Efficient Normalized Reduction and Generation of Equivalent Multivariate Binary Polynomials

Arnau Gàmez-Montolio*†, Enric Florit‡, Martin Brain* and Jacob M. Howe*
*City, University of London
{arnau.gamez-montolio, martin.brain, j.m.howe}@city.ac.uk
†Activision Research
‡Universitat de Barcelona
enricflorit@ub.edu

***Abstract*—Polynomials over fixed-width binary numbers (bytes, $\mathbb{Z}/2^w\mathbb{Z}$, bit-vectors, etc.) appear widely in computer science including obfuscation and reverse engineering, program analysis, automated theorem proving, verification, error-correcting codes and cryptography. As some fixed-width binary numbers do not have reciprocals, these polynomials behave differently to those normally studied in mathematics. In particular, polynomial equality is harder to determine; polynomials having different coefficients is not sufficient to show they always compute different values. Determining polynomial equality is a fundamental building block for most symbolic algorithms. For larger widths or multivariate polynomials, checking all inputs is computationally infeasible. This paper presents a study of the mathematical structure of null polynomials (those that evaluate to 0 for all inputs) and uses this to develop efficient algorithms to reduce polynomials to a normalized form. Polynomials in such normalized form are equal if and only if their coefficients are equal. This is a key building block for more mathematically sophisticated approaches to a wide range of fundamental problems.**

I. Introduction

A. Preliminary

A polynomial function over values of w bits (with wrapping) is called a *binary polynomial*. The following definitions are equivalent and can be found spread throughout literature; choosing one usually comes down to the author's background, preferences and underlying motivation for the study.

- Polynomials over $\mathbb{Z}_{2^w} = \mathbb{Z}/2^w\mathbb{Z}$, the ring of integers modulo 2^w.
- Polynomials over fixed-size bit-vectors of w bits.

When talking about binary polynomials in literature, it is implicitly assumed that they refer to *univariate* binary polynomials. Unless otherwise specified, we will also follow this naming convention. From a pure mathematical standpoint, binary polynomials constitute *just* a particular case of the broader research area of polynomial functions over the ring of integers modulo powers of prime numbers [6], [10]. Or, even more generally, over the ring of integers modulo an arbitrary integer [14], [23].

Workshop on Binary Analysis Research (BAR) 2024
1 March 2024, San Diego, CA, USA
ISBN 979-8-9894372-0-7
https://dx.doi.org/10.14722/bar.2024.23014
www.ndss-symposium.org

B. Context

Our main motivation for the study of binary polynomials comes from practical applications to code obfuscation and analysis within the field of software protection [2], [7]. However, the ability to reason about equivalent instances has applications in many other research and engineering areas. Binary polynomial reduction techniques have been used in the formal verification of Register-Transfer Level (RTL) code for processor datapath [8], [22]. Within the field of Satisfiability Modulo Theories (SMT), authors have also pointed out the fundamental need for algebraic techniques to deal with non-linear formulae over $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{Z}_{2^w}$ (a.k.a. the theory of bit-vectors), which have long been regarded as a particularly challenging area. The practical manifestation of these challenges is that many modern SMT solvers struggle to solve problems containing even low-order polynomials in low bit-widths or few variables. Approaches have been proposed [5] but it remains a pressing open problem. The particular properties of binary polynomials have also been used in program analysis [21]. Moreover, the use of binary permutation polynomials (i.e., invertible binary polynomials) over a ring [15] (instead of the usual assumption of polynomials over a finite field [11], [12]) has been crucial in the development of classic cryptographic algorithms like RSA [13], [17] or RC6 [16], as well as in the design of interleavers for turbo codes [18], [19], [24]–[26], error-correcting codes widely deployed in mobile and satellite communication systems.

C. Motivation

As stated before, the initial field that sparked our interest in binary polynomials is code obfuscation and analysis. Several obfuscation mechanisms can benefit from a deeper understanding and efficient generation of (equivalent) binary polynomials. More concretely, binary permutation polynomials are leveraged as *obfuscation primitives* in a number of semantics-preserving transformations that increase the complexity and diversity of the underlying code. In particular, we are not only interested in the direct application of polynomial transformations but also in combining them with state-of-the-art Mixed Boolean-Arithmetic (MBA) obfuscation techniques [9], [27], [28]. We briefly showcase some of these constructions.

1) Data encodings: prevent data values used for arbitrary computations from being revealed during program execution. An initial *encoding* function is first applied to the values to be hidden. An inverse *decoding* function is later applied in combination with (and possibly *blended within*) the operations that manipulate the initial data in order to preserve the original semantics.

Let P, Q be inverse binary permutation polynomials over $\mathbb{Z}_{2^{32}}$ (in other words, over 32-bit variables) defined as

$$P(x) = 1789355803x + 1391591831$$
$$Q(x) = 3537017619x + 624260299.$$

and consider the snippet of C-like code in Listing 1.

```
uint32_t a, b, r;
...
a = key1;
b = key2;
r = foo(a*b);
```

Listing 1. Original code before data encoding.

Now use P, Q as *encoding* and *decoding* functions to protect (obfuscate) the values of key_1, key_2 and their usage: $a = P(key_1)$, $b = P(key_2)$ and $r = foo(Q(a)Q(b))$. We obtain the obfuscated C-like code shown in Listing 2.

```
uint32_t a, b, r;
...
a = 1789355803*key1 + 1391591831;
b = 1789355803*key2 + 1391591831;
r = foo(4112253801*a*b + 1966380049*a + 1966380049*b
    ↪  + 1062639865);
```

Listing 2. Obfuscated code after data encoding.

2) Insertion of identities: increase the syntactic complexity of an algebraic (MBA) expression by *wrapping* it with the identity function generated by the composition of two inverse functions (e.g., binary permutation polynomials).

Let P, Q be inverse binary permutation polynomials over $\mathbb{Z}_{2^8}$ (8-bit variables) defined as

$$P(x) = 8x^2 + 151x + 111$$
$$Q(x) = 200x^2 + 183x + 223.$$

and consider the snippet of C-like code in Listing 3.

```
uint8_t x, y, z;
x = ...;
y = ...;
z = x + y;
```

Listing 3. Original code before identity insertion.

Now *wrap* the expression that computes z with the identity given by the composition of P and Q: $z = P(Q(x + y))$. We obtain the obfuscated C-like code shown in Listing 4.

```
uint8_t x, y, z;
x = ...;
y = ...;
z = 8*(200*(x + y)*(x + y) + 183*(x + y) + 223)
    ↪ *(200*(x + y)*(x + y) + 183*(x + y) + 223) +
    ↪ 151*(200*(x + y)*(x + y) + 183*(x + y) + 223)
    ↪  + 111;
```

Listing 4. Obfuscated code after identity insertion.

