \ldots & \frac{1}{2}\beta_{1n} \\ \frac{1}{2}\beta_{12} & \alpha_2 & \ldots & \frac{1}{2}\beta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}\beta_{1n} & \frac{1}{2}\beta_{2n} & \ldots & \alpha_n \end{pmatrix}. \tag{6.2}$$

This way of associating a quadratic differential operator with a symmetric matrix has the following property.

PROPERTY 6.1. *Over an algebraically closed field $\mathbb{F}$ of characteristic different from 2, the quadratic polynomial D is equivalent to a sum of r squares if and only if the corresponding symmetric matrix $\hat{D}$ is of rank r. In particular, the polynomial D is a perfect square if and only if $\hat{D}$ is of rank one.*

This property can be used to reduce our problem to the problem of finding "nice" basis of a given space of matrices. More specifically, our problem boils down to finding a set of rank one matrices that span a given space of matrices. The details are given in the algorithm below. It uses a subroutine which we will describe in the next subsection.

Input. An n-variate polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d.

Output. .

The Algorithm.

1. Given an arithmetic circuit of size s for the polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, use the naive method of computing derivatives to obtain a new circuit of size $O(sn^2)$, whose outputs are the second-order partial derivatives $\partial^2(f)$ of f.

2. Using the randomized algorithm for POLYDEP, obtain
   a basis for $(\partial^2(f))^{\perp}$. Each element in the basis of $(\partial^2(f))^{\perp}$ is a homogeneous quadratic polynomial in $\mathbb{F}[\partial_1, \ldots, \partial_n]$ in the natural way. Let this basis be

   $$\{D_1, \ldots, D_n\} \subset \mathbb{F}[\partial_1, \ldots, \partial_n].$$

3. From $D_1, \ldots, D_n$, obtain the corresponding symmetric matrices $\hat{D}_1, \ldots, \hat{D}_n$. Using the randomized algorithm described below, obtain another basis $\{\hat{E}_1, \ldots, \hat{E}_n\}$ of the vector space generated by $\{\hat{D}_1, \ldots, \hat{D}_n\}$ such that each $\hat{E}_i$ is a rank one symmetric matrix [2], if such a basis exists.

   Their corresponding quadratic polynomials $E_1, \ldots, E_n \subset$
   $\mathbb{F}[\partial_1, \ldots, \partial_n]$ are then perfect squares. Let

   $$E_i = \left(\sum_{j \in [n]} a_{ij} \partial_j \right)^2 .$$

   The matrix $A = (a_{ij})_{i,j \in [n]}$ is then the required linear transformation which makes f multilinear.

We now present an efficient randomized algorithm that given n linearly independent matrices of dimension $n \times n$, finds a basis consisting of rank-one matrices, if such a basis exists. Our proof will also show that such a basis, if it exists, is unique up to scalar multiples and permutations of the basis elements.

6.2 Randomized algorithm for finding a basis consisting of rank-one matrices. We are given n symmetric matrices $\hat{D}_1, \ldots, \hat{D}_n$, and we want to find another basis $\hat{E}_1, \ldots, \hat{E}_n$ of the space generated by the given matrices such that each $\hat{E}_i$ is of rank one. A rank one symmetric matrix is the outer product of a vector with itself. So for each $i \in [n]$, let $\hat{E}_i = \mathbf{v}_i^T \mathbf{v}_i$ where $\mathbf{v}_i \in \mathbb{F}^n$.

LEMMA 6.2. *Suppose that $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{F}^n$ are vectors. Then*

$$\text{DET}(z_1 \mathbf{v}_1^T \cdot \mathbf{v}_1 + \ldots + z_n \mathbf{v}_n^T \cdot \mathbf{v}_n) = z_1 z_2 \ldots z_n \cdot (\text{DET}(V))^2, \tag{6.3}$$

where $V = [\mathbf{v}_1^T \ldots \mathbf{v}_n^T]$ is the matrix whose columns are the $\mathbf{v}_i$'s.

[2] Here we are thinking of matrices as n^2-dimensional vectors

Proof. Let $M(\mathbf{z}) \stackrel{\text{def}}{=} z_1 \mathbf{v}_1^T \cdot \mathbf{v}_1 + \ldots + z_n \mathbf{v}_n^T \cdot \mathbf{v}_n$. Then $\textsc{Det}(M(\mathbf{z}))$ is a polynomial of degree n in the formal variables $z_1, \ldots, z_n$. If $z_i = 0$ then for every setting of the remaining variables, the matrix M is singular because its image is spanned by the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_n$, and is of rank at most $n-1$. Thus z_i divides $\textsc{Det}(M(\mathbf{z}))$ for all $i \in [n]$. Using Chinese remaindering, we have that $\prod z_i$ divides $\textsc{Det}(M(\mathbf{z}))$. Because the degree of $\textsc{Det}(M(\mathbf{z}))$ is n, we have

$$\textsc{Det}(M(\mathbf{z})) = \lambda \prod_{i \in [n]} z_i,$$

for some scalar $\lambda \in \mathbb{F}$. Setting all the z_i's to 1, we get

$$\begin{aligned} \lambda &= \textsc{Det}\left(\sum_{i \in [n]} \mathbf{v}_i^T \cdot \mathbf{v}_i \right) \\ &= \textsc{Det}(V \cdot V^T) \\ &= \textsc{Det}(V)^2. \end{aligned}$$

We thus have $\textsc{Det}(M(\mathbf{z})) = z_1 z_2 \ldots z_n \cdot (\textsc{Det}(V))^2$.

