# Average-Case Algorithms for Testing Isomorphism of Polynomials, Algebras, and Multilinear Forms

**Joshua A. Grochow**
Departments of Computer Science and Mathematics, University of Colorado Boulder, Boulder, CO, USA

**Youming Qiao**
Center for Quantum Software and Information, University of Technology Sydney, Ultimo, NSW 2007, Australia

**Gang Tang**
Center for Quantum Software and Information, University of Technology Sydney, Ultimo, NSW 2007, Australia

## Abstract

We study the problems of testing isomorphism of polynomials, algebras, and multilinear forms. Our first main results are average-case algorithms for these problems. For example, we develop an algorithm that takes two cubic forms $f, g \in \mathbb{F}_q[x_1, \ldots, x_n]$, and decides whether $f$ and $g$ are isomorphic in time $q^{O(n)}$ *for most* $f$. This average-case setting has direct practical implications, having been studied in multivariate cryptography since the 1990s. Our second result concerns the complexity of testing equivalence of alternating trilinear forms. This problem is of interest in both mathematics and cryptography. We show that this problem is polynomial-time equivalent to testing equivalence of symmetric trilinear forms, by showing that they are both Tensor Isomorphism-complete (Grochow & Qiao, *ITCS* 2021), therefore is equivalent to testing isomorphism of cubic forms over most fields.

**2012 ACM Subject Classification** Computing methodologies → Algebraic algorithms; Computing methodologies → Combinatorial algorithms

**Keywords and phrases** polynomial isomorphism, trilinear form equivalence, algebra isomorphism, average-case algorithms, tensor isomorphism complete, symmetric and alternating bilinear maps

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2021.38

**Related Version** *Full Version*: `https://arxiv.org/abs/2012.01085`

**Funding** *Joshua A. Grochow*: Partially supported by NSF grants DMS-1750319 and DMS-1622390.
*Youming Qiao*: Partially supported by the Australian Research Council DP200100950.
*Gang Tang*: Partially supported by the Australian Research Council DP160101652.

**Acknowledgements** We thank the anonymous reviewers for their careful reading and helpful suggestions.

## 1 Introduction

In this paper, we study isomorphism testing problems for polynomials, algebras, and multilinear forms. Our first set of results is algorithmic, namely average-case algorithms for these problems (Section 1.1). Our second result is complexity-theoretic, concerning the problems of testing equivalence of symmetric and alternating trilinear forms (Section 1.2).

### 1.1 Average-case algorithms for polynomial isomorphism and more

**The polynomial isomorphism problem.** Let $\mathbb{F}$ be a field, and let $X = \{x_1, \ldots, x_n\}$ be a set of variables. Let $\mathrm{GL}(n, \mathbb{F})$ be the general linear group consisting of $n \times n$ invertible matrices over $\mathbb{F}$. A natural group action of $A = (a_{i,j}) \in \mathrm{GL}(n, \mathbb{F})$ on the polynomial ring $\mathbb{F}[X]$ sends $f(x_1, \ldots, x_n)$ to $f \circ A := f(\sum_{j=1}^n a_{1,j}x_j, \ldots, \sum_{j=1}^n a_{n,j}x_j)$. The *polynomial isomorphism*

*problem* (PI) asks, given $f, g \in \mathbb{F}[X]$, whether there exists $A \in \mathrm{GL}(n, \mathbb{F})$ such that $f = g \circ A$. In the literature, this problem was also called the polynomial equivalence problem [1].

An important subcase of PI is when the input polynomials are required to be homogeneous of degree $d$. In this case, this problem is referred to as the homogeneous polynomial isomorphism problem, denoted as $d$-HPI. Homogeneous degree-3 (resp. degree-2) polynomials are also known as cubic (resp. quadratic) forms.

In this article, we assume that a polynomial is represented in algorithms by its list of coefficients of the monomials, though other representations like algebraic circuits are also possible in this context [13]. Furthermore, we shall mostly restrict our attention to the case when the polynomial degrees are constant.

**Motivations to study polynomial isomorphism.** The polynomial isomorphism problem has been studied in both multivariate cryptography and computational complexity. In 1996, inspired by the celebrated zero-knowledge protocol for graph isomorphism [9], Patarin proposed to use PI as the security basis of authentication and signature protocols [17]. This lead to a series of works on practical algorithms for PI; see [3,4,12] and references therein. In the early 2000s, Agrawal, Kayal and Saxena studied PI from the computational complexity perspective. They related PI with graph isomorphism and algebra isomorphism [1,2], and studied some special instances of PI [13] as well as several related algorithmic tasks [18].

Despite these works, little progress has been made on algorithms *with rigorous analysis* for the *general* PI. More specifically, Kayal's algorithm [13] runs in randomized polynomial time, works for the degree $d \geq 4$, and doesn't require the field to be finite. However, it only works in the *multilinear* setting, namely when $f$ and $g$ are isomorphic to a common multilinear polynomial $h$. The algorithms from multivariate cryptography [4] either are heuristic, or need unproven assumptions. While these works contain several nice ideas and insights, and their implementations show practical improvements, they are nonetheless heuristic in nature, and rigorously analyzing them seems difficult. Indeed, if any of these algorithms had worst-case analysis matching their heuristic performance, it would lead to significant progress on the long-open Group Isomorphism problem (see, e.g., [11,15]).

**Our result on polynomial isomorphism.** Our first result is an average-case algorithm with rigorous analysis for PI over a finite field $\mathbb{F}_q$. As far as we know, this is the first non-trivial algorithm with rigorous analysis for PI over finite fields. (The natural brute-force algorithm, namely enumerating all invertible matrices, runs in time $q^{n^2} \cdot \mathrm{poly}(n, \log q)$.) Furthermore, the average-case setting is quite natural, as it is precisely the one studied multivariate cryptography. We elaborate on this further after stating our result.

To state the result, let us define what a random polynomial means in this setting. Since we represent polynomials by their lists of coefficients, a random polynomial of degree $d$ is naturally the one whose coefficients of the monomials of degree $\leq d$ are independently randomly drawn from $\mathbb{F}_q$. We also consider the homogeneous setting where only monomials of degree $= d$ are of interest.

▶ **Theorem 1.** *Let $d \geq 3$ be a constant. Let $f, g \in \mathbb{F}_q[x_1, \ldots, x_n]$ be (resp. homogeneous) polynomials of degree $\leq d$ (resp. $= d$). There exists an $q^{O(n)}$-time algorithm that decides whether $f$ and $g$ are isomorphic, for all but at most $\frac{1}{q^{\Omega(n)}}$ fraction of $f$.*

*Furthermore, if $f$ and $g$ are isomorphic, then this algorithm also computes an invertible matrix $A$ which sends $f$ to $g$.*

Let us briefly indicate the use of this average-case setting in multivariate cryptography. In the authentication scheme described in [17], the public key consists of two polynomials $f, g \in \mathbb{F}_q[x_1, \ldots, x_n]$, where $f$ is a random polynomial, and $g$ is obtained by applying a random invertible matrix to $f$. Then $f$ and $g$ are public keys, and any isomorphism from $f$ to $g$ can serve as the private key. Therefore, the algorithm in Theorem 1 can be used to recover a private key for most $f$.

**Adapting the algorithm strategy to more isomorphism problems.** In [1, 2], the algebra isomorphism problem (AI) was studied and shown to be polynomial-time equivalent to PI over most fields. In [11], many more problems are demonstrated to be polynomial-time equivalent to PI, including the trilinear form equivalence problem (TFE). In these reductions, due to the blow-up of the parameters, the $q^{O(n)}$-time algorithm in Theorem 1 does not translate to moderately exponential-time, average-case algorithms for these problems. The algorithm design idea, however, does translate to give moderately exponential-time, average-case algorithms for AI and TFE. This will be shown in Section 3.2.

## 1.2 Complexity of symmetric and alternating trilinear form equivalence

**From cubic forms to symmetric and alternating trilinear forms.** In the context of polynomial isomorphism, cubic forms are of particular interest. In complexity theory, it was shown that $d$-HPI reduces to cubic form isomorphism over fields with $d$th roots of unity [1,2], or over fields of characteristic 0 or $> d$ [11]. In multivariate cryptography, cubic form isomorphism also received special attention, since using higher degree forms results in less efficiency in the cryptographic protocols.

Just as quadratic forms are closely related with symmetric bilinear forms, cubic forms are closely related with symmetric trilinear forms. Let $\mathbb{F}$ be a field of characteristic not 2 or 3, and let $f = \sum_{1 \le i \le j \le k \le n} a_{i,j,k} x_i x_j x_k \in \mathbb{F}[x_1, \ldots, x_n]$ be a cubic form. For any $i, j, k \in [n]$, let $1 \le i' \le j' \le k' \le n$ be the result of sorting $i, j, k$ in the increasing order, and set $a_{i,j,k} = a_{i',j',k'}$. Then we can define a symmetric[1] trilinear form $\phi_f : \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ by

$$\phi_f(u, v, w) = \sum_{i \in [n]} a_{i,i,i} u_i v_i w_i + \frac{1}{3} \cdot \sum_{\substack{i,j,k \in [n] \\ |\{i,j,k\}|=2}} a_{i,j,k} u_i v_j w_k + \frac{1}{6} \cdot \sum_{\substack{i,j,k \in [n] \\ i,j,k \text{ all different}}} a_{i,j,k} u_i v_j w_k.$$

It can be seen easily that for any $v = (v_1, \ldots, v_n)^{\mathrm{t}} \in \mathbb{F}^n$, $f(v_1, \ldots, v_n) = \phi_f(v, v, v)$.