3) Opaque constants: conceal (the use of) sensitive constants by replacing them with an algebraic (MBA) expression on an arbitrary number of variables. This expression will always evaluate to the *opaqued* constant during runtime computation, regardless of the concrete values its variables are assigned to.

Working over $\mathbb{Z}_{2^8}$ (8-bit variables), let E be a null MBA expression (i.e., an MBA expression that always evaluates to 0), P, Q be inverse permutation polynomials defined as

$$E(x, y) = x - y + 2(\neg x \wedge y) - (x \oplus y)$$
$$P(x) = 248x^2 + 97x$$
$$Q(x) = 136x^2 + 161x$$

and consider the snippet of C-like code in Listing 5.

```
uint8_t k = 123;
foo(k);
```

Listing 5. Original code before opaque constant.

Now replace the constant $k = 123$ with the expression obtained from expanding $P(E(x, y) + Q(k))$. It is easy to see that the runtime evaluation of such expression will be equal to k regardless of the concrete values of the variables x, y, since $P(E(x, y) + Q(k)) = P(0 + Q(k)) = P(Q(k)) = k$. We obtain the obfuscated C-like code show in Listing 6.

```
uint8_t x, y;
x = ...;
y = ...;
foo(195 + 97*x + 159*y + 194*~(x | ~y) + 159*(x ^ y)
    ↪  + (163 + x + 255*y + 2*~(x | ~y) + 255*(x ^
    ↪ y)) * (232 + 248*x + 8*y + 240*~(x | ~y) +
    ↪ 8*(x ^ y)));
```

Listing 6. Obfuscated code after opaque constant.

It is clear that generating arbitrary equivalent binary polynomials directly enables the possibility of introducing a higher diversity in any obfuscation transformation that relies on binary polynomials as *obfuscation primitives*. Furthermore, being able to produce a normalized reduction of such polynomial transformations is certainly meaningful from a reverse engineering, program analysis and general algebraic manipulation perspective.

D. Approach

Due to the ubiquitous usage of binary permutation polynomials, several authors have focused on the inversion problem. An initial study of the inversion of the quadratic case for turbo codes' interleavers was presented in [20]. An algorithm to generate (a rather small set of) pairs of same-degree inverse binary permutation polynomials was introduced in [28] within the context of MBA obfuscation techniques. There have also been recent research efforts aiming to provide a more general and efficient approach to computing inverse binary permutation polynomials, also influenced by obfuscation purposes [3], [4].

However, we take a step back from permutation polynomials and the inversion problem and address more fundamental questions about binary polynomials in general. While the pure mathematical and theoretical development of polynomial functions over the ring of integers modulo powers of primes or

arbitrary integers [6], [10], [14], [23] is undoubtedly valuable and a necessary starting point, it is at the same time too generic and does not provide explicit enough results (let alone efficient computation algorithms) that can be directly applied to the real-world scenarios described, where binary polynomials are of great, immediate and very concrete relevance.

Thus, influenced and motivated by the many practical applications, we focus on the theoretical (yet concrete) study and development of binary polynomials in particular, with the end goal of obtaining explicit results and efficient algorithms to deal with the problem of generation, normalization and algebraic manipulation of equivalent binary polynomials.

E. Contributions

This paper makes the following contributions:

1) We perform an exhaustive study and theoretical development of the mathematical structure and properties of null binary polynomials (see Section III).
2) We provide a minimal set of generators for the ideal of null binary polynomials (see Theorem III.10).
3) We give a formula to compute the number of equivalent binary polynomials up to a chosen degree (see Proposition III.13).
4) We propose an efficient algorithm to, given a binary polynomial, compute a *normalized* equivalent binary polynomial with the lowest possible degree (see Theorem III.4, Lemma III.11 and Algorithm 1).
5) We propose an efficient algorithm to, given a binary polynomial, arbitrarily compute equivalent binary polynomials up to a chosen degree (see Algorithm 2).
6) We generalize these results (see Section IV) and algorithms (see Section V-C) to multivariate binary polynomials. The procedure we obtain is particularly effective when dealing with sparse polynomials (see Algorithm 3).

II. Notation

Throughout the paper, we work with the standard notions of (unitary and commutative) rings and their ideals, which the reader can find in Chapter 1 of [1]. We recall them briefly. A (commutative) *group* is a triple $(G, +, 0)$, where $+$ is a binary commutative operation on G such that $0 + x = x + 0 = x$ for all $x \in G$, opposites exist ($x + (-x) = 0$), and the standard associativity rules hold. A *ring* $(R, +, 0, \cdot, 1)$ is a tuple such that $(R, +, 0)$ is a group, $\cdot$ is an associative binary operation which distributes with respect to $+$, and 1 is the neutral element for $\cdot$.

The central concept we need is that of an ideal. An *ideal* I of a ring R is a subset $I \subset R$ which is itself a group, and such that $\alpha \cdot x \in I$ for every $\alpha \in R$ and $x \in I$. For example, an ideal I of the integers $(\mathbb{Z}, +, 0, \cdot, 1)$ always consists of the multiples of a single integer $n \in \mathbb{Z}$, called a *generator* of I.

If $g_1, \dots, g_n$ are fixed elements of R, we can form the ideal *generated* by them, denoted

$$\langle g_1, \dots, g_n \rangle,$$

which consists of all sums of the form $\alpha_1 g_1 + \cdots + \alpha_n g_n$ for all possible $\alpha_1, \dots, \alpha_n \in R$. The elements $g_1, \dots, g_n$ are called *generators* of the ideal. There can be more than one set of generators for the same ideal. In general, ideals cannot be generated by a single element (see the ideals $\mathcal{Z}^w$ and $\mathcal{Z}^{w,k}$ below).

More generally, an *R-module* is a group M with an *action* of R. This means having an operation $R \times M \to M$ which for every $\alpha \in R$ and $m \in M$ yields an element $\alpha \cdot m \in M$. Ideals are a particular case of modules. The full definition and some properties of modules can be found in Chapter 2 of [1]. We will mostly use modules when considering subgroups of ideals, $M \subset I$.

Modules can be thought of as generalizations of vector spaces. When an R-module can be described as a cartesian product $M = R^n = R \times \cdots \times R$ (or more generally, an infinite product of copies of R), we say M is *free*. Equivalently, the module has a set of *generators* $g_1, \dots, g_n$ which are linearly independent in the usual sense. We say $g_1, \dots, g_n$ form a *basis* of M. The size of a basis is called the *rank* of M.

Given a ring R, $R[x]$ denotes the ring of univariate polynomials with coefficients in R. Given some positive integer k, the ring $R[x_1, \dots, x_k]$ is the ring of polynomials in k variables and coefficients in R. These rings are in turn free R-modules. An R-basis for $R[x]$ is the *canonical basis*, $\{1, x, x^2, \dots\}$.