Corollary 6.1. *Let $\hat{D}_1, \ldots, \hat{D}_n \in \mathbb{F}^{n \times n}$ be symmetric matrices. Suppose that there exist vectors $\mathbf{v}_1, \ldots \mathbf{v}_n$ such that*

$$\hat{D}_i = \sum_{j=1}^{n} \alpha_{ij} \mathbf{v}_j^T \cdot \mathbf{v}_j. \tag{6.4}$$

Then

$$\textsc{Det}(z_1 \hat{D}_1 + \ldots z_n \hat{D}_n) = \textit{constant} \cdot \ell_1 \ell_2 \ldots \ell_n,$$

where for all $j \in [n]$, $\ell_j = \sum_{i=1}^{n} \alpha_{ij} z_i$ is a linear form over $z_1, \ldots, z_n$.

Corollary 6.1 suggests an algorithm.

Theorem 6.1. *There exists a randomized polynomial-time algorithm that given n symmetric matrices $\hat{D}_1, \ldots, \hat{D}_n \in \mathbb{F}^{n \times n}$, finds a basis for the space generated by them consisting of matrices of rank one, if such a basis exists.*

Proof. We write down an arithmetic circuit for the polynomial

$$F(z_1, \ldots, z_n) \stackrel{\text{def}}{=} \textsc{Det}(z_1 \hat{D}_1 + \ldots + z_n \hat{D}_n).$$

Then we use Kaltofen's algorithm [Kal89] to factor $F(z_1, z_2, \ldots, z_n)$ in randomized polynomial time. By Corollary 6.1, we can use the linear factors $\ell_1, \ell_2, \ldots, \ell_n$ of this polynomial, which are unique up to scalar multiples and permutations, to solve the equations (6.4), and get the rank one matrices as required.

This completes the description of our algorithm.

7 Generalizations of equivalence testing.

The algorithm presented in the previous section generalizes and we obtain:

Theorem 7.1. **Multilinearization.** *Given a polynomial $f \in \mathbb{F}(\mathbf{X})$ which has the property that $\partial^2(f)$ has dimension $\binom{n}{2}$, we can efficiently determine whether f is equivalent to a multilinear polynomial and if so, find an invertible matrix A such that $f(A \cdot \mathbf{X})$ is multilinear.* [3]

The algorithm for testing equivalence to sum of powers of linear forms also generalizes although certain degenerate cases need to be ruled out.

[3] Notice that if f is equivalent to a multilinear polynomial then $\partial^2(f)$ can have dimension at most $\binom{n}{2}$.

THEOREM 7.2. **Polynomial Decomposition** *There is a randomized polynomial-time algorithm that given an n-variate polynomial $f(\mathbf{X})$ as an arithmetic circuit, finds a decomposition of $f(\mathbf{X})$, if it exists, provided* $\mathrm{DET}(H_f(\mathbf{X}))$ *is a regular polynomial, i.e. it has n variables upto equivalence.*

We do not know whether there exists such an efficient polynomial decomposition algorithm for *all* polynomials. We postpone this generalization to the appendix.

8 Breaking a multilinear variant of Patarin's authentication scheme

We have already noted that the polynomial equivalence problem is at least as difficult as graph isomorphism. However, unlike Graph Isomorphism which is believed to be easy on the average and therefore unsuitable for cryptographic applications, no such heuristic average-case algorithms are known for the polynomial equivalence problem. This, combined with the fact that just like graph isomorphism polynomial equivalence also admits a *perfect zero-knowledge proof* led Patarin [Pat96] to propose an authentication scheme based on this problem. Patarin [Pat96] has proposed an authentication scheme that allows *perfect zero-knowledge* authentication. It is based on the presumed *average-case* hardness of the polynomial equivalence problem. In contrast to some earlier authentication schemes which were based on the presumed average-case hardness of the integer factoring problem or the discrete logarithm problem, the main attraction of the scheme is that the polynomial equivalence problem is not known to have an efficient quantum algorithm, even in the average case.

In the key generation phase of Patarin's scheme, a random n-variate polynomial $f(\mathbf{X})$ of degree d is picked. (For concreteness, we fix d to be 4 in the sequel. The results naturally generalize for higher values of d.) Then two random invertible matrices $A_1, A_2 \in \mathbb{F}^{n\times n}$ are chosen and

$$\begin{aligned} f_1(\mathbf{X}) &:= f(A_1 \cdot \mathbf{X}) \\ f_2(\mathbf{X}) &:= f(A_2 \cdot \mathbf{X}) \end{aligned}$$

are computed. Then f_1, f_2 are made public while the secret is the linear transformation $S := A_1^{-1} \cdot A_2$ which sends f_1 to f_2, i.e. $f_1(S \cdot \mathbf{X}) = f_2(\mathbf{X})$. In the appendix we show that if the polynomial f is chosen to be a *random multilinear* polynomial of degree 4, then the scheme can be completely broken - the secret S can be recovered in randomized polynomial time.