In the theory of bilinear forms, symmetric and skew-symmetric bilinear forms are two important special subclasses. For example, they are critical in the classifications of classical groups [19] and finite simple groups [21]. For trilinear forms, we also have skew-symmetric trilinear forms. In fact, to avoid some complications over fields of characteristics 2 or 3, we shall consider alternating trilinear forms which are closely related to skew-symmetric ones. For trilinear forms, the exceptional groups of type $E_6$ can be constructed as the stabilizer of certain symmetric trilinear forms, and those of type $G_2$ can be constructed as the stabilizer of certain alternating trilinear forms.

We say that a trilinear form $\phi : \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ is *alternating*, if whenever two arguments of $\phi$ are equal, $\phi$ evaluates to zero. Note that this implies skew-symmetry, namely for any $u_1, u_2, u_3 \in \mathbb{F}^n$ and any $\sigma \in \mathrm{S}_3$, $\phi(u_1, u_2, u_3) = \mathrm{sgn}(\sigma) \cdot \phi(u_{\sigma(1)}, u_{\sigma(2)}, u_{\sigma(3)})$. Over fields of characteristic zero or $> 3$, this is equivalent to skew-symmetry.

[1] That is, for any permutation $\sigma \in \mathrm{S}_3$, $\phi(u_1, u_2, u_3) = \phi(u_{\sigma(1)}, u_{\sigma(2)}, u_{\sigma(3)})$.

**The trilinear form equivalence problem.** Given a trilinear form $\phi : \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$, $A \in \mathrm{GL}(n, \mathbb{F})$ naturally acts on $\phi$ by sending it to $\phi \circ A := \phi(A^{-1}(u), A^{-1}(v), A^{-1}(w))$. The *trilinear form equivalence problem* then asks, given two trilinear forms $\phi, \psi : \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$, whether there exists $A \in \mathrm{GL}(n, \mathbb{F})$, such that $\phi = \psi \circ A$. Over fields of characteristic not 2 or 3, two cubic forms $f$ and $g$ are isomorphic if and only if $\phi_f$ and $\phi_g$ are equivalent, so cubic form isomorphism is polynomial-time equivalent to symmetric trilinear form equivalence over such fields. Note that for clarity, we reserve the term "isomorphism" for polynomials (and cubic forms), and use "equivalence" for multilinear forms.

**Motivations to study alternating trilinear form equivalence.** Our main interest is to study the complexity of alternating trilinear form equivalence, with the following motivations.

The first motivation comes from cryptography. To store a symmetric trilinear form on $\mathbb{F}_q^n$, $\binom{n+2}{3}$ field elements are required. To store an alternating trilinear form on $\mathbb{F}_q^n$, $\binom{n}{3}$ field elements are needed. The difference between $\binom{n+2}{3}$ and $\binom{n}{3}$ could be significant for practical purposes. For example, when $n = 9$, $\binom{n+2}{3} = \binom{11}{3} = 165$, while $\binom{n}{3} = \binom{9}{3} = 84$. This means that in the authentication protocol of Patarin [17], using alternating trilinear forms instead of cubic forms for $n = 9$,[2] one saves almost one half in the public key size, which is an important saving in practice.

The second motivation originates from comparing symmetric and alternating bilinear forms. It is well-known that, *in the bilinear case*, the structure of alternating forms is simpler than that for symmetric ones [14]. Indeed, up to equivalence, an alternating bilinear form is completely determined by its rank over any field, while the classification of symmetric bilinear forms depends crucially on the underlying field. For example, recall that over $\mathbb{R}$, a symmetric form is determined by its "signature", so just the rank is not enough.

A third motivation is implied by the representation theory of the general linear groups; namely that alternating trilinear forms are the "last" natural case for $d = 3$. If we consider the action of $\mathrm{GL}(n, \mathbb{C})$ acting on $d$-tensors in $\mathbb{C}^n \otimes \mathbb{C}^n \otimes \cdots \otimes \mathbb{C}^n$ diagonally (that is, the same matrix acts on each tensor factor), it is a classical result [19] that the invariant subspaces of $(\mathbb{C}^n)^{\otimes d}$ under this action are completely determined by the irreducible representations of $\mathrm{GL}(n, \mathbb{C})$. When $d = 3$, there are only three such representations, which correspond precisely to: symmetric trilinear forms, Lie algebras, and alternating trilinear forms. From the complexity point of view, it was previously shown that isomorphism of symmetric trilinear forms [1, 2] and Lie algebras [11] are equivalent to algebra isomorphism. Here we show that the last case, isomorphism of alternating trilinear forms, is also equivalent to the others.

**The complexity of alternating trilinear form equivalence.** Given the above discussion on the comparison between symmetric and alternating bilinear forms, one may wonder whether alternating trilinear form equivalence was easier than symmetric trilinear form equivalence. Interestingly, we show that this is not the case; rather, they are polynomial-time equivalent.

▶ **Theorem 2.** *The alternating trilinear form equivalence problem is polynomial-time equivalent to the symmetric trilinear form equivalence problem.*

[2] The parameters of the cryptosystem are $q$ and $n$. When $q = 2$, $n = 9$ is not secure as it can be solved in practice [5]. So $q$ needs to be large for $n = 9$ to be secure. Interestingly, according to [4, pp. 227], the parameters $q = 16$ and $n = 8$ seemed difficult for practical attacks via Gröbner basis.

Note here that the reduction from alternating to symmetric trilinear form equivalence requires us to go through the tensor isomorphism problem, which causes polynomial blow-ups in the dimensions of the underlying vector spaces. Therefore, though these two problems are polynomial-time equivalent, these problems may result in cryptosystems with different efficiencies for a given security level.

## 1.3 Previous works

**The relation between $PI$ and $AI$.** As mentioned in Section 1.1, the degree-$d$ homogeneous polynomial isomorphism problem ($d$-HPI) was shown to be almost equivalent to the algebra isomorphism problem (AI) in [1, 2]. (See Section 3.2 for the formal definition of algebra isomorphism problem.) Here, almost refers to that for the reduction from $d$-HPI to AI in [1, 2], the underlying fields are required to contain a $d$th root of unity. When $d = 3$, this means that the characteristic of the underlying field $p$ satisfies that $p = 2 \mod 3$ or $p = 0$, which amounts to half of the primes. In [11], another reduction from 3-HPI to AI was presented, which works for fields of characteristics not 2 or 3. The reduction from AI to 3-HPI in [2] works over any field.

**The tensor isomorphism complete class.** In [8, 11], polynomial-time equivalences are proved between isomorphism testing of many more mathematical structures, including tensors, matrix spaces, polynomial maps, and so on. These problems arise from many areas: besides multivariate cryptography and computational complexity, they appear in quantum information, machine learning, and computational group theory. This motivates the authors of [11] to define the tensor isomorphism complete class TI, which we recall here.

▶ **Definition 3** (The $d$-Tensor Isomorphism problem, and the complexity class TI)**.** *$d$-Tensor Isomorphism over a field $\mathbb{F}$ is the problem: given two $d$-way arrays $\mathtt{A} = (a_{i_1,\ldots,i_d})$ and $\mathtt{B} = (b_{i_1,\ldots,i_d})$, where $i_k \in [n_k]$ for $k \in [d]$, and $a_{i_1,\ldots,i_d}, b_{i_1,\ldots,i_d} \in \mathbb{F}$, decide whether there are $P_k \in \mathrm{GL}(n_k, \mathbb{F})$ for $k \in [d]$, such that for all $i_1, \ldots, i_d$,*

$$a_{i_1,\ldots,i_d} = \sum_{j_1,\ldots,j_d} b_{j_1,\ldots,j_d} (P_1)_{i_1,j_1} (P_2)_{i_2,j_2} \cdots (P_d)_{i_d,j_d}. \tag{1}$$

*For any field $\mathbb{F}$, $\mathrm{TI}_{\mathbb{F}}$ denotes the class of problems that are polynomial-time Turing (Cook) reducible to $d$-Tensor Isomorphism over $\mathbb{F}$, for some $d$. A problem is $\mathrm{TI}_{\mathbb{F}}$-complete, if it is in $\mathrm{TI}_{\mathbb{F}}$, and $d$-Tensor Isomorphism over $\mathbb{F}$ for any $d$ reduces to this problem.*

*When a problem is naturally defined and is $\mathrm{TI}_{\mathbb{F}}$-complete over any $\mathbb{F}$, then we can simply write that it is* TI*-complete.*

**Average-case algorithms for matrix space isometry.** In [6, 15], motivated by testing isomorphism of $p$-groups (widely believed to be the hardest cases of Group Isomorphism, see e.g. [10]), the algorithmic problem alternating matrix space isometry was studied. (In the literature [20], this problem was also known as the alternating bilinear map pseudo-isometry problem.) That problem asks the following: given two linear spaces of alternating matrices $\mathcal{A}, \mathcal{B} \leq \Lambda(n, q)$, decide whether there exists $T \in \mathrm{GL}(n, q)$, such that $\mathcal{A} = T^{\mathrm{t}}\mathcal{B}T = \{T^{\mathrm{t}}BT : B \in \mathcal{B}\}$. (See Section 2 for the definition of alternating matrices.) The main result of [6], improving upon the one in [15], is an average-case algorithm for this problem in time $q^{O(n+m)}$, where $m = \dim(\mathcal{A})$.

## 1.4 Remarks on the technical side

**Techniques for proving Theorem 1.** The algorithm for PI in Theorem 1 is based on the algorithmic idea from [6, 15]. However, to adapt that idea to the PI setting, there are several interesting conceptual and technical difficulties.

One conceptual difficulty is that for alternating matrix space isometry, there are actually two GL actions, one is by $\mathrm{GL}(n, q)$ as explicitly described above, and the other is by $\mathrm{GL}(m, q)$ performing the basis change of matrix spaces. The algorithm in [6] crucially uses that the $\mathrm{GL}(m, q)$ action is "independent" of the $\mathrm{GL}(n, q)$ action. For PI, there is only one $\mathrm{GL}(n, q)$-action acting on all the variables. Fortunately, as we show in Section 3.1, there is still a natural way of applying the the basic idea from [6, 15].