We occasionally also mention tensor products, which are introduced in Chapter 2 of [1]. If M and N are R-modules, their tensor product is denoted by $M \otimes_R N$. If we assume that they are *free* so that they have R-bases

$$\{m_1, m_2, \dots\}, \{n_1, n_2, \dots\}$$

then $\{m_i \otimes n_j\}_{i,j}$ is a basis for the free R-module $M \otimes_R N$. This is completely analogous to the case of vector spaces and can be taken as a definition. Notably, this applies to polynomial rings, which are free (of countable rank). The example relevant to us is the isomorphism

$$R[x_1, \dots, x_k] \simeq R[x_1] \otimes_R \cdots \otimes_R R[x_k].$$

This is made explicit by saying that

$$\{x_1^{r_1} x_2^{r_2} \cdots x_k^{r_k}\}_{r_1, \dots, r_k \geq 0}$$

is an R-basis for $R[x_1, \dots, x_k]$, while

$$\{x_1^{r_1} \otimes x_2^{r_2} \otimes \cdots \otimes x_k^{r_k}\}_{r_1, \dots, r_k \geq 0}$$

is an R-basis for $R[x_1] \otimes_R \cdots \otimes_R R[x_k]$.

If t is any real number, $[t]$ denotes its *floor*, which is the largest integer m such that $m \leq t$. For a nonzero integer n, we write $\nu_2(n)$ for the largest r such that 2^r divides n. We also write $\nu_2(0) = \infty$. The function $\nu_2 : \mathbb{Z} \to \mathbb{N} \cup \{\infty\}$ is called the *2-adic valuation* and satisfies the usual properties:

1) $\nu_2(ab) = \nu_2(a) + \nu_2(b)$,
2) $\nu_2(a + b) \geq min(\nu_2(a), \nu_2(b))$.

It will be relevant to know the 2-adic valuation of a factorial number, $n!$, which is seen to be

$$\nu_2(n!) = \sum_{r=1}^{\infty} \left[\frac{n}{2^r} \right].$$

Note that this is in fact a sum of finitely many nonzero terms, since there is always some $r \geq 1$ such that $\frac{n}{2^r} < 1$.

For every nonnegative integer n, we define its *superfactorial* by the formula

$$\mathrm{sf}(n) = \begin{cases} 1, & n = 0 \\ 1! \cdot 2! \cdots n!, & n > 0. \end{cases}$$

III. Null polynomials modulo 2^w

Let R be a ring. Every polynomial $P(x) \in R[x]$ defines by evaluation a function $R \to R$. If we take another $Q(x) \in R[x]$ defining the same function as P, then $P - Q$ is a polynomial which vanishes identically on all of R (that is, the function given by $P - Q$ is constant equal to 0). This motivates the following.

Definition III.1. *We say a polynomial $P(x) \in R[x]$ is a* null polynomial *if $P(a) = 0$ for all $a \in R$.*

The set Z of all null polynomials for R is easily seen to form an ideal of $R[x]$.

Remark. *We say an arbitrary function $f : R \to R$ is polynomial if there is some polynomial $P(x) \in R[x]$ such that $f(a) = P(a)$ for all $a \in R$. The set of polynomial functions forms a ring under addition and multiplication. It is easy to show that this ring is isomorphic to the quotient ring $R[x]/Z$. In other words, "two polynomials defining the same function" is an equivalence relation.*

If R is a subring of $\mathbb{C}$, it is well-known that there are no (nonzero) null polynomials in $R[x]$. The situation is different if R is a ring with finitely many elements. From now on we fix some integer $w \geq 1$ and let $R := \mathbb{Z}_{2^w} = \mathbb{Z}/2^w\mathbb{Z}$.

We denote the ideal of null polynomials in $\mathbb{Z}_{2^w}[x]$ by $\mathcal{Z}^w$. For every positive integer d, $\mathbb{Z}_{2^w}[x]_d$ denotes the set of polynomials of degree at most d. It is convenient for us to define $\mathcal{Z}_d^w := \mathcal{Z}^w \cap \mathbb{Z}_{2^w}[x]_d$, the subgroup of polynomials $P \in \mathcal{Z}^w$ of degree at most d. Then we have

$$\mathcal{Z}_d^w \subseteq \mathcal{Z}_{d+1}^w \quad \text{and} \quad \mathcal{Z}^w = \bigcup_{d=0}^{\infty} \mathcal{Z}_d^w.$$

We will later see that $\mathcal{Z}^w$ is not a principal ideal, i.e. $\mathcal{Z}^w$ is not generated by a single element. By Hilbert's basis theorem, it has a finite number of generators as a $\mathbb{Z}_{2^w}[x]$-ideal. In particular, we will provide an explicit minimal set of $\mathbb{Z}_{2^w}[x]$-generators for $\mathcal{Z}^w$.

Let $P(x) \in \mathbb{Z}_{2^w}[x]$ be a polynomial of degree d and fix an integer $\delta \geq d$. It follows from the previous remark that the number of polynomials of degree at most δ defining the same function as P equals $\#\mathcal{Z}_\delta^w$. Hence, we just need to compute $\#\mathcal{Z}_\delta^w$ to know the number of equivalent polynomial functions over $\mathbb{Z}_{2^w}[x]$ up to degree δ. We will make this computation more explicit at the end of this section.

Let us show some first properties of $\mathcal{Z}^w$.

Lemma III.2. *The following statements hold:*

1) *$\#\mathcal{Z}_d^w$ is a power of 2.*
2) *Every $h \in \mathcal{Z}^w$ has constant term equal to 0.*
3) *$\mathcal{Z}_0^w = \mathcal{Z}_1^w = \{0\}$.*

Proof: To see 1), we note that $\mathcal{Z}_d^w$ is a subgroup of $\mathbb{Z}_{2^w}[x]_d$, which has order $2^{w(d+1)}$. To see 2), let $h(x) \in \mathcal{Z}^w$, which we write as

$$h(x) = a_d x^d + \cdots + a_1 x + a_0.$$

By definition we have $h(a) \equiv 0 \pmod{2^w}$ for all $a \in \mathbb{Z}_{2^w}$. In particular, this must be the case for $h(0)$, but then $h(0) \equiv a_0 \equiv 0 \pmod{2^w}$. This also shows $\mathcal{Z}_0^w = \{0\}$, which is the first part of 3).