We first describe Patarin's scheme in more detail.

8.1 The authentication scheme of Patarin.

Alice (also sometimes called the prover) holds some secret s and wishes to authenticate herself to Bob (also sometimes called the verifier) by proving that she does indeed hold the secret. However she wishes to do it in such a manner that Bob (and also any eavesdropper) gains no information at all about the secret s itself.

Public: Two n-variate polynomials $f_1(\mathbf{X})$ and $f_2(\mathbf{X})$ over a field $\mathbb{F}$.

Secret: A linear map $A \in \mathbb{F}^{(n\times n)*}$ such that $f_2(\mathbf{X}) = f_1(A \cdot \mathbf{X})$.

Step 1. Alice chooses a random $B \in \mathbb{F}^{(n\times n)*}$ and computes $g(\mathbf{X}) := f_1(B \cdot \mathbf{X})$.

Step 2. Alice gives the polynomial $g(\mathbf{X})$ to Bob.

Step 3. Bob asks Alice either to

(a) Prove that f_1 and g are equivalent; or,

(b) Prove that f_2 and g are equivalent.

Moreover, Bob chooses to ask (a) or (b) randomly with the same probability $\frac{1}{2}$.

Step 4. Alice complies. If Bob asks (a) then Alice reveals B. If Bob asks (b) then Alice reveals $A^{-1} \cdot B$.

It is easy to prove that this protocol is perfect zero-knowledge and that if somebody doesnt know/cannot compute an equivalence map from f_1 to f_2 then probability of success is at most $\frac{1}{2}$. In the key-generation phase, one generates the public polynomials f and g and the secret map A as follows.

Key Generation. Choose a random n-variate polynomial h of constant degree d (say $d = 4$). Choose matrices B and C uniformly at random from $\mathbb{F}^{(n\times n)*}$. Compute

$$\begin{aligned} f_1(\mathbf{X}) &:= h(B \cdot \mathbf{X}) \\ f_2(\mathbf{X}) &:= h(C \cdot \mathbf{X}) \\ A &:= B^{-1} \cdot C \end{aligned}$$

The public data f_1 and f_2 are then broadcast and the secret A is kept by Alice.

We now show that if in the key generation pase as described above, h is chosen to be a *random multilinear polynomial* then with high probability (over the choise of h) one can recover the secret A from the public data f_1 and f_2 in randomized polynomial-time.

8.2 Cryptanalysis. We now give a randomized polynomial-time algorithm to break this scheme. The algorithm is in two steps.

1. **Multilinearization.** We find invertible matrices B_1 and B_2 such that $g_1(\mathbf{X}) := f_1(B_1 \cdot \mathbf{X})$ and $g_2(\mathbf{X}) := f_2(B_2 \cdot \mathbf{X})$ are both multilinear polynomials.

2. **Normalization.** We find diagonal matrices C_1 and C_2 such that $h_1(\mathbf{X}) := g_1(C_1 \cdot \mathbf{X})$ and $h_2(\mathbf{X}) = g_2(C_2 \cdot \mathbf{X})$ are equivalent via a permutation matrix P.

3. **Permutation.** We find a permutation matrix P such that $h_1(P \cdot \mathbf{X}) = h_2(\mathbf{X})$. The secret S equals $C_2^{-1} \cdot B_2^{-1} \cdot B_1 \cdot C_1 \cdot P$.

In carrying out these two steps we will be crucially using the fact that $f(\mathbf{X})$ was chosen randomly. It implies the following facts:

FACT 8.1. *1. With high probability, the secret S is unique, i.e. there is a unique matrix S such that*

$$f_2(\mathbf{X}) = f_1(S \cdot \mathbf{X}).$$

2. *With high probability, the polynomials g_1 and g_2 are unique* upto permutations of variables and scalar multiplications of variables. *i.e. if $h_1(\mathbf{X})$ is another multilinear polynomial which is equivalent to $g_1(\mathbf{X})$ then h_1 can be obtained from g_1 by sending every variable to a scalar multiple of some other variable.*

3. *With high probability, the matrix P is unique.*

4. *With high probability, the set of second order partial derivatives of f has dimension $\binom{n}{2}$ (and therefore so do f_1 and f_2).*

Step One: Multilinearization Step One of the algorithm is accomplished using the following theorem from section 7.

THEOREM 8.1. *Let $f_1(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ be a polynomial whose second order partial derivatives has dimension $\binom{n}{2}$. Then given f_1, we can efficiently compute a matrix B such that $f_1(B_1 \cdot \mathbf{X})$ is multilinear.*

Using this, our problem boils down to finding an equivalence between two given multilinear polynomials, where the equivalence itself sends every variable to a scalar multiple of some other variable.