One technical difficulty is that the analysis in [6] relies on properties of random alternating matrices, while for 3-HPI, the analysis relies on properties of random symmetric matrices. To adapt the proof strategy in [6] (based on [15]) to the symmetric setting is not difficult, but suggests some interesting differences between symmetric and alternating matrices (see Appendix A in the full version of this paper).

**Techniques for proving Theorem 2.** By [8], the trilinear form equivalence problem is in TI, and so are the special cases symmetric and alternating trilinear form equivalence. The proof of Theorem 2 goes by showing that both symmetric and alternating trilinear form equivalence are TI-hard.

Technically, the basic proof strategy is to adapt a gadget construction, which originates from [8] and then is further used in [11]. To use that gadget in the trilinear form setting does require several non-trivial ideas. First, we identify the right TI-complete problem to start with, namely the alternating (resp. symmetric) matrix space isometry problem. Second, we need to arrange a 3-way array $\mathtt{A}$, representing a linear basis of an alternating (resp. symmetric) matrix spaces, into one representing an alternating trilinear form. This requires 3 copies of $\mathtt{A}$, assembled in an appropriate manner. Third, we need to add the gadget in three directions (instead of just two as in previous results). All these features were not present in [8, 11]. The correctness proof also requires certain tricky twists compared with those in [8] and [11].

## 2 Preliminaries

**Notations.** We collect the notations here, though some of them have appeared in Section 1. Let $\mathbb{F}$ be a field. Vectors in $\mathbb{F}^n$ are column vectors. Let $e_i$ denote the $i$th standard basis vector of $\mathbb{F}^n$. Let $\mathrm{M}(\ell \times n, \mathbb{F})$ be the linear space of $\ell \times n$ matrices over $\mathbb{F}$, and set $\mathrm{M}(n, \mathbb{F}) := \mathrm{M}(n \times n, \mathbb{F})$. Let $I_n$ denote the identity matrix of size $n$. For $A \in \mathrm{M}(n, \mathbb{F})$, $A$ is *symmetric* if $A^{\mathrm{t}} = A$, and *alternating* if for every $v \in \mathbb{F}^n$, $v^{\mathrm{t}} A v = 0$. When the characteristic of $\mathbb{F}$ is not 2, $A$ is alternating if and only if $A$ is skew-symmetric. Let $\mathrm{S}(n, \mathbb{F})$ be the linear space of $n \times n$ symmetric matrices over $\mathbb{F}$, and let $\Lambda(n, \mathbb{F})$ be the linear space of alternating matrices over $\mathbb{F}$. When $\mathbb{F} = \mathbb{F}_q$, we may write $\mathrm{M}(n, \mathbb{F}_q)$ as $\mathrm{M}(n, q)$. We use $\langle \cdot \rangle$ to denote the linear span.

**3-way arrays.** A 3-way array over a field $\mathbb{F}$ is an array with three indices whose elements are from $\mathbb{F}$. We use $\mathrm{M}(n_1 \times n_2 \times n_3, \mathbb{F})$ to denote the linear space of 3-way arrays of side lengths $n_1 \times n_2 \times n_3$ over $\mathbb{F}$.

Let $\mathtt{A} \in \mathrm{M}(\ell \times n \times m, \mathbb{F})$. For $k \in [m]$, the $k$th *frontal* slice of $\mathtt{A}$ is $(a_{i,j,k})_{i \in [\ell], j \in [n]} \in \mathrm{M}(\ell \times n, \mathbb{F})$. For $j \in [n]$, the $j$th *vertical* slice of $\mathtt{A}$ is $(a_{i,j,k})_{i \in [\ell], k \in [m]} \in \mathrm{M}(\ell \times m, \mathbb{F})$. For $i \in [\ell]$, the $i$th *horizontal* slice of $\mathtt{A}$ is $(a_{i,j,k})_{j \in [n], k \in [m]} \in \mathrm{M}(n \times m, \mathbb{F})$. We shall often think of $\mathtt{A}$ as a matrix tuple in $\mathrm{M}(\ell \times n, \mathbb{F})^m$ consisting of its frontal slices.

A natural action of $(P, Q, R) \in \mathrm{GL}(\ell, \mathbb{F}) \times \mathrm{GL}(n, \mathbb{F}) \times \mathrm{GL}(m, \mathbb{F})$ sends a 3-way array $\mathtt{A} \in \mathrm{M}(\ell \times n \times m, \mathbb{F})$ to $P^{\mathrm{t}}\mathtt{A}^{R}Q$, defined as follows. First represent $\mathtt{A}$ as an $m$-tuple of $\ell \times n$ matrices $\mathbf{A} = (A_1, \ldots, A_m) \in \mathrm{M}(\ell \times n, \mathbb{F})^m$. Then $P$ and $Q$ send $\mathbf{A}$ to $P^{\mathrm{t}}\mathbf{A}Q = (P^{\mathrm{t}}A_1Q, \ldots, P^{\mathrm{t}}A_mQ)$, and $R = (r_{i,j})$ sends $\mathbf{A}$ to $(A'_1, \ldots, A'_m)$ where $A'_i = \sum_{j\in[m]} r_{i,j}A_j$. Clearly, the actions of $P$, $Q$, and $R$ commute. The resulting $m$-tuple of $\ell \times n$ matrices obtained by applying $P$, $Q$, and $R$ to $\mathbf{A}$ is then $P^{\mathrm{t}}\mathbf{A}^{R}Q$. Note that up to possibly relabelling indices, the entries of $P^{\mathrm{t}}\mathbf{A}^{R}Q$ are explicitly defined as in Equation 1.

**Useful results.** Let $\mathbf{A} = (A_1, \ldots, A_m), \mathbf{B} = (B_1, \ldots, B_m) \in \mathrm{M}(n, \mathbb{F})^m$. Given $T \in \mathrm{GL}(n, \mathbb{F})$, let $T^{\mathrm{t}}\mathbf{A}T = (T^{\mathrm{t}}A_1T, \ldots, T^{\mathrm{t}}A_mT)$. We say that $\mathbf{A}$ and $\mathbf{B}$ are *isometric*, if there exists $T \in \mathrm{GL}(n, \mathbb{F})$ such that $T^{\mathrm{t}}\mathbf{A}T = \mathbf{B}$. Let $\mathrm{Iso}(\mathbf{A}, \mathbf{B}) = \{T \in \mathrm{GL}(n, \mathbb{F}) : \mathbf{A} = T^{\mathrm{t}}\mathbf{B}T\}$, and set $\mathrm{Aut}(\mathbf{A}) := \mathrm{Iso}(\mathbf{A}, \mathbf{A})$. Clearly, $\mathrm{Aut}(\mathbf{A})$ is a subgroup of $\mathrm{GL}(n, q)$, and $\mathrm{Iso}(\mathbf{A}, \mathbf{B})$ is a coset of $\mathrm{Aut}(\mathbf{A})$.

▶ **Theorem 4** ([7, 12]). *Let $\mathbf{A}, \mathbf{B} \in \mathrm{S}(n, q)^m$ (resp. $\Lambda(n, q)^m$) for some odd $q$. There exists a $\mathrm{poly}(n, m, q)$-time deterministic algorithm which takes $\mathbf{A}$ and $\mathbf{B}$ as inputs and outputs $\mathrm{Iso}(\mathbf{A}, \mathbf{B})$, specified by (if nonempty) a generating set of $\mathrm{Aut}(\mathbf{A})$ (by the algorithm in [7]) and a coset representative $T \in \mathrm{Iso}(\mathbf{A}, \mathbf{B})$ (by the algorithm in [12]).*

## 3 Average-case algorithms for polynomial isomorphism and more

We shall present the algorithm for the cubic form isomorphism problem in detail in Section 3.1. We will state our results for problems like algebra isomorphism in Section 3.2.

### 3.1 Cubic form isomorphism over fields of odd order

We present the algorithm for cubic form isomorphism over fields of odd characteristic, as this algorithm already captures the essence of the idea, and cubic forms are most interesting from the PI perspective as mentioned in Section 1.2. A full proof of Theorem 1, which is a relatively minor extension of Theorem 5, can be found in the full version of this paper.

▶ **Theorem 5.** *Let $\mathbb{F}_q$ be a finite field of odd order, and $X = \{x_1, \ldots, x_n\}$ be a set of commuting variables. Let $f, g \in \mathbb{F}_q[X]$ be two cubic forms. There exists a deterministic algorithm that decides whether $f$ and $g$ are isomorphic in time $q^{O(n)}$, for all but at most $\frac{1}{q^{\Omega(n)}}$ fraction of $f$.*

**Proof.** Let $r$ be a constant to be determined later on, and suppose $n$ is sufficiently larger than $r$. (Indeed, by Lemma 6, the constant $r$ can be taken as 8.) Our goal is to find $T \in \mathrm{GL}(n, q)$, such that $f = g \circ T$.

The algorithm consists of two main steps. Let us first give an overview of the two steps.

In the first step, we show that there exists a set of at most $q^{O(rn)}$-many $T_1 \in \mathrm{GL}(n, q)$, such that every $T \in \mathrm{GL}(n, q)$ can be written as $T_1T_2$, where $T_2$ is of the form

$$\begin{bmatrix} I_r & 0 \\ 0 & R \end{bmatrix}. \tag{2}$$

Furthermore, such $T_1$ can be enumerated in time $q^{O(rn)}$. We then set $g_1 = g \circ T_1$.