To see $\mathcal{Z}_1^w = \{0\}$, let $h(x) \in \mathcal{Z}_1^w$. We already know that $h(x) = a_1 x$ for some $a_1 \in \mathbb{Z}_{2^w}$. But again by definition we must have $h(1) \equiv 0 \pmod{2^w}$, showing $a_1 \equiv 0 \pmod{2^w}$. ∎

The polynomials $1, x, x^2, \ldots$ form the *canonical basis* of $\mathbb{Z}_{2^w}[x]$. For consistency and convenience, we use the notation in [14] to write polynomials in terms of factorial powers. Let $x^{(0)} = 1$ and $x^{(1)} = x$. For every $j \geq 2$ we define the polynomial

$$x^{(j)} = x(x-1)\cdots(x-j+1),$$

which has degree j. The polynomials $x^{(0)}, x^{(1)}, \ldots, x^{(d)}$ form a $\mathbb{Z}_{2^w}$-basis for $\mathbb{Z}_{2^w}[x]_d$, called the *factorial basis*. To check this, we write their coefficients in matrix form, placing each polynomial in a row starting with the independent term, and we obtain a triangular matrix with ones in the diagonal. Hence the determinant equals 1, and this is indeed a basis.

For $j \geq 0$, we let

$$c_j := 2^{\max(w - \nu_2(j!), 0)} \text{ and } G_j(x) := c_j x^{(j)}.$$

For example, if $w = 8$, then $G_2(x) := 2^7 x(x-1) = 128x^2 - 128x$. This is clearly a null polynomial since either x or $x-1$ will be even. Thus, we can always write $x(x-1) = 2x'$ for some x', and $G_2(x) = 2^8 x' \equiv 0 \pmod{2^8}$ trivially.

Remark. *In general, we think of the polynomials G_j in $\mathbb{Z}_{2^w}[x]$, but sometimes we will consider c_j as a proper integer.*

Lemma III.3. *For $j \geq 0$, $G_j(x)$ is a null polynomial of degree j, i.e. $G_j \in \mathcal{Z}_j^w$.*

Proof: For all integers $0 \leq k < j$ we have $x^{(j)}(k) = 0$. Let $k \geq j$. The equality $x^{(j)}(k) = 0$ then follows from the fact that the product of j consecutive integers is divisible by $j!$. Indeed, the binomial coefficient $\binom{k}{j}$ can be expressed as

$$\binom{k}{j} = \frac{k!}{(k-j)!j!} = \frac{k(k-1)\cdots(k-j+1)}{j!} = \frac{x^{(j)}(k)}{j!}$$

and is always an integer. This implies that $\nu_2(x^{(j)}(k)) \geq \nu_2(j!)$ for every integer $k \geq j$, and therefore $G_j(k) \equiv 0 \pmod{2^w}$ for all integers k. ∎

The following result characterizes all polynomials in $\mathcal{Z}^w$.

Theorem III.4. *The group $\mathcal{Z}_d^w$ is generated as a (non-free) $\mathbb{Z}_{2^w}$-module by $G_2, G_3, \ldots, G_d$. More concretely, every $H(x) \in \mathcal{Z}_d^w$ can be uniquely written as*

$$H(x) = \sum_{j=2}^{d} h_j G_j,$$

where $h_j \in \mathbb{Z}_{2^w}$ is the residue class of an integer $0 \le \tilde{h}_j \le 2^{\min(\nu_2(j!),w)} - 1$.

Proof: We follow the proof of [14, Theorem 2.1]. By Lemma III.3, any $H(x)$ as in the statement is a null polynomial. Conversely, let

$$H(x) = \sum_{j=0}^{d} h_j x^{(j)}$$

be any polynomial in $\mathcal{Z}_d^w$. The difference operator

$$\Delta H(x) := H(x+1) - H(x)$$

is linear and satisfies

$$\Delta^j x^{(i)} = i(i-1)\cdots(i-j+1)x^{(i-j)}.$$

Hence we have $\Delta^j H(0) \equiv h_j j! \equiv 0 \pmod{2^w}$, so that

$$h_j \equiv 0 \pmod{2^{\max(w-\nu_2(j!),0)}}.$$

If $\tilde{h}_j$ is any integer representing this residue class, this is equivalent to the divisibility condition $c_j \mid \tilde{h}_j$. This proves any $H(x) \in \mathcal{Z}_d^w$ has the form in the statement. The uniqueness comes from standard results on congruences, plus the conditions $h_j \equiv 0 \pmod{2^{\max(w-\nu_2(j!),0)}}$ and $h_j \in \mathbb{Z}_{2^w}$. In particular, we have $h_0 \equiv h_1 \equiv 0 \pmod{2^w}$. ∎

Definition III.5. *We denote by d_w the smallest positive integer such that*

$$\nu_2(d_w!) \ge w.$$

Recall that a polynomial $P(x) \in \mathbb{Z}_{2^w}[x]$ is said to be *monic* if its leading coefficient is equal to 1.

Lemma III.6. *Let t be the smallest integer for which $\mathcal{Z}_t^w$ contains a monic polynomial. Then $t = d_w$.*

Proof: The leading coefficient of G_{d_w} is $c_{d_w} = 2^{\max(w-\nu_2(d_w!),0)} = 2^0 = 1$. Therefore $\mathcal{Z}_{d_w}^w$ contains a monic polynomial, and $t \le d_w$.

Suppose by way of contradiction that $t < d_w$. Let $H(x) \in \mathcal{Z}_t^w$ be an arbitrary polynomial. By Theorem III.4, $H(x)$ can be written as

$$H(x) = \sum_{j=2}^{t} h_j G_j,$$

with some condition on each h_j. Let $k \le t$ be the largest index such that $h_k \neq 0$. Then the leading coefficient of H is a multiple of that of G_k, which equals $c_k = 2^{\max(w-\nu_2(k!),0)}$. By assumption, $k \le t < d_w$, and by definition of d_w we have $w - \nu_2(k!) > 0$. Thus, H has leading coefficient (a multiple of) $c_k = 2^{w-\nu_2(k!)} \neq 1$.

Since H was arbitrary, we have reached a contradiction, because $\mathcal{Z}_t^w$ does not contain any monic polynomial. It follows that $t = d_w$. ∎

Definition III.7. *If $P(x) \in \mathcal{Z}_{d_w}^w$ is monic, we say P is a* least-degree monic null polynomial.

Remark. *The* least-degree monic null polynomial *is not unique, which can be seen by adding any nonzero null polynomial of strictly smaller degree.*

For practical applications, it will be convenient to have a straightforward formula that computes the value of d_{2^k}.

Lemma III.8. *If $w = 2^k$, then $d_w = w+2$. As a consequence, for general values of w we have the inequalities*

$$2^{[\log_2(w)]} + 2 \le d_w \le 2^{[\log_2(w)]+1} + 2.$$

In other words, $d_w = O(w)$.