Step Two: Normalization Let $F(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ be a multilinear polynomial. We say that $F(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ is *normalized* if the product of its nonzero coefficients is 1. We say that F is normalized with respect to the variable x_1 if $F_0(x_2, \ldots, x_n)$ is normalized, where

$$F(\mathbf{X}) = F_1(x_2, x_3, \ldots, x_n) \cdot x_1 + F_0(x_2, x_3, \ldots, x_n).$$

The idea of the second step is that we find $\lambda_1, \lambda_2, \ldots, \lambda_n$ such that $g_1(\lambda_1 x_1, \ldots, \lambda_n x_n)$ is normalized with respect to each variable x_i. We do the same thing for g_2. Finding these λ_i's amounts to solving a system of equations of the form

$$\begin{aligned} \lambda_1^{e_{11}} \cdot \lambda_2^{e_{12}} \cdot \ldots \cdot \lambda_n^{e_{1n}} &= \alpha_1 \\ \lambda_1^{e_{21}} \cdot \lambda_2^{e_{22}} \cdot \ldots \cdot \lambda_n^{e_{2n}} &= \alpha_2 \\ \vdots & \vdots \\ \lambda_1^{e_{n1}} \cdot \lambda_2^{e_{n2}} \cdot \ldots \cdot \lambda_n^{e_{nn}} &= \alpha_n \end{aligned}$$

By taking logarithms, finding the solution to equations of the form above basically amounts to solving a set of linear equations and thus this step can be done efficiently. The key thing is:

FACT 8.2. *If the multilinear polynomials $g_1(\mathbf{X})$ and $g_2(\mathbf{X})$ are equivalent via a transformation that sends every variable to a scalar multiple of another variable then their normalized counterparts h_1 and h_2 are equivalent via a permutation of the variables.*

Step Three: Permutation We are given $h_1, h_2 \in \mathbb{F}[\mathbf{X}]$ and we want to find a permutation matrix P such that $h_1(P \cdot \mathbf{X}) = h_2(\mathbf{X})$. We are given $h_1, h_2 \in \mathbb{F}[\mathbf{X}]$ and we want to find a permutation matrix P such that $h_1(P \cdot \mathbf{X}) = h_2(\mathbf{X})$. We associate the polynomial $h_i(\mathbf{X})$ with a weighted hypergraph H_i ($i \in [2]$) in the natural way. There are n nodes in each H_i, with each of the nodes correponding to a variable x_j. The hyperedges correspond to subsets of size $d = 4$ of the variables. The weight of the hyperedge corresponding to the set of variables $\{x_{j_1}, x_{j_2}, x_{j_3}, x_{j_4}\}$ is the coefficient of the corresponding monomial $x_{j_1} \cdot x_{j_2} \cdot x_{j_3} \cdot x_{j_4}$. In this way our problem boils down to finding an isomorphism between two hypergraphs. Hypergraph isomorphism is a special case of graph isomorphism and therefore it can be solved efficiently *on the average*. To be more concrete, the required hypergraph isomorphism algorithm can be obtained in the f following manner. Miller [Mil79] gives a reduction from hypergraph isomophism to graph isomorphism. This reduction combined with the appropriate average case algorithms for graph isomorphism, such as for example the algorithm of Babai [Bab86], gives an average case algorithm that suits our purpose. This completes our description of the cryptanalysis of the aforesaid variant of Patarin's authentication scheme.

9 Discussion and open problems

A randomized polynomial-time algorithm for polynomial equivalence remains an open problem, even in the average-case and even when the polynomials are given verbosely as a list of coefficients rather than as arithmetic circuits. As intermediate steps towards this goal and as interesting mathematical tasks in themselves, we would like to pose the following two problems which are special cases of polynomial equivalence testing: devise an efficient algorithm (if such an algorithm exists) to test if a given polynomial is equivalent to

1. The Determinant
2. The Permanent.

References

[AS06] M. Agrawal and N. Saxena. Equivalence of F-algebras and cubic forms. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 115–126, 2006.

[Bab86] László Babai. A las vegas-nc algorithm for isomorphism of graphs with bounded multiplicity of eigenvalues. In *FOCS*, pages 303–312, 1986.

[BS83] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330, 1983.

[Car06] E. Carlini. *Reducing the number of variables of a polynomial,* Algebraic geometry and geometric modelling, pages 237–247. Mathematics and Visualization. Springer, 2006.

[FP06] Jean-Charles Faugere and Ludovic Perret. Polynomial equivalence problems: Theoretical and practical aspects. In *Proceedings of the 25th EUROCRYPT*, pages 30–47, 2006.

[Har75] D. K. Harrison. A grothendieck of higher degree forms. *Journal of Algebra*, 35:123–128, 1975.

[Kal89] E. Kaltofen. Factorization of polynomials given by straight-line programs. *Randomness and Computation*, 5:375–412, 1989.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press Cambridge, 1997.