In the second step, we focus on searching for $T_2$ such that $f = g_1 \circ T_2$. The key observation is that those $T_2$ as in Equation 2 leave $x_i$, $i \in [r]$, invariant, and send $x_j$, $j \in [r+1, n]$, to a linear combination of $x_k$, $k \in [r+1, n]$. It follows that for any fixed $i \in [r]$, $T_2$ sends $\sum_{r+1\le j\le k\le n} a_{i,j,k}x_ix_jx_k$ to a linear combination of $x_ix_jx_k$, $r+1 \le j \le k \le n$. We will

use this observation to show that for a random $f$, the number of $T_2$ satisfying $f = g_1 \circ T_2$ is upper bounded by $q^n$ with high probability. Furthermore, such $T_2$, if they exist, can be enumerated efficiently. This allows us to go over all possible $T_2$ and test if $f = g_1 \circ T_2$.

**The first step.** We show that there exist at most $q^{O(rn)}$-many $T_1 \in \mathrm{GL}(n,q)$, such that any $T \in \mathrm{GL}(n,q)$ can be written as $T_1T_2$ where $T_2$ is of the form as in Equation 2.

Recall that $e_i$ is the $i$th standard basis vector. Let $E_r = \langle e_1, \ldots, e_r\rangle$, and let $F_r = \langle e_{r+1}, \ldots, e_n\rangle$. Suppose for $i \in [r]$, $T(e_i) = u_i$, and $T(F_r) = V \leq \mathbb{F}_q^n$. Let $T_1$ be any matrix that satisfies $T_1(e_i) = u_i$, and $T_1(F_r) = V$. Let $T_2 = T_1^{-1}T$. Then $T_2$ satisfies that for $i \in [r]$, $T_2(e_i) = e_i$, and $T_2(F_r) = F_r$. In other words, $T_2$ is of the form in Equation 2.

We then need to show that these $T_1$ can be enumerated in time $q^{O(rn)}$.

Recall that $T_1$ is determined by the images of $e_i$, $i \in [r]$, and $F_r \leq \mathbb{F}_q^n$. So we first enumerate matrices of the form $\begin{bmatrix} u_1 & \ldots & u_r & e_{r+1} & \ldots & e_n \end{bmatrix}$, where $u_i \in \mathbb{F}_q^n$ are linearly independent. We then need to enumerate the possible images of $F_r$. Let $U = \langle u_1, \ldots, u_r\rangle$. Then the image of $F_r$ is a complement subspace of $U$. It is well-known that the number of complement subspaces of a dimension-$r$ space is $\sim q^{r(n-r)}$. To enumerate all complement subspaces of $U$, first compute one complement subspace $V = \langle v_1, \ldots, v_{n-r}\rangle$. Then it is easy to verify that, when going over $A = (a_{i,j})_{i\in[r],j\in[n-r]} \in \mathrm{M}(r \times (n-r), q)$, $\langle v_j + \sum_{i\in[r]} a_{i,j}u_i : j \in [n-r]\rangle$ go over all complement subspaces of $U$. It follows that we can enumerate matrices $T_1$ of the form $\begin{bmatrix} u_1 & \ldots & u_r & v_1 + \sum_{i\in[r]} a_{i,1}u_i & \ldots & v_{n-r} + \sum_{i\in[r]} a_{i,n-r}u_i \end{bmatrix}$.

**The second step.** In Step 1, we computed a set of invertible matrices $\{T_1\} \subseteq \mathrm{GL}(n,q)$ such that every $T \in \mathrm{GL}(n,q)$ can be written as $T = T_1T_2$ where $T_2 = \begin{bmatrix} I_r & 0 \\ 0 & R \end{bmatrix}$. So we set $h := g \circ T_1$ and focus on finding $T_2$ of the above form such that $f = h \circ T_2$.

Suppose $f = \sum_{1\leq i\leq j\leq k\leq n} \alpha_{i,j,k}x_ix_jx_k$, and $h = \sum_{1\leq i\leq j\leq k\leq n} \beta_{i,j,k}x_ix_jx_k$. For $i \in [r]$, define $f_i = \sum_{r+1\leq j\leq k\leq n} \alpha_{i,j,k}x_ix_jx_k$. Similarly define $h_i$.

The key observation is that, due to the form of $T_2$, we have that $f_i = h_i \circ T_2$. This is because for $i \in [r]$, $T_2$ sends $x_i$ to $x_i$, and for $j \in [r+1, n]$, $T_2$ sends $x_j$ to a linear combination of $x_k$, $k \in [r+1, n]$.

Let $\ell = n - r$. We then rename the variable $x_{r+i}$, $i \in [\ell]$ as $y_i$. Let $Y = \{y_1, \ldots, y_\ell\}$. Then from $f$, we define $r$ quadratic forms in $Y$,

$$\forall i \in [r], c_i = \sum_{1\leq j\leq k\leq \ell} \alpha'_{i,j,k}y_jy_k, \text{ where } \alpha'_{i,j,k} = \alpha_{i,r+j,r+k}. \tag{3}$$

Correspondingly, we define $r$ quadratic forms $d_i = \sum_{1\leq j\leq k\leq \ell} \beta'_{i,j,k}y_jy_k$, $i \in [r]$, from $g_1$.

Our task now is to search for the $R \in \mathrm{GL}(\ell,q)$ such that for every $i \in [r]$, $c_i = d_i \circ R$.

To do that, we adopt the classical representation of quadratic forms as symmetric matrices. Here we use the assumption that $q$ is odd. Using the classical correspondence between quadratic forms and symmetric matrices, from $c_i$ we construct

$$C_i = \begin{bmatrix} \alpha'_{i,1,1} & \frac{1}{2}\alpha'_{i,1,2} & \cdots & \frac{1}{2}\alpha'_{i,1,\ell} \\ \frac{1}{2}\alpha'_{i,1,2} & \alpha'_{i,2,2} & \cdots & \frac{1}{2}\alpha'_{i,2,\ell} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}\alpha'_{i,1,\ell} & \frac{1}{2}\alpha'_{i,2,\ell} & \cdots & \alpha'_{i,\ell,\ell} \end{bmatrix} \in \mathrm{S}(\ell,q). \tag{4}$$

Similarly define $D_i$ from $d_i$. It is classical that $c_i = d_i \circ R$ if and only if $C_i = R^{\mathrm{t}}D_iR$.

Let $\mathbf{C} = (C_1, \ldots, C_r) \in \mathrm{S}(\ell, q)^r$, and $\mathbf{D} = (D_1, \ldots, D_r) \in \mathrm{S}(\ell, q)^r$. Recall that $\mathrm{Aut}(\mathbf{C}) = \{R \in \mathrm{GL}(\ell, \mathbb{F}) : R^{\mathrm{t}}\mathbf{C}R = \mathbf{C}\}$, and $\mathrm{Iso}(\mathbf{C}, \mathbf{D}) = \{R \in \mathrm{GL}(\ell, \mathbb{F}) : \mathbf{C} = R^{\mathrm{t}}\mathbf{D}R\}$. Clearly, $\mathrm{Iso}(\mathbf{C}, \mathbf{D})$ is a (possibly empty) coset of $\mathrm{Aut}(\mathbf{C})$. When $\mathrm{Iso}(\mathbf{C}, \mathbf{D})$ is non-empty, $|\mathrm{Iso}(\mathbf{C}, \mathbf{D})| = |\mathrm{Aut}(\mathbf{C})|$. Our main technical lemma is the following, obtained by adapting certain results in [6, 15] to the symmetric matrix setting. Its proof can be found in the full version of this paper.

▶ **Lemma 6.** *Let $\mathbf{C} = (C_1, \ldots, C_8) \in \mathrm{S}(\ell, q)^8$ be a random symmetric matrix tuple. Then we have $|\mathrm{Aut}(\mathbf{C})| \leq q^\ell$ for all but at most $\frac{1}{q^{\Omega(\ell)}}$ fraction of such $\mathbf{C}$.*

Given this lemma, we can use Theorem 4 to decide whether $\mathbf{C}$ and $\mathbf{D}$ are isometric, and if so, compute $\mathrm{Iso}(\mathbf{C}, \mathbf{D})$ represented as a coset in $\mathrm{GL}(\ell, q)$. By Lemma 6, for all but at most $\frac{1}{q^{\Omega(\ell)}}$ fraction of $\mathbf{C}$, $|\mathrm{Iso}(\mathbf{C}, \mathbf{D})| \leq q^\ell \leq q^n$. With $\mathrm{Iso}(\mathbf{C}, \mathbf{D})$ as a coset at hand, we can enumerate all elements in $\mathrm{Aut}(\mathbf{C})$ by the standard recursive closure algorithm [16] and therefore all elements in $\mathrm{Iso}(\mathbf{C}, \mathbf{D})$. We then either conclude that $|\mathrm{Iso}(\mathbf{C}, \mathbf{D})| > q^n$, or have all $\mathrm{Iso}(\mathbf{C}, \mathbf{D})$ at hand. In the former case we conclude that $\mathbf{C}$ does not satisfy the required generic condition. In the latter case, we enumerate $R \in \mathrm{Iso}(\mathbf{C}, \mathbf{D})$, and check whether $T_2 = \begin{bmatrix} I_r & 0 \\ 0 & R \end{bmatrix}$ is an isomorphism from $f$ to $g_1$.

**The algorithm outline.** We now summarise the above steps in the following algorithm outline. In the following we assume that $n \gg 8$; otherwise we can use the brute-force algorithm.

**Input** Cubic forms $f, g \in \mathbb{F}_q[x_1, \ldots, x_n]$.

**Output** One of the following: (1) "$f$ does not satisfy the generic condition"; (2) "$f$ and $g$ are not isomorphic"; (3) an isomorphism $T \in \mathrm{GL}(n, q)$ sending $g$ to $f$.