Proof: We use the formula $\nu_2(n!) = \sum_{r=1}^{\infty} \left[\frac{n}{2^r}\right]$. By direct computation, we have

$$\begin{aligned}\nu_2(w!) = \nu_2(2^k!) &= \sum_{r=1}^{\infty} \left[\frac{2^k}{2^r}\right] = \sum_{r=1}^{k} 2^{k-r} \\ &= \sum_{s=0}^{k-1} 2^s = 2^k - 1 = w - 1.\end{aligned}$$

It follows that $d_w > w$. In fact, because $w+1$ is odd, we have $\nu_2((w+1)!) = \nu_2(w!)$, so $d_w \ge w+2$. And because $w+2$ is even, we have

$$\nu_2((w+2)!) = \nu_2(w+2) + \nu_2(w!) = \nu_2(w+2) + w - 1 \ge w.$$

We conclude $d_w = w + 2$.

For the inequalities, we let $k = [\log_2 w]$. Then $2^k \le w \le 2^{k+1}$. The map $w \mapsto d_w$ is nondecreasing, so we obtain $d_{2^k} \le d_w \le d_{2^{k+1}}$. But we have already shown that $d_{2^k} = 2^k + 2$ and $d_{2^{k+1}} = 2^{k+1} + 2$. ∎

Next we relate the different submodules $\mathcal{Z}_d^w$ of $\mathcal{Z}^w$. We introduce the following notation:

$$(1+x)\mathcal{Z}_d^w = \{P(x) + xQ(x) \mid P, Q \in \mathcal{Z}_d^w\}.$$

Proposition III.9. *For every $d \ge 0$, we have the inclusion $(1+x)\mathcal{Z}_d^w \subseteq \mathcal{Z}_{d+1}^w$. Moreover, this inclusion is an equality whenever d is even.*

Proof: To see the inclusion, let $P(x), Q(x) \in \mathcal{Z}_d^w$ be null polynomials of degree at most d. Then, $P + xQ$ is also a null polynomial of degree at most $d+1$. It is then clear that $P + xQ \in \mathcal{Z}_{d+1}^w$.

Suppose now that d is even. We know that $\mathcal{Z}_d^w$ (respectively, $\mathcal{Z}_{d+1}^w$) is generated by $G_2, \ldots, G_d$ (resp. $G_2, \ldots, G_d, G_{d+1}$). We observe that G_d and G_{d+1} have the same leading coefficient. Indeed, G_d has leading coefficient $2^{\max(w-\nu_2(d!),0)}$, while G_{d+1} has leading coefficient $2^{\max(w-\nu_2((d+1)!),0)}$. But now

$$\nu_2((d+1)!) = \nu_2((d+1)\cdot d!) = \nu_2(d+1) + \nu_2(d!) = \nu_2(d!),$$

since $d+1$ is odd and therefore $\nu_2(d+1) = 0$. Hence we can write $G_{d+1} = G_d \cdot (x - d) \in (1+x)\mathcal{Z}_d^w$, concluding $(1+x)\mathcal{Z}_d^w = \mathcal{Z}_{d+1}^w$. ∎

With the above inclusion result, we are ready to find generators of $\mathcal{Z}^w$ as a $\mathbb{Z}_{2^w}[x]$-ideal.

Theorem III.10. *If $d \ge d_w$, then $(1+x)\mathcal{Z}_d^w = \mathcal{Z}_{d+1}^w$. In particular, $\mathcal{Z}^w$ is generated as a $\mathbb{Z}_{2^w}[x]$-ideal by the set $\{G_2, G_4, \ldots, G_{d_w}\}$. Moreover, this is a minimal generating set for $\mathcal{Z}^w$, in the sense that $G_j \notin \langle G_2, \ldots, G_{j-2}, G_{j+2}, \ldots, G_{d_w}\rangle$, for every $j = 2, 4, \ldots, d_w$.*

Proof: Let $d \geq d_w$ and $P(x) \in \mathcal{Z}^w_{d+1}$. Let $Q(x) \in \mathcal{Z}^w_d$ be a null polynomial with leading coefficient 1. Writing P as

$$P(x) = a_{d+1}x^{d+1} + \cdots + a_1 x + a_0,$$

we see that the polynomial $P - a_{d+1}xQ$ has degree d. Therefore $P - a_{d+1}xQ \in \mathcal{Z}^w_d$, and so there is some $h(x) \in \mathcal{Z}^w_d$ such that $P = h + a_{d+1}xQ$. The equality is now clear since $h + a_{d+1}xQ \in (1+x)\mathcal{Z}^w_d$.

From the equality $(1+x)\mathcal{Z}^w_d = \mathcal{Z}^w_{d+1}$, we can obtain a generating set for $\mathcal{Z}^w$ from a generating set for $\mathcal{Z}^w_{d_w}$, which by Theorem III.4 can be taken to be $G_2, G_3, G_4, \ldots, G_{d_w}$. But since we are considering $\mathbb{Z}_{2^w}[x]$-multiples, we can obviate the odd-degree polynomials by using Proposition III.9.

To see that $G_2, G_4, \ldots, G_{d_w}$ is a minimal generating set, it is enough to observe that $G_{j+2} \notin \langle G_2, \ldots, G_j \rangle$ for all $j = 2, 4, \ldots, d_w - 2$. Otherwise, the leading coefficient $c_{j+2} = 2^{\max(w - \nu_2((j+2)!), 0)}$ of G_{j+2} would have to be a multiple of c_j. But this is never the case for $j \leq d_w$, because $\nu_2(j!)$ is strictly smaller that $\nu_2((j+2)!)$. ∎

To end this section, we go back to equivalent polynomials.

Lemma III.11. *Let $f : \mathbb{Z}_{2^w} \to \mathbb{Z}_{2^w}$ be any polynomial function. There exists a polynomial $Q(x) \in \mathbb{Z}_{2^w}[x]$ of degree $\deg(Q) < d_w$ defining the same function as f.*

Proof: Let $P(x) \in \mathbb{Z}_{2^w}[x]$ be a polynomial representing the function f. Because G_{d_w} is a monic polynomial, we can perform Euclidean division of P by G_{d_w}. This means that there are polynomials $b(x)$ and $Q(x)$ such that

$$P = G_{d_w} b + Q,$$

with $\deg(Q) < \deg(G_{d_w}) = d_w$. Since G_{d_w} is a null polynomial, we indeed have $f(a) = P(a) \equiv Q(a) \pmod{2^w}$ for every $a \in \mathbb{Z}_{2^w}$. ∎

Definition III.12. *Given a polynomial $P(x) \in \mathbb{Z}_{2^w}[x]$, we let the reduced degree of P be the degree of the smallest polynomial Q defining the same function as P. We denote it by* $\operatorname{rdeg}(P)$.