[MH74] Y. I. Manin and M. Hazewinkel. *Cubic forms: algebra, geometry, arithmetic.* North-Holland Publishing Co., Amsterdam, 1974.

[Mil79] Gary L. Miller. Graph isomorphism, general remarks. *J. Comput. Syst. Sci.*, 18(2):128–142, 1979.

[Pat96] J. Patarin. Hidden field equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *Advances in Cryptology* — EUROCRYPT 1996, pages 33–48, 1996.

[PGC98] Jacques Patarin, Louis Goubin, and Nicolas Courtois. Improved algorithms for isomorphisms of polynomials. In *EUROCRYPT*, pages 184–200, 1998.

[Sax06] N. Saxena. *Automorphisms of rings and applications to complexity.* PhD thesis, Indian Institute of Technology Kanpur, 2006.

A Examples 1 and 2.

Recall the following statements from the introduction.

Example: Formula size of a quadratic polynomial. Let Φ be an arithmetic formula. The size of the formula Φ, denoted $L(\Phi)$ is defined to be the number of multiplication gates in it. For a polynomial f, $L(f)$ is the size of the smallest formula computing f. Then for a homogeneous quadratic polynomial f, we have that

$$L(f) = \left\lceil \frac{r}{2} \right\rceil,$$

where r is as given by Lemma 1.1.

Sketch of Proof : Let $f(\mathbf{X})$ be equivalent to $x_1^2 + \ldots + x_r^2$. Using the identity $y^2 + z^2 = (y + \sqrt{-1}z) \cdot (y - \sqrt{-1}z)$, we can replace the *sum of squares* representation above with a *sum of products of pairs.* That is,

$$f(\mathbf{X}) \sim \begin{cases} x_1x_2 + x_3x_4 + \ldots + x_{r-1}x_r & \text{if } r \text{ is even,} \\ x_1x_2 + x_3x_4 + \ldots + x_{r-2}x_{r-1} + x_r^2 & \text{if } r \text{ is odd,} \end{cases}$$

Let $g(\mathbf{X}) \stackrel{\text{def}}{=} x_1x_2 + x_3x_4 + \ldots + x_{r-1}x_r$. For any homogeneous quadratic polynomial, there is a homogeneous $\Sigma\Pi\Sigma$ (sum of product of sums) formula of minimal formula size for computing that polynomial. Using this the formula size for $g(\mathbf{X})$ can be deduced to be $\frac{r}{2}$ and furthermore that $L(f)$ is exactly $\left\lceil \frac{r}{2} \right\rceil$. □

Example: Graph Isomorphism many-one reduces to testing equivalence of cubic polynomials. [4]

Sketch of Proof : Let the two input graphs be $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Let $|V_1| = |V_2| = n$. Define the cubic polynomial f_{G_1} as follows:

$$f_{G_1}(\mathbf{X}) \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i^3 + \sum_{\{i,j\} \in E_1} x_i \cdot x_j.$$

Polynomial f_{G_2} is defined analogously. It suffices to prove that G_1 is isomorphic to G_2 if and only if f_{G_1} is equivalent to f_{G_2}. The forward direction is easy. For the other direction, assume that f_{G_1} is equivalent to f_{G_2}

[4] Agrawal and Saxena [AS06] showed the stronger result that graph isomorphism reduces to testing equivalence of *homogeneous* cubic polynomials, also known as cubic forms .

via the matrix A. i.e.

$$f_{G_1}(A \cdot \mathbf{X}) = f_{G_2}(\mathbf{X}).$$

Then the homogeneous cubic part of f_{G_1} must be equivalent, via A, to the homogeneous cubic part of f_{G_2} and the same thing holds for the homogeneous quadratic part. Corollary 5.1 describes the automorphisms of the polynomial $x_1^3 + \ldots + x_n^3$ and it says that there exists a permutation $\pi \in S_n$ and integers $i_1, i_2, \ldots, i_n \in \{0, 1, 2\}$ such that

$$A \cdot \mathbf{X} = (\omega_1^i \cdot x_{\pi(1)}, \omega_2^i \cdot x_{\pi(2)}, \ldots, \omega_n^i \cdot x_{\pi(n)}).$$

Using the equivalence via A of the homogeneous quadratic parts of f_{G_1} and f_{G_2}, we obtain that π in fact describes an isomorphism from G_1 to G_2.

□

A.1 A randomized algorithm for POLYDEP. Recall the following lemma:

Lemma: Given a vector of m polynomials $\mathbf{f} = (f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_m(\mathbf{X}))$ in which every f_i is as usual specified by a circuit, we can compute a basis for the space $\mathbf{f}^\perp$ in randomized polynomial time.