**Algorithm outline 1.** Set $r = 8$, and $\ell = n - r$.

2. Compute $W = \{T_1\} \subseteq \mathrm{GL}(n, q)$ using the procedure described in Step 1.
   // `Every` $T \in \mathrm{GL}(n, q)$ `can be written as` $T_1T_2$ `where` $T_2$ `is of the form in Equation 2.`
3. For every $T_1 \in W$, do the following:
   a. $h :\leftarrow g \circ T_1$.
   b. For $i \in [\ell]$, $y_i \leftarrow x_{r+i}$.
   c. For $i \in [r]$, let $C_i \in \mathrm{S}(\ell, q)$ be defined in Equation 4. Let $D_i \in \mathrm{S}(\ell, q)$ be defined from $h$ in the same way. Let $\mathbf{C} = (C_1, \ldots, C_r)$, and $\mathbf{D} = (D_1, \ldots, D_r)$.
   d. Use Theorem 4 to decide whether $\mathbf{C}$ and $\mathbf{D}$ are isometric. If not, break from the loop. If so, compute one isometry $R$.
   e. Use Theorem 4 to compute a generating set of $\mathrm{Aut}(\mathbf{C})$. Use the recursive closure algorithm to enumerate $\mathrm{Aut}(\mathbf{C})$. During the enumertion, if $|\mathrm{Aut}(\mathbf{C})| > q^\ell$, report "$f$ does not satisfy the generic condition." Otherwise, we have the whole $\mathrm{Aut}(\mathbf{C})$ at hand, which is of size $\leq q^\ell$.
   f. Given $R$ from Line 3d and $\mathrm{Aut}(\mathbf{C})$ from Line 3e, the whole set $\mathrm{Iso}(\mathbf{C}, \mathbf{D})$ can be computed. For every $R \in \mathrm{Iso}(\mathbf{C}, \mathbf{D})$, check whether $T_2 = \begin{bmatrix} I_r & 0 \\ 0 & R \end{bmatrix}$ sends $h$ to $f$. If so, return $T = T_1T_2$ as an isomorphism sending $g$ to $f$.
4. Return that "$f$ and $g$ are not isomorphic".

**Correctness and timing analyses.** The correctness of the algorithm relies on the simple fact that if $f$ satisfies the genericity condition, and $f$ and $g$ are isomorphic via some $T \in \mathrm{GL}(n,q)$, then this $T$ can be decomposed as $T_1T_2$ for some $T_1 \in W$ from Line 2. Then by the analysis in Step 2, $T_2 = \begin{bmatrix} I_r & 0 \\ 0 & R \end{bmatrix}$ where $R \in \mathrm{Iso}(\mathbf{C},\mathbf{D})$. When $f$ satisfies the genericity condition, $\mathrm{Iso}(\mathbf{C},\mathbf{D})$ will be enumerated, so this $R$ will surely be encountered.

To estimate the time complexity of the algorithm, note that $|W| \leq q^{O(rn)}$, and $|\mathrm{Iso}(\mathbf{C},\mathbf{D})| \leq q^{\ell} = q^{n-r}$. As other steps are performed in time $\mathrm{poly}(n,m,q)$, enumerating over $W$ and $\mathrm{Iso}(\mathbf{C},\mathbf{D})$ dominates the time complexity. Recall that $r = 8$. So the total time complexity is upper bounded by $q^{O(n)}$. ◀

## 3.2 Trilinear form equivalence and algebra isomorphism

We describe our results on trilinear form equivalence and algebra isomorphism, and leave the modifications required to achieve these results in the full version of this paper.

**Trilinear form equivalence.** The trilinear form equivalence problem was stated in Section 1.2. In algorithms, a trilinear form $f$ is naturally represented as a 3-way array $\mathtt{A} = (a_{i,j,k})$ where $a_{i,j,k} = f(e_i, e_j, e_k)$. A random trilinear form over $\mathbb{F}_q$ denotes the setting when $\alpha_{i,j,k}$ are independently sampled from $\mathbb{F}_q$ uniformly at random.

▶ **Theorem 7.** *Let $f : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ be a random trilinear form, and let $g : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ be an arbitrary trilinear form. There exists a deterministic algorithm that decides whether $f$ and $g$ are equivalent in time $q^{O(n)}$, for all but at most $\frac{1}{q^{\Omega(n)}}$ fraction of $f$.*

**Algebra isomorphism.** Let $V$ be a vector space. An algebra is a bilinear map $* : V \times V \to V$. This bilinear map $*$ is considered as the product. Algebras most studied are those with certain conditions on the product, including unital ($\exists v \in V$ such that $\forall u \in V$, $v * u = u$), associative ($(u * v) * w = u * (v * w)$), and commutative ($u * v = v * u$). The authors of [1, 2] study algebras satisfying these conditions. Here we consider algebras without such restrictions. Two algebras $*, \cdot : V \times V \to V$ are *isomorphic*, if there exists $T \in \mathrm{GL}(V)$, such that $\forall u, v \in V$, $T(u) * T(v) = T(u \cdot v)$. As customary in computational algebra, an algebra is represented by its structure constants, i.e. suppose $V \cong \mathbb{F}^n$, and fix a basis $\{e_1, \ldots, e_n\}$. Then $e_i * e_j = \sum_{k \in [n]} \alpha_{i,j,k} e_k$, and this 3-way array $\mathtt{A} = (\alpha_{i,j,k})$ records the structure constants of the algebra with product $*$. A random algebra over $\mathbb{F}_q$ denotes the setting when $\alpha_{i,j,k}$ are independently sampled from $\mathbb{F}_q$ uniformly at random.

▶ **Theorem 8.** *Let $f : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^n$ be a random algebra, and let $g : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^n$ be an arbitrary algebra. There exists a deterministic algorithm that decides whether $f$ and $g$ are isomorphic in time $q^{O(n)}$, for all but at most $\frac{1}{q^{\Omega(n)}}$ fraction of $f$.*

# 4 Complexity of symmetric and alternating trilinear form equivalence

As mentioned in Section 1.4, the proof of Theorem 2 follows by showing that symmetric and alternating trilinear form equivalence are TI-hard (recall Definition 3). In the following we focus on the alternating case. The symmetric case can be tackled in a straightforward way, by starting from the TI-complete problem of symmetric matrix tuple pseudo-isometry, from [11, Theorem B], and modifying the alternating gadget to a symmetric one.

▶ **Proposition 9.** *The alternating trilinear form equivalence problem is* TI*-hard.*

**Proof. The starting TI-complete problem.** We use the following TI-complete problem from [11]. Let $\mathbf{A} = (A_1, \ldots, A_m), \mathbf{B} = (B_1, \ldots, B_m) \in \Lambda(n, \mathbb{F})^m$ be two tuples of alternating matrices. We say that $\mathbf{A}$ and $\mathbf{B}$ are pseudo-isometric, if there exist $C \in \mathrm{GL}(n, \mathbb{F})$ and $D = (d_{i,j}) \in \mathrm{GL}(m, \mathbb{F})$, such that for any $i \in [m]$, $C^{\mathrm{t}}(\sum_{j\in[m]} d_{i,j}A_j)C = B_i$. By [11, Theorem B], the alternating matrix tuple pseudo-isometry problem is TI-complete. Without loss of generality, we assume that $\dim(\langle A_i\rangle) = \dim(\langle B_i\rangle)$, as if not, then they cannot be pseudo-isometric, and this dimension condition is easily checked.

An alternating trilinear form $\phi : \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ naturally corresponds to a 3-way array $\mathtt{A} = (a_{i,j,k}) \in \mathrm{M}(n \times n \times n, \mathbb{F})$, where $a_{i,j,k} = \phi(e_i, e_j, e_k)$. Then $\mathtt{A}$ is also alternating, i.e. $a_{i,j,k} = 0$ if $i = j$ or $i = k$ or $j = k$, and $a_{i,j,k} = \mathrm{sgn}(\sigma)a_{\sigma(i),\sigma(j),\sigma(k)}$ for any $\sigma \in \mathrm{S}_3$. So in the following, we present a construction of an alternating 3-way array from an alternating matrix tuple, in such a way that two alternating matrix tuples are pseudo-isometric if and only if the corresponding alternating trilinear forms are equivalent.

**Constructing alternating 3-way arrays from alternating matrix tuples.** Given $\mathbf{A} = (A_1, \ldots, A_m) \in \Lambda(n, \mathbb{F})^m$, we first build the $n \times n \times m$ tensor $\mathtt{A}$ which has $A_1, \ldots, A_m$ as its frontal slices. Then we will use essentially the following construction twice in succession. We will give two viewpoints on this construction: one algebraic, in terms of trilinear forms, and another "matricial", in terms of 3-way arrays. Different readers may prefer one viewpoint over the other; our opinion is that the algebraic view makes it easier to verify the alternating property while the matricial view makes it easier to verify the reduction. We thank an anonymous review for the suggestion of the algebraic viewpoint. The construction is, in some sense, the 3-tensor analogue of taking an ordinary matrix $A$ and building the alternating matrix $\begin{bmatrix} 0 & A \\ -A^{\mathrm{t}} & 0 \end{bmatrix}$.

*Notation:* Just as the transpose acts on matrices by $(A^{\mathrm{t}})_{i,j} = A_{j,i}$, for a 3-tensor $\mathtt{A}$, we have six possible "transposes" corresponding to the six permutations of the three coordinates. Given $\sigma \in S_3$, we write $\mathtt{A}^\sigma$ for the 3-tensor defined by $(\mathtt{A}^\sigma)_{i_1,i_2,i_3} = \mathtt{A}_{i_{\sigma(1)},i_{\sigma(2)},i_{\sigma(3)}}$. Similarly, given a trilinear form $A(x, y, z) = \sum_{i,j,k} a_{i,j,k}x_iy_jz_k$, we have $A^\sigma(x, y, z) = A((x, y, z)^\sigma)$.