As we hinted above, we can now compute the number of polynomials equivalent to a fixed one up to a certain degree.

Proposition III.13. *Let $P(x) \in \mathbb{Z}_{2^w}[x]$ be a polynomial and fix some integer $\delta \geq \operatorname{rdeg}(P)$. The number of polynomials in $\mathbb{Z}_{2^w}[x]_\delta$ defining the same function as $P(x)$ is*

$$\#\mathcal{Z}^w_\delta = \begin{cases} 2^{v_2(\mathrm{sf}(\delta))}, & \delta < d_w \\ 2^{v_2(\mathrm{sf}(d_w - 1))} \cdot 2^{w(\delta - d_w + 1)}, & \delta \geq d_w \end{cases}$$

where $\mathrm{sf}(n)$ is the n^{th} superfactorial.

Proof: By Theorem III.4, every $H(x) \in \mathcal{Z}^w_\delta$ is written uniquely as

$$H(x) = \sum_{j=2}^{\delta} h_j G_j,$$

where $h_j \in \mathbb{Z}_{2^w}$ is the residue class of an integer $0 \leq \tilde{h}_j \leq 2^{\min(\nu_2(j!), w)} - 1$. The number of possible tuples $(h_2, h_3, \ldots, h_\delta)$, and therefore the number of elements in $\mathcal{Z}^w_\delta$, is then equal to $\prod_{j=0}^{\delta} 2^{\min(\nu_2(j!), w)}$. By definition of d_w, if $\delta < d_w$ then

$$\begin{aligned} \prod_{j=2}^{\delta} 2^{\min(\nu_2(j!), w)} &= \prod_{j=2}^{\delta} 2^{\nu_2(j!)} \\ &= 2^{\sum_{j=2}^{\delta} \nu_2(j!)} \\ &= 2^{\nu_2(\prod_{j=2}^{\delta} j!)} \\ &= 2^{v_2(\mathrm{sf}(\delta))}. \end{aligned}$$

The equality $\nu_2(\prod_{j=2}^{\delta} j!) = \mathrm{sf}(\delta)$ is clear, since $\nu_2(0!) = \nu_2(1!) = 0$. If $\delta \geq d_w$ instead, we have

$$\begin{aligned} \prod_{j=0}^{\delta} 2^{\min(\nu_2(j!), w)} &= \prod_{j=0}^{d_w - 1} 2^{\nu_2(j!)} \cdot \prod_{j=d_w}^{\delta} 2^w \\ &= 2^{\nu_2(\mathrm{sf}(d_w - 1))} \cdot 2^{w(\delta - d_w + 1)}. \end{aligned}$$

∎

IV. The multivariate case

We now generalize the results of the previous section to null polynomials in the ring $\mathbb{Z}_{2^w}[x_1, \ldots, x_k]$ of polynomials in k variables. As before, this can be applied to the study of polynomial functions $\mathbb{Z}^k_{2^w} \to \mathbb{Z}_{2^w}$.

To deal with degrees in $\mathbb{Z}_{2^w}[x_1, \ldots, x_k]$ we let $\mathbb{N}$ be the natural numbers starting from 0, and consider multi-degrees $\mathbf{d} \in \mathbb{N}^k$. Given $\mathbf{d} = (d_1, \ldots, d_k)$ and $\mathbf{d}' = (d'_1, \ldots, d'_k) \in \mathbb{N}^k$, we write $\mathbf{d} \leq \mathbf{d}'$ if $d_i \leq d'_i$ for every $i = 1, \ldots, k$ [1]. If $\mathbf{d} \leq \mathbf{d}'$ but $\mathbf{d} \neq \mathbf{d}'$, we write $\mathbf{d} \lneq \mathbf{d}'$. We denote the zero vector by $\mathbf{0}$. We let $\mathbf{e}_i$ be the vector in $\mathbb{N}^k$ which has a 1 in the i^{th} component and zeros in every other component. The notation $x^{\mathbf{d}}$ denotes the monomial $\prod_{i=1}^{k} x_i^{d_i}$.

We denote the ideal of null polynomials in $\mathbb{Z}_{2^w}[x_1, \ldots, x_k]$ by $\mathcal{Z}^{w,k}$. As in the univariate case, $\mathcal{Z}^{w,k}$ is not a principal ideal. Given $\mathbf{d} \in \mathbb{N}^k$, we let $\mathcal{Z}^{w,k}_{\mathbf{d}}$ be the subgroup of polynomials $P \in \mathcal{Z}^{w,k}$ where the maximum degree of the variable x_i in a given term is at most d_i. Then we have

$$\begin{aligned} \mathcal{Z}^{w,k}_{\mathbf{d}} &\subseteq \mathcal{Z}^{w,k}_{\mathbf{d}+\mathbf{e}_i} \\ \mathcal{Z}^{w,k} &= \bigcup_{\mathbf{d} \in \mathbb{N}^k} \mathcal{Z}^{w,k}_{\mathbf{d}} \end{aligned}$$

As in the univariate case, we want to use a *factorial basis* for $\mathbb{Z}_{2^w}[x_1, \ldots, x_k]$. Given a multi-degree $\mathbf{d} = (d_1, \ldots, d_k) \in \mathbb{N}^k$, we let

$$x^{(\mathbf{d})} := \prod_{i=1}^{k} x_i^{(d_i)}$$

(this notation should not be confused with the monomial $x^{\mathbf{d}}$). The set $\{x^{(\mathbf{d}')}\}_{\mathbf{d}' \leq \mathbf{d}}$ is a $\mathbb{Z}_{2^w}$-basis of $\mathbb{Z}_{2^w}[x_1, \ldots, x_k]_{\mathbf{d}}$, the $\mathbb{Z}_{2^w}$-vector space of polynomials whose monomials have multi-degree at most $\mathbf{d}$. For every $\mathbf{j} \in \mathbb{N}^k$, we let

$$\begin{aligned} c_{\mathbf{j}} &:= 2^{\max(w - \sum_{i=1}^{k} \nu_2(j_i!), 0)} \\ G_{\mathbf{j}}(x_1, \ldots, x_k) &:= c_{\mathbf{j}} x_1^{(j_1)} \cdots x_k^{(j_k)} = c_{\mathbf{j}} x^{(\mathbf{j})}. \end{aligned}$$

[1] This is not standard notation. We remark that this is only a partial order, and in particular it is not a monomial order.