Proof. We will prove this by showing that $\mathbf{f}^\perp$ is actually the nullspace of a small, efficiently computable matrix. Suppose that $\mathbf{f}^\perp$ is spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_t$. Pick m points $\mathbf{a}_1, \ldots, \mathbf{a}_m$ in $\mathbb{F}^n$ and consider the $m \times m$ matrix M defined as follows:

$$M \stackrel{\text{def}}{=} \begin{pmatrix} f_1(\mathbf{a}_1) & f_2(\mathbf{a}_1) & \ldots & f_m(\mathbf{a}_1) \\ f_1(\mathbf{a}_2) & f_2(\mathbf{a}_2) & \ldots & f_m(\mathbf{a}_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\mathbf{a}_m) & f_2(\mathbf{a}_m) & \ldots & f_m(\mathbf{a}_m) \end{pmatrix} \tag{A.1}$$

Notice that for each $\mathbf{b}_i$ in the basis of $\mathbf{f}^\perp$, we always have $M \cdot \mathbf{b}_i = 0$ by definition. We now claim that with high probability over a random choice of the points $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{F}^n$, the matrix M has rank $(m - t)$. If this happens, then it means that the nullspace of M is exactly the space spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_t$, thereby enabling us to compute a basis of $\mathbf{f}^\perp$ efficiently. Towards this end, it is sufficient to prove the following claim:

Claim 7. *Let $P(\mathbf{X}_1, \ldots, \mathbf{X}_m)$ be the $m \times m$ matrix with entries in $\mathbb{F}(\mathbf{X}_1, \ldots, \mathbf{X}_m)$ defined as follows:*

$$P(\mathbf{X}_1, \ldots, \mathbf{X}_m) \stackrel{\text{def}}{=} \begin{pmatrix} f_1(\mathbf{X}_1) & f_2(\mathbf{X}_1) & \ldots & f_m(\mathbf{X}_1) \\ f_1(\mathbf{X}_2) & f_2(\mathbf{X}_2) & \ldots & f_m(\mathbf{X}_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\mathbf{X}_m) & f_2(\mathbf{X}_m) & \ldots & f_m(\mathbf{X}_m) \end{pmatrix}. \tag{A.2}$$

Then P has rank $(m - t)$.

Proof of Claim 7: Without loss of generality we can assume that the polynomials $f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_{m-t}(\mathbf{X})$ are $\mathbb{F}$-linearly independent and the rest of the polynomials are $\mathbb{F}$-linear combinations of these first $(m-t)$ polynomials. It is then sufficient to prove that the submatrix

$$Q \stackrel{\text{def}}{=} \begin{pmatrix} f_1(\mathbf{X}_1) & f_2(\mathbf{X}_1) & \ldots & f_{m-t}(\mathbf{X}_1) \\ f_1(\mathbf{X}_2) & f_2(\mathbf{X}_2) & \ldots & f_{m-t}(\mathbf{X}_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\mathbf{X}_{m-t}) & f_2(\mathbf{X}_{m-t}) & \ldots & f_{m-t}(\mathbf{X}_{m-t}) \end{pmatrix}$$

has full rank, or equivalently, the determinant of Q is a nonzero polynomial. Now, expanding $\textsc{Det}(Q)$ along the first row we have

$$\textsc{Det}(Q) = \sum_{j=1}^{m-t} (-1)^{j+1} f_j(\mathbf{X}_1) \cdot Q_{1j},$$

where Q_{ij} is the determinant of the ij-th minor.
Notice that every Q_{1k}, $k \in [m-t]$, is a polynomial in the set of variables $\mathbf{X}_2, \ldots, \mathbf{X}_{m-t}$. By induction, every Q_{1k} is a nonzero polynomial (since every subset of a set of $\mathbb{F}$-linearly independent polynomials is also $\mathbb{F}$-linearly independent). If $\textsc{Det}(Q)$ was the zero polynomial then plugging in random values for $\mathbf{X}_2, \ldots, \mathbf{X}_{m-t}$ would give us a nonzero $\mathbb{F}$-linear dependence among $f_1(\mathbf{X}_1), f_2(\mathbf{X}_1), \ldots, f_{m-t}(\mathbf{X}_1)$, which is a contradiction. Hence $\textsc{Det}(Q)$ must be nonzero, proving the claim. □

This also completes the proof of Lemma 4.1.

B Minimizing variables

This section is devoted to a proof of theorem 4.1. But first we give an example of a polynomial with redundant variables.

Example 8. *The number of essential variables in the quadratic polynomial $f(x_1, x_2, x_3) = x_1^2 + 2x_1x_2 + x_2^2 + x_3^2$ is just two because notice that $f = (x_1 + x_2)^2 + x_3^2$ and thus after making the invertible linear transformation*

$$A \quad : \quad \begin{array}{l} x_1 + x_2 \mapsto x_1 \\ x_3 \mapsto x_2 \\ x_2 \mapsto x_3 \end{array}$$

we get that $f(A \cdot \mathbf{X}) = x_1^2 + x_2^2$ is just a function of two variables x_1 and x_2.