Given a 3-way array $\mathtt{A} \in \mathrm{M}(n \times m \times d, \mathbb{F})$, we will make use of $\mathtt{A}^{(23)}$ and $\mathtt{A}^{(13)}$:

- $\mathtt{A}^{(23)}$ is $n \times d \times m$ and has $\mathtt{A}^{(23)}_{i,j,k} = \mathtt{A}_{i,k,j}$. Equivalently, the $k$-th frontal slice of $\mathtt{A}^{(23)}$ is the $k$-th vertical slice of $\mathtt{A}$.
- $\mathtt{A}^{(13)}$ is $d \times m \times n$ and has $\mathtt{A}^{(13)}_{i,j,k} = \mathtt{A}_{k,j,i}$. Equivalently, the $k$-th frontal slice of $\mathtt{A}^{(13)}$ is the transpose of the $k$-th horizontal slice of $\mathtt{A}$.

▶ **Example 10** (Running example)**.** Let us examine a simple example as follows. Let $\mathbf{A} = (A) \in \Lambda(2, \mathbb{F})^1$, where $A = \begin{bmatrix} 0 & a \\ -a & 0 \end{bmatrix}$. Then $\mathtt{A} = (A)$; $\mathtt{A}^{(23)} = (A'_1, A'_2) \in \mathrm{M}(2 \times 1 \times 2, \mathbb{F})$, where $A'_1 = \begin{bmatrix} 0 \\ -a \end{bmatrix}$, and $A'_2 = \begin{bmatrix} a \\ 0 \end{bmatrix}$; $\mathtt{A}^{(13)} = (A''_1, A''_2) \in \mathrm{M}(1 \times 2 \times 2, \mathbb{F})$, where $A''_1 = \begin{bmatrix} 0 & a \end{bmatrix}$, and $A''_2 = \begin{bmatrix} -a & 0 \end{bmatrix}$.

From the above $\mathtt{A}$, $\mathtt{A}^{(23)}$, and $\mathtt{A}^{(13)}$, we construct $\tilde{\mathtt{A}} \in \mathrm{M}((n + m) \times (n + m) \times (n + m), \mathbb{F})$ as follows. We divide $\tilde{\mathtt{A}}$ into the following eight blocks. Set $\tilde{\mathtt{A}} = (\tilde{\mathtt{A}}_1, \tilde{\mathtt{A}}_2)$ (two block frontal slices) where $\tilde{\mathtt{A}}_1 = \begin{bmatrix} 0_{n\times n\times n} & \mathtt{A}^{(23)} \\ \mathtt{A}^{(13)} & 0 \end{bmatrix}$, and $\tilde{\mathtt{A}}_2 = \begin{bmatrix} -\mathtt{A} & 0 \\ 0 & 0_{m\times m\times m} \end{bmatrix}$, where $0_{n\times n\times n}$ indicates the $n \times n \times n$ zero tensor, and analogously for $0_{m\times m\times m}$ (the remaining sizes can be determined from these and the fact that $\mathtt{A}$ is $n \times n \times m$).

The corresponding construction on trilinear forms is as follows. The original trilinear form is $A(x,y,z) = \sum_{i,j\in[n],k\in[m]} a_{i,j,k}x_iy_jz_k$, where $x = (x_1,\ldots,x_n)$, $y = (y_1,\ldots,y_n)$, and $z = (z_1,\ldots,z_m)$, and we have $A(x,y,z) = -A(y,x,z)$. The new trilinear form will be $\tilde{A}(x',y',z')$, where

$$\begin{aligned} x' = (x^{(1)},x^{(2)}) &= (x_1^{(1)},\ldots,x_n^{(1)},x_1^{(2)},\ldots,x_m^{(2)}) \\ y' = (y^{(1)},y^{(2)}) &= (y_1^{(1)},\ldots,y_n^{(1)},y_1^{(2)},\ldots,y_m^{(2)}) \\ z' = (z^{(1)},z^{(2)}) &= (z_1^{(1)},\ldots,z_n^{(1)},z_1^{(2)},\ldots,z_m^{(2)}). \end{aligned}$$

This new form will satisfy $\tilde{A}(x',y',z') = \sum_{i,j,k\in[n+m]} \tilde{a}_{i,j,k}x'_iy'_jz'_k$. Let us unravel what this looks like from the above description of $\tilde{\mathtt{A}}$. We have

$$\begin{aligned} \tilde{A}(x',y',z') &= \sum_{i\in[n],j\in[m],k\in[n]} (\tilde{\mathtt{A}}_1)_{i,n+j,k}x'_iy'_{n+j}z'_k + \sum_{i\in[m],j,k\in[n]} (\tilde{\mathtt{A}}_1)_{n+i,j,k}x'_{n+i}y'_jz'_k \\ &\quad + \sum_{i,j\in[n],k\in[m]} (\tilde{\mathtt{A}}_2)_{i,j,k}x'_iy'_jz'_{n+k} \\ &= \sum_{i\in[n],j\in[m],k\in[n]} \mathtt{A}^{(23)}_{i,j,k}x'_iy'_{n+j}z'_k + \sum_{i\in[m],j,k\in[n]} \mathtt{A}^{(13)}_{i,j,k}x'_{n+i}y'_jz'_k \\ &\quad - \sum_{i,j\in[n],k\in[m]} \mathtt{A}_{i,j,k}x'_iy'_jz'_{n+k} \\ &= \sum_{i\in[n],j\in[m],k\in[n]} \mathtt{A}_{i,k,j}x'_iy'_{n+j}z'_k + \sum_{i\in[m],j,k\in[n]} \mathtt{A}_{k,j,i}x'_{n+i}y'_jz'_k \\ &\quad - \sum_{i,j\in[n],k\in[m]} \mathtt{A}_{i,j,k}x'_iy'_jz'_{n+k} \\ &= A(x^{(1)},z^{(1)},y^{(2)}) + A(z^{(1)},y^{(1)},x^{(2)}) - A(x^{(1)},y^{(1)},z^{(2)}). \end{aligned}$$

From this formula, and the fact that $A(x,y,z) = -A(y,x,z)$, we can now more easily verify that $\tilde{A}$ is alternating in all three arguments. Since the permutations (13) and (23) generate $S_3$, it suffices to verify it for these two. We have

$$\begin{aligned} \tilde{A}^{(13)}(x',y',z') &= \tilde{A}(z',y',x') \\ &= A(z^{(1)},x^{(1)},y^{(2)}) + A(x^{(1)},y^{(1)},z^{(2)}) - A(z^{(1)},y^{(1)},x^{(2)}) \\ &= -A(x^{(1)},z^{(1)},y^{(2)}) + A(x^{(1)},y^{(1)},z^{(2)}) - A(z^{(1)},y^{(1)},x^{(2)}) \\ &= -\tilde{A}(x',y',z'). \end{aligned}$$

Similarly, we have:

$$\begin{aligned} \tilde{A}^{(23)}(x',y',z') &= \tilde{A}(x',z',y') \\ &= A(x^{(1)},y^{(1)},z^{(2)}) + A(y^{(1)},z^{(1)},x^{(2)}) - A(x^{(1)},z^{(1)},y^{(2)}) \\ &= A(x^{(1)},y^{(1)},z^{(2)}) - A(z^{(1)},y^{(1)},x^{(2)}) - A(x^{(1)},z^{(1)},y^{(2)}) \\ &= -\tilde{A}(x',y',z'), \end{aligned}$$

as claimed.

▶ **Example 11** (Running example, continued from Example 10)**.** We can write out $\tilde{\mathtt{A}}$ in this case explicitly. The first block frontal slice $\tilde{\mathtt{A}}_1$ is $3\times3\times2$, consisting of the two frontal slices

$$\left(\begin{array}{cc:c} 0 & 0 & 0 \\ 0 & 0 & -a \\ \hdashline 0 & a & 0 \end{array}\right) \text{ and } \left(\begin{array}{cc:c} 0 & 0 & a \\ 0 & 0 & 0 \\ \hdashline -a & 0 & 0 \end{array}\right)$$

while the second block frontal slice $\tilde{\mathtt{A}}_2$ is the $3 \times 3 \times 1$ matrix

$$\left(\begin{array}{cc:c} 0 & -a & 0 \\ a & 0 & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right)$$

It can be verified easily that $\tilde{\mathtt{A}} = (a_{i,j,k})$ is alternating: the nonzero entries are $a_{2,3,1} = -a$, $a_{3,2,1} = a$, $a_{1,3,2} = a$, $a_{3,1,2} = -a$, $a_{1,2,3} = -a$, and $a_{2,1,3} = a$, which are consistent with the signs of the permutations.

**The gadget construction.** We now describe the gadget construction. The gadget can be described as a block 3-way array as follows. Construct a 3-way array $\mathtt{G}$ of size $(n+1)^2 \times (n+1)^2 \times (n+m)$ over $\mathbb{F}$ as follows. For $i \in [n]$, the $i$th frontal slice of $\mathtt{G}$ is

$$\begin{bmatrix} 0 & 0 & \dots & 0 & I_{n+1} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ -I_{n+1} & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{bmatrix},$$

where 0 here denotes the $(n+1) \times (n+1)$ all-zero matrix, $I_{n+1}$ is at the $(1, i+1)$th block position, and $-I_{n+1}$ is at the $(i+1, 1)$th block position. For $n+1 \le i \le n+m$, the $i$th frontal slice of $\mathtt{G}$ is the all-zero matrix. We also need the following 3-way arrays derived from $\mathtt{G}$. We will use $\mathtt{G}^{(13)}$ and $\mathtt{G}^{(23)}$. Note that $\mathtt{G}^{(13)}$ is of size $(n+m) \times (n+1)^2 \times (n+1)^2$, and its $i$th horizontal slice is the $i$th frontal slice of $\mathtt{G}$. Similarly, $\mathtt{G}^{(23)}$ is of size $(n+1)^2 \times (n+m) \times (n+1)^2$, and its $j$th vertical slice is the $j$th frontal slice of $\mathtt{G}$.