For example, if $w = 8$ and we consider two variables x_1, x_2, then $G_{(2,2)}(x_1, x_2) := 2^6 x_1(x_1 - 1)x_2(x_2 - 1) = 64x_1^2x_2^2 - 64x_1x_2^2 - 64x_1^2x_2 + 64x_1x_2$. This is clearly a null polynomial since either x_1 or $x_1 - 1$ will be even, and either x_2 or $x_2 - 1$ will be even as well. Thus, we can always write $x_1(x_1 - 1)x_2(x_2 - 1) = 2^2x_1'x_2'$ for some x_1', x_2', and $G_{(2,2)}(x_1, x_2) = 2^8x_1'x_2' \equiv 0 \pmod{2^8}$ trivially.

Remark. *The fact that $\{x^{(\mathbf{d})}\}_{\mathbf{d}\in\mathbb{N}^k}$ is a basis is equivalent to the isomorphism of rings*

$$\mathbb{Z}_{2^w}[x_1, \ldots, x_k] \simeq \mathbb{Z}_{2^w}[x_1] \otimes_{\mathbb{Z}_{2^w}} \cdots \otimes_{\mathbb{Z}_{2^w}} \mathbb{Z}_{2^w}[x_k],$$

where $\otimes_{\mathbb{Z}_{2^w}}$ denotes the tensor product of $\mathbb{Z}_{2^w}$-algebras. This observation will be useful to our algorithmic approach to reduce the multivariate case to the univariate setting.

Theorem IV.1 (Generalization of Theorem III.4)**.** *The group $\mathcal{Z}_{\mathbf{d}}^{w,k}$ is generated as a $\mathbb{Z}_{2^w}$-module by the set of polynomials $\{G_{\mathbf{j}}\}_{\mathbf{j}\le\mathbf{d}}$. More concretely, every $H(x_1, \ldots, x_k) \in \mathcal{Z}_{\mathbf{d}}^{w,k}$ can be uniquely written as*

$$H(x_1, \ldots, x_k) = \sum_{\substack{\mathbf{j}\in\mathbb{N}^k \\ \mathbf{j}\le\mathbf{d}}} h_{\mathbf{j}} G_{\mathbf{d}},$$

where $h_{\mathbf{j}} \in \mathbb{Z}_{2^w}$ is the residue class of an integer $0 \le \tilde{h}_{\mathbf{j}} \le 2^{\sum_{i=1}^k \nu_2(j_i!)} - 1$.

Proof: Adapting the proof of Lemma III.3, one sees that every $G_{\mathbf{j}}$ is a null polynomial. Therefore, every $H(x)$ as in the statement belongs to $\mathcal{Z}_{\mathbf{d}}^{w,k}$. Conversely, let

$$H(x) = \sum_{\mathbf{j}\le\mathbf{d}} h_{\mathbf{j}} x^{(\mathbf{j})}$$

be some null polynomial in $\mathcal{Z}_{\mathbf{d}}^{w,k}$. We define the i^{th} partial difference operator Δ_i by

$$\Delta_i H(x_1, \ldots, x_k) := H(x_1, \ldots, x_i + 1, \ldots, x_k) - H(x_1, \ldots, x_i, \ldots, x_k).$$

A short computation shows that $\Delta_i x^{(\mathbf{j})} = i x^{(\mathbf{j}-\mathbf{e}_i)}$, and therefore

$$\Delta_i^t x^{(\mathbf{j})} = j_i(j_i - 1)\cdots(j_i - t + 1)x^{(\mathbf{j}-t\mathbf{e}_i)}.$$

It follows that

$$\Delta_1^{j_1}\Delta_2^{j_2}\cdots\Delta_k^{j_k} H(0) \equiv h_{(j_1,\ldots,j_k)} j_1!\cdots j_k! \equiv 0 \pmod{2^w},$$

and therefore $h_{(j_1,\ldots,j_k)} \equiv 0 \pmod{2^{\max(w-\sum\nu_2(j_i!),0)}}$. The rest of the argument is identical to the proof of Theorem III.4. ∎

Proposition IV.2. *For every $\mathbf{d} = (d_1, \ldots, d_k) \in \mathbb{N}^k$ and every index $i = 1, \ldots, k$, we have the inclusion $(1 + x_i)\mathcal{Z}_{\mathbf{d}}^{w,k} \subseteq \mathcal{Z}_{\mathbf{d}+\mathbf{e}_i}^{w,k}$. Moreover, this inclusion is an equality whenever d_i is even.*

Proof: The proof is a generalization of Proposition III.9. To see the inclusion, let $P(x), Q(x) \in \mathcal{Z}_{\mathbf{d}}^{w,k}$ be null polynomials where the exponent of each variable x_i is at most d_i. If we fix an index i, then $P + x_iQ$ is also a null polynomial in which the degree of the variable x_i is at most $d_i + 1$. It is then clear that $P + x_iQ \in \mathcal{Z}_{\mathbf{d}+\mathbf{e}_i}^{w,k}$.

Suppose now that d_i is even. We know that $\mathcal{Z}_{\mathbf{d}}^{w,k}$ (respectively, $\mathcal{Z}_{\mathbf{d}+\mathbf{e}_i}^{w,k}$) is generated by $\{G_{\mathbf{j}}\}_{\mathbf{j}\le\mathbf{d}}$ (resp. $\{G_{\mathbf{j}}\}_{\mathbf{j}\le\mathbf{d}+\mathbf{e}_i}$).

We observe that G_{d_i} and G_{d_i+1} have the same leading coefficient, as $\nu_2((d_i + 1)!) = \nu_2((d_i)!)$. Hence we can write $G_{\mathbf{d}+\mathbf{e}_i} = G_{\mathbf{d}} \cdot (x_i - d_i) \in (1 + x_i)\mathcal{Z}_{\mathbf{d}}^{w,k}$, concluding $(1 + x_i)\mathcal{Z}_{\mathbf{d}}^{w,k} = \mathcal{Z}_{\mathbf{d}+\mathbf{e}_i}^{w,k}$. ∎

It is now convenient for us to introduce the following notation before proceeding further. Consider the sets of indices

$$\mathcal{J} = \{(j_1, \ldots, j_k) \in 2\mathbb{N}^k \mid 0 < \sum_{i=1}^k \nu_2(j_i!) < w\},$$

$$\mathcal{J}^m = \left\{ \mathbf{j} + 2\mathbf{e}_i \;\middle|\; \begin{array}{l} \mathbf{j} + 2\mathbf{e}_i \notin \mathcal{J}; \\ \mathbf{j} + 2\mathbf{e}_i - 2\mathbf{e}_{i'} \in \mathcal{J}\ \forall i' = 1, \ldots, k \end{array} \right\}.$$

With this, consider the sets of polynomials

$$\mathcal{B}_1 = \{G_{\mathbf{j}} \mid \mathbf{j} \in \mathcal{J}\},$$
$$\mathcal{B}_2 = \{G_{\mathbf{j}} \mid \mathbf{j} \in \mathcal{J}^m\}.$$

The polynomials in $\mathcal{B}_1$ are set so that they are analogs of the polynomials $\{G_j\}_{j=2,\ldots,d_w-2}$ in the univariate case. As for $\mathcal{B}_2$, the definition ensures that the sum of valuations for every index goes "just above" w.