The vanishing of partials. We now reprove a lemma due to Carlini [Car06]. Let us examine the situation when a variable is redundant. Let $g(\mathbf{X}) = f(A \cdot \mathbf{X})$ where A is an $n \times n$ invertible matrix. If g does not depend upon x_i then

$$\frac{\partial g}{\partial x_i} = 0 \quad \Leftrightarrow \quad \sum_{k=1}^{n} a_{ki} \cdot \frac{\partial f}{\partial x_k}(A \cdot \mathbf{X}) = 0$$

Thus there exists a vector $\mathbf{a} \in \mathbb{F}^n$ such that $\mathbf{a} \cdot \partial(f) = 0$, where $\partial(f) \stackrel{\text{def}}{=} (\partial_1 f, \partial_2 f, \ldots, \partial_n f)$. Using the notation of section 4.1, this can be written succintly as

$$\mathbf{a} \in \partial(f)^{\perp}$$

Suppose that $\mathbf{b}_1, \ldots, \mathbf{b}_t \in \mathbb{F}^n$ is a basis of the space $\partial(f)^{\perp}$. Now there exists $n-t$ independent vectors $\mathbf{a}_1, \ldots, \mathbf{a}_{n-t}$ such that the vector space $\mathbb{F}^n$ is spanned by $\mathbf{a}_1, \ldots, \mathbf{a}_{n-t}, \mathbf{b}_1, \ldots, \mathbf{b}_t$. Consider the invertible matrix whose columns are $\mathbf{a}_1, \ldots, \mathbf{a}_{n-t}, \mathbf{b}_1, \ldots, \mathbf{b}_t$ respectively. Let $g(\mathbf{X}) \stackrel{\text{def}}{=} f(A \cdot \mathbf{X})$. Then for $n - t + 1 \leq i \leq n$,

$$\begin{aligned} \frac{\partial g}{\partial x_i} &= \mathbf{b}_{i-n+t} \cdot \partial(f)(A \cdot \mathbf{X}) \\ &= 0 \ \ (\text{since } \mathbf{b}_{i-n+t} \cdot \partial(f)(\mathbf{X}) = 0) \end{aligned}$$

We thus have:

Lemma B.1. *(Carlini [Car06]) The number of redundant variables in a polynomial $f(\mathbf{X})$ equals the dimension of $\partial(f)^{\perp}$. Furthermore, given a basis of $\partial(f)^{\perp}$, we can easily come up with a linear transformation A on the variables such that the polynomial $f(A \cdot \mathbf{X})$ depends on only the first $(n - \dim(\partial(f)^{\perp}))$ variables.*

Notice that arithmetic circuits for each polynomial in $\partial(f)$ can be easily computed in poly($|f|$) time, where $|f|$ is the size of the circuit for f. This computation can be made even more efficient using the algorithm of Baur and Strassen [BS83]. Thereafter, a basis for the space $\partial(f)^{\perp}$ can be efficiently computed using Lemma 4.1. In this way we have:

Theorem Given a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ with m essential variables, we can compute in randomized polynomial time an invertible linear transformation $A \in \mathbb{F}^{(n\times n)*}$ such that $f(A \cdot \mathbf{X})$ depends on the first m variables only.

C Polynomial Decomposition

We are now set to generalize the sum of powers problem considered in section 5. Given a polynomial $f(\mathbf{X})$, we want to write it as the sum of two polynomials on disjoint sets of variables. That is, our aim is to find an invertible linear transformation A on the variables such that

$$f(A \cdot \mathbf{X}) = g(x_1, \ldots, x_t) + h(x_{t+1}, \ldots, x_n)$$

We first consider the special case of the above we just want to partition the set of variables $X = Y \uplus Z$ so that $f(\mathbf{X}) = g(\mathbf{y}) + h(\mathbf{z})$.

LEMMA C.1. *Given a low-degree polynomial $f(\mathbf{X})$, we can efficiently compute a partition $X = Y \uplus Z$ of the variables such that $f(\mathbf{X}) = g(\mathbf{y}) + h(\mathbf{z})$, if such a partition exists.*

Proof. Observe that given $f(\mathbf{X})$ and two variables x_i and x_j we can efficiently determine whether there is any monomial in f which contains both these variables by plugging in randomly chosen value for the remaining variables and determining whether the resulting bivariate polynomial has any such monomial or not. Now create an undirected graph G_f whose nodes are the variables and there is an edge between the nodes x_i and x_j if and only if there is a monomial in $f(\mathbf{X})$ which contains both x_i and x_j. We find the connected components of G_f. The partitioning of the set of variables induced by the connected components of G_f gives the required partition of variables needed for decomposition.

Our main interest though is in devising an algorithm for polynomial decomposition that allows arbitary invertible linear transformations of the variables. Now let $f(\mathbf{X})$ be a regular polynomial. Suppose that for some invertible linear transformation $A \in \mathbb{F}^{(n\times n)*}$:

$$f(A \cdot \mathbf{X}) = g(x_1, \ldots, x_t) + h(x_{t+1}, \ldots, x_n)$$

Without loss of generality, we can assume that $\textsc{Det}(A) = 1$. Let $F(\mathbf{X}) = f(A \cdot \mathbf{X})$. Then observe that

$$\textsc{Det}(H_F)(\mathbf{X}) = \textsc{Det}(H_g)(\mathbf{X}) \cdot \textsc{Det}(H_h)(\mathbf{X})$$