Finally, construct a 3-tensor $\hat{\mathtt{A}}$ as follows. It consists of the two block frontal slices

$$\begin{bmatrix} \tilde{\mathtt{A}} & 0 \\ 0 & -\mathtt{G} \end{bmatrix} \text{ and } \begin{bmatrix} 0 & \mathtt{G}^{(13)} \\ \mathtt{G}^{(23)} & 0 \end{bmatrix}.$$

To see how this all fits together, let $\mathtt{G}_1$ be the $(n+1)^2 \times (n+1)^2 \times n$ tensor consisting of the first $n$ frontal slices of $\mathtt{G}$ (these are the only nonzero frontal slices of $\mathtt{G}$). Then we may view $\hat{\mathtt{A}}$ as having three block frontal slices, namely:

$$\begin{bmatrix} 0_{n\times n\times n} & \mathtt{A}^{(23)} & 0 \\ \mathtt{A}^{(13)} & 0_{m\times m\times n} & 0 \\ 0 & 0 & -\mathtt{G}_1 \end{bmatrix}, \begin{bmatrix} -\mathtt{A} & 0 & 0 \\ 0 & 0_{m\times m\times m} & 0 \\ 0 & 0 & 0_{(n+1)^2\times(n+1)^2\times m} \end{bmatrix},$$

and

$$\begin{bmatrix} 0_{n\times n\times (n+1)^2} & 0 & \mathtt{G}_1^{(13)} \\ 0 & 0_{m\times m\times (n+1)^2} & 0 \\ \mathtt{G}_1^{(23)} & 0 & 0 \end{bmatrix}.$$

We claim that $\hat{\mathtt{A}}$ is alternating. To verify this is straightforward but somewhat tedious. So we use the following example from which a complete proof can be extracted easily.

▶ **Example 12** (Running example, continued from Example 11)**.** Let $\mathtt{A}$ be the $2 \times 2 \times 1$ tensor with alternating frontal slice $A = \begin{bmatrix} 0 & a \\ -a & 0 \end{bmatrix}$. In particular, $n = 2, m = 1$, so $\mathtt{G}$ will have size $(n+1)^2 \times (n+1)^2 \times (n+m) = 9 \times 9 \times 3$, and $\mathtt{A}$ will have size $n + m + (n+1)^2 = 12$ in all three directions. We will write out the first $n + m = 3$ frontal slices explicitly, as those are the only ones involving $\mathtt{A}$, and leave the last 9 (involving only transposes of $\mathtt{G}_1$) unwritten.

$$\left(\begin{array}{cc:c:ccc} 0 & 0 & 0 & & & \\ 0 & 0 & -a & & & \\ \hdashline 0 & a & 0 & & & \\ \hdashline & & & 0_3 & I_3 & 0 \\ & & & -I_3 & 0_3 & 0 \\ & & & 0 & 0 & 0_3 \end{array}\right), \left(\begin{array}{cc:c:ccc} 0 & 0 & a & & & \\ 0 & 0 & 0 & & & \\ \hdashline -a & 0 & 0 & & & \\ \hdashline & & & 0_3 & 0 & I_3 \\ & & & 0 & 0_3 & 0 \\ & & & -I_3 & 0 & 0_3 \end{array}\right),$$

and $$\left(\begin{array}{cc:c:ccc} 0 & a & 0 & & & \\ -a & 0 & 0 & & & \\ \hdashline 0 & 0 & 0 & & & \\ \hdashline & & & 0_3 & 0 & 0 \\ & & & 0 & 0_3 & 0 \\ & & & 0 & 0 & 0_3 \end{array}\right)$$

and the remaining 9 frontal slices look like

$$\left(\begin{array}{cc:c:ccc} 0 & 0 & 0 & & & \\ 0 & 0 & 0 & & \mathtt{G}_1^{(13)} & \\ \hdashline 0 & 0 & 0 & & 0_{1\times 9\times 9} & \\ \hdashline & & & 0_{3\times 3\times 9} & 0 & 0 \\ \mathtt{G}_1^{(23)} & & 0_{9\times 1\times 9} & 0 & 0_{3\times 3\times 9} & 0 \\ & & & 0 & 0 & 0_{3\times 3\times 9} \end{array}\right)$$

Since the $a$'s only appear in positions with the same indices as they did in $\tilde{\mathtt{A}}$ (see Example 11), that portion is still alternating. For the $\mathtt{G}$ parts, note that the identity matrices in the first three frontal slices, when having their indices transposed, end up either in the $\mathtt{G}_1^{(13)}$ portion or the $\mathtt{G}_1^{(23)}$ portion, with appropriate signs.

**Proof of correctness.** Let $\mathbf{A}, \mathbf{B} \in \Lambda(n, \mathbb{F})^m$. Let $\hat{\mathtt{A}} = \left(\begin{bmatrix} \tilde{\mathtt{A}} & 0 \\ 0 & -\mathtt{G} \end{bmatrix}, \begin{bmatrix} 0 & \mathtt{G}^{(13)} \\ \mathtt{G}^{(23)} & 0 \end{bmatrix}\right)$, $\hat{\mathtt{B}} = \left(\begin{bmatrix} \tilde{\mathtt{B}} & 0 \\ 0 & -\mathtt{G} \end{bmatrix}, \begin{bmatrix} 0 & \mathtt{G}^{(13)} \\ \mathtt{G}^{(23)} & 0 \end{bmatrix}\right) \in \mathrm{M}((n+m+(n+1)^2) \times (n+m+(n+1)^2) \times (n+m+(n+1)^2), \mathbb{F})$ be constructed from $\mathbf{A}$ and $\mathbf{B}$ using the procedure above, respectively.

We claim that $\mathbf{A}$ and $\mathbf{B}$ are pseudo-isometric if and only if $\hat{\mathtt{A}}$ and $\hat{\mathtt{B}}$ are equivalent as trilinear forms.

**The only if direction.** Suppose $P^{\mathrm{t}}\mathbf{A}P = \mathbf{B}^Q$ for some $P \in \mathrm{GL}(n, \mathbb{F})$ and $Q \in \mathrm{GL}(m, \mathbb{F})$.

We will construct a trilinear form equivalence from $\hat{\mathtt{A}}$ to $\hat{\mathtt{B}}$ of the form $S = \begin{bmatrix} P & 0 & 0 \\ 0 & Q^{-1} & 0 \\ 0 & 0 & R \end{bmatrix} \in \mathrm{GL}(n + m + (n+1)^2, \mathbb{F})$, where $R \in \mathrm{GL}((n+1)^2, \mathbb{F})$ is to be determined later on.

Recall that $\hat{\mathtt{A}} = (\begin{bmatrix}\tilde{\mathtt{A}} & 0\\ 0 & -\mathtt{G}\end{bmatrix}, \begin{bmatrix}0 & \mathtt{G}^{(13)}\\ \mathtt{G}^{(23)} & 0\end{bmatrix})$, $\hat{\mathtt{B}} = (\begin{bmatrix}\tilde{\mathtt{B}} & 0\\ 0 & -\mathtt{G}\end{bmatrix}, \begin{bmatrix}0 & \mathtt{G}^{(13)}\\ \mathtt{G}^{(23)} & 0\end{bmatrix})$. It can be verified that the action of $S$ sends $\tilde{\mathtt{A}}$ to $\tilde{\mathtt{B}}$. It remains to show that, by choosing an appropriate $R$, the action of $S$ also sends $\mathtt{G}$ to $\mathtt{G}$.

Let $\mathtt{G}_1$ be the first $n$ frontal slices of $\mathtt{G}$, and $\mathtt{G}_2$ the last $m$ frontal slices from $\mathtt{G}$. Then the action of $S$ sends $\mathtt{G}_1$ to $R^{\mathrm{t}}\mathtt{G}_1^P R$, and $\mathtt{G}_2$ to $R^{\mathrm{t}}\mathtt{G}_2^{Q^{-1}} R$. Since $\mathtt{G}_2$ is all-zero, the action of $S$ on $\mathtt{G}_2$ results in an all-zero tensor, so we have $R^{\mathrm{t}}\mathtt{G}_2^{Q^{-1}} R = \mathtt{G}_2$.

We then turn to $\mathtt{G}_1$. For $i \in [n+1]$, consider the $i$th horizontal slice of $\mathtt{G}_1$, which is of the form $H_i = \begin{bmatrix}0 & B_{1,i} & B_{2,i} & \dots & B_{n,i}\end{bmatrix}$, where 0 denotes the $n \times (n+1)$ all-zero matrix, and $B_{j,i}$ is the $n \times (n+1)$ elementary matrix with the $(j,i)$th entry being 1, and other entries being 0. Note that those non-zero entries of $H_i$ are in the $(k(n+1)+i)$th columns, for $k \in [n]$. Let $P^{\mathrm{t}} = \begin{bmatrix}p_1 & \dots & p_n\end{bmatrix}$, where $p_i$ is the $i$th column of $P^{\mathrm{t}}$. Then $P$ acts on $H_i$ from the left, which yields $P^{\mathrm{t}}H_i = \begin{bmatrix}0 & P_{1,i} & \dots & P_{n,i}\end{bmatrix}$, where $P_{j,i}$ denotes the $n \times (n+1)$ matrix with the $i$th column being $p_j$, and the other columns being 0.