Now by Lemma 5.1 we have

$$\textsc{Det}(H_f)(A \cdot \mathbf{X}) = \textsc{Det}(H_g)(A \cdot \mathbf{X}) \cdot \textsc{Det}(H_h)(A \cdot \mathbf{X}).$$

Also observe that $\textsc{Det}(H_g)(\mathbf{X})$ is in fact a polynomial in the variables $x_1, \ldots, x_t$ whereas $\textsc{Det}(H_h)(\mathbf{X})$ is a polynomial in the remaining $(n-t)$ variables $x_{t+1}, \ldots, x_n$. This motivates us to look at a multiplicative version of the polynomial decomposition problem. Let $D(\mathbf{X})$ be the polynomial $\textsc{Det}(H_f)(\mathbf{X})$. Then we want to make an invertible linear transformation on the variables and write D as the product of polynomials on disjoint sets of variables.

A multiplicative version of polynomial decomposition We are given a polynomial $D(\mathbf{X})$ and we want to make a linear transformation B on the variables to get a factorization of the form

$$D(B \cdot \mathbf{X}) = \prod_{i=1}^{k} C_i(x_1, \ldots, x_{t_1}),$$

where the individual C_i's are 'multiplicatively indecomposable'.
Towards this end, let us make a definition. For a polynomial $f(\mathbf{X})$, we denote by $f^{\perp\perp}$ the vector space orthogonal to $\partial(f)^{\perp}$. That is,

$$f^{\perp\perp} \stackrel{\text{def}}{=} \{\mathbf{a} \in \mathbb{F}^n | \mathbf{a} \cdot \mathbf{v} = 0 \ \ \forall \mathbf{v} \in \partial(f)^{\perp}\}$$

Intuitively, a basis for $f^{\perp\perp}$ corresponds to the essential variables of $f(\mathbf{X})$. Let us note down some basic properties of $f^{\perp\perp}$.

PROPERTY C.1. *Let* $f(\mathbf{X}) = g(A \cdot \mathbf{X})$. *Then*

$$g^{\perp\perp} = \{(A^T)^{-1} \cdot \mathbf{v} \quad : \quad \mathbf{v} \in f^{\perp\perp}\}$$

Notice that any factor $C(\mathbf{X})$ of the multivariate polynomial $D(\mathbf{X})$ depends on a subset of the variables which $D(\mathbf{X})$ itself depends upon. Furthermore $D(\mathbf{X})$ does depend on all the variables in any divisor $C(\mathbf{X})$.

LEMMA C.2. *If a polynomial* $D(\mathbf{X})$ *has the factorization*

$$D(\mathbf{X}) = C_1(\mathbf{X})^{e_1} \cdot C_2(\mathbf{X})^{e_2} \cdot \ldots \cdot C_k(\mathbf{X})^{e_k},$$

then the space $D^{\perp\perp}$ *is the linear span of the spaces* ${C_1}^{\perp\perp}, {C_2}^{\perp\perp}, \ldots, {C_k}^{\perp\perp}$.

Lemma C.2 together with Kaltofen's algorithm for factoring low-degree polynomials allows us to devise an efficient algorithm for a multiplicative version of polynomial decomposition.

THEOREM C.1. *There exists an efficient randomized algorithm that given a regular low-degree polynomial* $D(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, *computes an invertible linear transformation* $A \in \mathbb{F}^{(n\times n)*}$ *such that*

$$D(A \cdot \mathbf{X}) = \prod_{i=1}^{k} C_i(x_1, \ldots, x_{t_1}),$$

where the individual C_i*'s are multiplicatively indecomposable, if such a transformation* A *exists.*

Polynomial Decomposition Algorithm We now give the algorithm for the usual notion of decomposition of polynomials. **Input.** A regular low-degree n-variate polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$.

Output. An invertible linear transformation A such that $f(A \cdot \mathbf{X})$ is the sum of two polynomials on disjoint sets of variables.

The Algorithm.

1. Compute an arithmetic circuit $D(\mathbf{X})$ which computes $\text{DET}(H_f(\mathbf{X}))$.
2. Use the multiplicative polynomial decomposition algorithm of theorem C.1 to determine a linear transformation $A \in \mathbb{F}^{(n\times n)*}$ such that
$$D(A \cdot \mathbf{X}) = \prod_{i=1}^{k} C_1(x_1, \ldots, x_{t_1}),$$
where the individual C_i's are multiplicatively indecomposable. If no such A exists then output no decomposition exists.
3. Use the algorithm of Lemma C.1 check if $f(A \cdot \mathbf{X})$ can be written as the sum of two polynomials on disjoint sets of variables. If so output A else output no such decomposition exists.

The following theorem summarizes the conditions under which the above algorithm is guaranteed to give the right answer.

THEOREM C.2. *Given a* n*-variate polynomial* $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, *the algorithm above finds a decomposition of* $f(\mathbf{X})$, *if it exists, in randomized polynomial time provided* $\text{DET}(H_f(\mathbf{X}))$ *is a regular polynomial, i.e. it has* n *variables upto equivalence.*