Let us first set $R = \begin{bmatrix}I_{n+1} & 0\\ 0 & \hat{R}\end{bmatrix}$, where $\hat{R}$ is to be determined later on. Then the left action of $R$ on $\mathtt{G}_1$ preserves $H_i$ through $I_{n+1}$. The right action of $R$ on $\mathtt{G}_1$ translates to the right action of $\hat{R}$ on $H_i$. To send $P^{\mathrm{t}}H_i$ back to $H_i$, $\hat{R}$ needs to act on those $(k(n+1)+i)$th columns of $H_i$, $i \in [n+1]$, as $P^{-1}$. Note that for $H_i$ and $H_j$, $i \neq j$, those columns with non-zero entries are disjoint. This gives $\hat{R}$ the freedom to handle different $H_i$'s separately. In other words, $\hat{R}$ can be set as $P^{-1} \otimes I_{n+1}$. This ensures that for every $H_i$, $P^{\mathrm{t}}H_i\hat{R} = H_i$. To summarize, we have $R^{\mathrm{t}}\mathtt{G}_1^P R = \mathtt{G}_1$, and this concludes the proof for the only if direction.

**The if direction.** Suppose $\hat{\mathtt{A}}$ and $\hat{\mathtt{B}}$ are isomorphic as trilinear forms via $P \in \mathrm{GL}(n+m+(n+1)^2, \mathbb{F})$. Set $P = \begin{bmatrix}P_{1,1} & P_{1,2} & P_{1,3}\\ P_{2,1} & P_{2,2} & P_{2,3}\\ P_{3,1} & P_{3,2} & P_{3,3}\end{bmatrix}$, where $P_{1,1}$ is of size $n \times n$, $P_{2,2}$ is of size $m \times m$, and $P_{3,3}$ is of size $(n+1)^2 \times (n+1)^2$. Consider the ranks of the frontal slices of $\hat{\mathtt{A}}$.

- The ranks of the first $n$ frontal slices are in $[2(n+1), 4n]$. This is because a frontal slice in this range consists of two copies of vertical slices of $\mathtt{A}$ (whose ranks are between $[0, n-1]$ due to the alternating condition), and one frontal slice of $\mathtt{G}$ (whose ranks are of $2(n+1)$).
- The ranks of the $n+1$ to $n+m$ frontal slices are in $[0, n]$. This is because a frontal slice in this range consists of only just one frontal slice of $\mathtt{A}$.
- The ranks of the last $n(n+1)$ vertical slices are in $[0, 2n]$. This is because a frontal slice in this range consists of two copies of horizontal slices of $\mathtt{G}$ (whose ranks are either $n$ or 1; see e.g. the form of $H_i$ in the proof of the only if direction).

By the discussions above, we claim that that $P$ must be of the form $\begin{bmatrix}P_{1,1} & 0 & 0\\ P_{2,1} & P_{2,2} & P_{2,3}\\ P_{3,1} & P_{3,2} & P_{3,3}\end{bmatrix}$. To see this, for the sake of contradiction, suppose there are non-zero entries in $P_{1,2}$ or $P_{1,3}$. Then a non-trivial linear combination of the first $n$ frontal slices is added to one of the last $(m+(n+1)^2)$ frontal slices. This implies that for this slice, the lower-right $(n+1)^2 \times (n+1)^2$ submatrix is of the form $\begin{bmatrix}0 & a_1 I_{n+1} & a_2 I_{n+1} & \dots & a_n I_{n+1}\\ -a_1 I_{n+1} & 0 & 0 & \dots & 0\\ -a_2 I_{n+1} & 0 & 0 & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ -a_n I_{n+1} & 0 & 0 & \dots & 0\end{bmatrix}$, where one of $a_i \in \mathbb{F}$ is non-zero. Then this slice is of rank $\geq 2(n+1)$, which is unchanged by left (resp. right) multiplying $P^{\mathrm{t}}$ (resp. $P$), so it cannot be equal to the corresponding slice of $\hat{\mathtt{B}}$ which is of rank $\leq 2n$. We then arrived at the desired contradiction.

Now consider the action of such $P$ on the $n+1$ to $n+m$ frontal slices. Note that these slices are of the form $\begin{bmatrix} A_i & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. (Recall that the last $m$ slices of $\mathtt{G}$ are all-zero matrices.) Then we have $\begin{bmatrix} P_{1,1}^{\mathrm{t}} & P_{2,1}^{\mathrm{t}} & P_{3,1}^{\mathrm{t}} \\ 0 & P_{2,2}^{\mathrm{t}} & P_{3,2}^{\mathrm{t}} \\ 0 & P_{2,3}^{\mathrm{t}} & P_{3,3}^{\mathrm{t}} \end{bmatrix} \begin{bmatrix} A_i & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{1,1} & 0 & 0 \\ P_{2,1} & P_{2,2} & P_{2,3} \\ P_{3,1} & P_{3,2} & P_{3,3} \end{bmatrix} = \begin{bmatrix} P_{1,1}^{\mathrm{t}} A_i P_{1,1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. Since $P^{\mathrm{t}} \hat{\mathtt{A}}^P P = \hat{\mathtt{B}}$, we have $P^{\mathrm{t}} \hat{\mathtt{A}} P = \hat{\mathtt{B}}^{P^{-1}}$. Observe that for the upper-left $n \times n$ submatrices of the frontal slices of $\hat{\mathtt{B}}$, $P^{-1}$ simply performs a linear combination of $B_i$'s. It follows that every $P_{1,1}^{\mathrm{t}} A_i P_{1,1}$ is in the linear span of $B_i$. Since we assumed $\dim(\langle A_i \rangle) = \dim(\langle B_i \rangle)$, we have that $\mathbf{A}$ and $\mathbf{B}$ are pseudo-isometric. This concludes the proof of Proposition 9. ◀

## References


1 Manindra Agrawal and Nitin Saxena. Automorphisms of finite rings and applications to complexity of problems. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, pages 1–17, 2005. doi:10.1007/978-3-540-31856-9_1.

2 Manindra Agrawal and Nitin Saxena. Equivalence of $\mathbb{F}$-algebras and cubic forms. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, pages 115–126, 2006. doi:10.1007/11672142_8.

3 Jérémy Berthomieu, Jean-Charles Faugère, and Ludovic Perret. Polynomial-time algorithms for quadratic isomorphism of polynomials: The regular case. *J. Complexity*, 31(4):590–616, 2015. doi:10.1016/j.jco.2015.04.001.

4 Charles Bouillaguet. *Etudes d'hypothèses algorithmiques et attaques de primitives cryptographiques*. PhD thesis, PhD thesis, Université Paris-Diderot–École Normale Supérieure, 2011.

5 Charles Bouillaguet, Jean-Charles Faugère, Pierre-Alain Fouque, and Ludovic Perret. Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In *International Workshop on Public Key Cryptography*, pages 473–493. Springer, 2011. doi:10.1007/978-3-642-19379-8_29.

6 Peter A. Brooksbank, Yinan Li, Youming Qiao, and James B. Wilson. Improved algorithms for alternating matrix space isometry: From theory to practice. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.26.

7 Peter A. Brooksbank and James B. Wilson. Computing isometry groups of Hermitian maps. *Trans. Amer. Math. Soc.*, 364:1975–1996, 2012. doi:10.1090/S0002-9947-2011-05388-2.

8 Vyacheslav Futorny, Joshua A. Grochow, and Vladimir V. Sergeichuk. Wildness for tensors. *Lin. Alg. Appl.*, 566:212–244, 2019. doi:10.1016/j.laa.2018.12.022.

9 Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991. doi:10.1145/116825.116852.

10 Joshua A. Grochow and Youming Qiao. Algorithms for group isomorphism via group extensions and cohomology. *SIAM J. Comput.*, 46(4):1153–1216, 2017. Preliminary version in IEEE Conference on Computational Complexity (CCC) 2014 (DOI:10.1109/CCC.2014.19). Also available as arXiv:1309.1776 [cs.DS] and ECCC Technical Report TR13-123. doi:10.1137/15M1009767.

11 Joshua A. Grochow and Youming Qiao. On the complexity of isomorphism problems for tensors, groups, and polynomials I: Tensor Isomorphism-completeness. In *ITCS*, Leibniz International Proceedings in Informatics (LIPIcs), pages 31:1–31:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.31.

**12** Gábor Ivanyos and Youming Qiao. Algorithms based on *-algebras, and their applications to isomorphism of polynomials with one secret, group isomorphism, and polynomial identity testing. *SIAM J. Comput.*, 48(3):926–963, 2019. doi:10.1137/18M1165682.

**13** Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1409–1421, 2011. doi:10.1137/1.9781611973082.108.

**14** Serge Lang. *Algebra*. Number 211 in Graduate Texts in Mathematics. Springer-Verlag, New York, third enlarged edition, 2002.

**15** Yinan Li and Youming Qiao. Linear algebraic analogues of the graph isomorphism problem and the Erdős–Rényi model. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 463–474. IEEE Computer Society, 2017. arXiv:1708.04501, version 2. doi:10.1109/FOCS.2017.49.

**16** Eugene M. Luks. Permutation groups and polynomial-time computation. In *Groups and computation (New Brunswick, NJ, 1991)*, volume 11 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 139–175. Amer. Math. Soc., Providence, RI, 1993.

**17** Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 33–48, 1996. doi:10.1007/3-540-68339-9_4.

**18** Nitin Saxena. *Morphisms of rings and applications to complexity*. PhD thesis, Indian Institute of Technology, Kanpur, May 2006. URL: https://www.cse.iitk.ac.in/users/nitin/papers/thesis.pdf.

**19** H. Weyl. *The classical groups: their invariants and representations*, volume 1. Princeton University Press, 1997.

**20** James B. Wilson. Decomposing $p$-groups via Jordan algebras. *J. Algebra*, 322:2642–2679, 2009. doi:10.1016/j.jalgebra.2009.07.029.

**21** R. Wilson. *The Finite Simple Groups*, volume 251 of *Graduate Texts in Mathematics*. Springer London, 2009.