The following result generalizes Theorem III.10.

Theorem IV.3. *Let $\mathbf{d} = (d_1, \ldots, d_k) \in \mathbb{N}^k$. If $\sum_{i=1}^k \nu_2(d_i!) \ge w$, then*

$$(1 + x_i)\mathcal{Z}_{\mathbf{d}}^{w,k} = \mathcal{Z}_{\mathbf{d}+\mathbf{e}_i}^{w,k}$$

for every $i = 1, \ldots, k$. Moreover, $\mathcal{Z}^{w,k}$ is generated as a $\mathbb{Z}_{2^w}[x_1, \ldots, x_k]$-ideal by the minimal set $\mathcal{B} := \mathcal{B}_1 \cup \mathcal{B}_2$.

Proof: See Appendix A. ∎

V. Algorithms

A. Univariate algorithms

We now describe algorithms to work with polynomials defining equivalent functions $\mathbb{Z}_{2^w} \to \mathbb{Z}_{2^w}$. As Theorem III.4 shows, the natural basis to obtain equivalent polynomials is the factorial basis of $\mathbb{Z}_{2^w}[x]$.

Since we need to work using a finite basis, we let d be a fixed degree, and let F be the matrix describing the change of basis from $\{1, x, \ldots, x^d\}$ to $\{1, x^{(1)}, \ldots, x^{(d)}\}$. We let $C = F^{-1}$. The rows of the matrix C are obtained by expanding each polynomial $x^{(i)}$ of the factorial basis and reading their coefficients. These can also be generated recursively using the identity

$$x^{(i+1)} = (x - i) \cdot x^{(i)}.$$

The matrices C and F are triangular, an important property towards implementation efficiency. We use the notations $F[r, c]$ and $C[r, c]$ to indicate taking the first r rows and c columns of the corresponding matrix. For our algorithms, we assume that the necessary matrices have been precomputed.

Definition V.1. *Let $P(x) \in \mathbb{Z}_{2^w}[x]$ be any polynomial. The* normalization *of P is the unique polynomial $\tilde{P}$ such that*

- $\deg(\tilde{P}) = \operatorname{rdeg}(P) < d_w$,

- $P(a) \equiv \tilde{P}(a) \pmod{2^w}$ *for every* $a \in \mathbb{Z}_{2^w}$,
- $\tilde{P} = \sum_{j=0}^{\operatorname{rdeg}(P)} a_j x^{(j)}$ *with* $0 \le a_j \le 2^{w-\nu_2(j!)} - 1$.

The normalized polynomial $\tilde{P}$ exists by Lemma III.11 and is unique by Theorem III.4.

Given a polynomial

$$P(x) = a_0 + a_1 x + \cdots + a_d x^d \in \mathbb{Z}_{2^w}[x]$$

we let $v_P = (a_0, a_1, \ldots, a_d)$ be the vector whose components are the coefficients of P. Conversely, we may interpret a vector $v \in \mathbb{Z}_{2^w}^{d+1}$ as a degree-d polynomial. We index the entries of v as $v[j]$ for $j = 0, 1, \ldots, d$.

Recall that we have defined $c_j = 2^{w-\nu_2(j!)}$, for $0 \le j \le d_w - 1$.

Algorithm 1 Reduction of a univariate polynomial

```
function NormalizePolynomial(v_P)
    ℓ ← min(deg(P) + 1, d_w)
    u ← F[ℓ, deg(P) + 1] · v_P
    for j ← 2, ..., ℓ do
        u[j] ← u[j] % c_j
    end for
    ṽ ← C[ℓ, ℓ] · u
    return ṽ
end function
```

Proposition V.2. *Algorithm 1 computes the normalization* $\tilde{P}(x)$ *of* $P(x)$ *correctly in* $O(w \cdot \deg(P))$ *operations in* $\mathbb{Z}_{2^w}$.

Proof: Write the polynomial $P(x)$ in the factorial basis, so that

$$P(x) = \sum_{i=0}^{\deg(P)} a_i x^{(i)}.$$

We observe that the polynomial

$$Q(x) = \sum_{i=0}^{\ell-1} a_i x^{(i)},$$

where $\ell = \min(\deg(P) + 1, d_w)$, defines the same function $\mathbb{Z}_{2^w} \to \mathbb{Z}_{2^w}$ as P, since $x^{(j)} = G_j$ is a null polynomial for every $j \ge d_w$. The normalization of P is therefore equal to the normalization of Q. In turn, $\tilde{Q}$ can be obtained by reducing every coefficient a_i modulo $2^{w-\nu_2(i!)}$, for $i = 0, 1, \ldots, \deg(Q)$. This shows the correctness of the Algorithm.

Regarding the complexity, the dominating step is the multiplication

$$F[\ell, \deg(f) + 1] \cdot v_P,$$

which is equivalent to performing no more than d_w scalar products between vectors of dimension $\deg(P)+1$. This yields the complexity $O(d_w \cdot \deg(P))$. Finally, we have the equality $O(d_w) = O(w)$ from Lemma III.8. ∎

Proposition V.3. *Let* $f(x) \in \mathbb{Z}_{2^w}[x]$. *Algorithm 2 produces a polynomial* $Q(x) \in \mathbb{Z}_{2^w}[x]$ *of degree* $\delta \ge \operatorname{rdeg}(P)$ *defining the same function as* P. *In fact, every such polynomial is given by this procedure.*

Algorithm 2 Generate equivalent univariate polynomial

```
Require: δ ≥ rdeg(P)
function EquivalentPolynomial(v_P, δ)
    u ← F[δ + 1, deg(P) + 1] · v_P
    for j ← 2, ..., δ + 1 do
        choose an arbitrary s_j, 0 ≤ s_j ≤ 2^min(ν_2(j!),w) − 1
        u[j] ← u[j] + s_j · c_j
    end for
    v' ← C[δ + 1, δ + 1] · u
    return v'
end function
```

Proof: As in the proof of Proposition V.2, the coordinates of the vector

$$u = F[\delta + 1, \deg(f) + 1] \cdot v_P$$

represent a polynomial Q in factorial basis of degree at most δ defining the same function as P. Adding a multiple of $c_j = 2^{\max(w-\nu_2(j!),0)}$ to any of the j^{th} component of u is the same as adding a multiple of G_j to the polynomial Q. But G_j defines the zero function, so we again obtain the same function defined by P. Finally, Theorem III.4 says that any zero polynomial we can add is of this form. ∎

Remark. *The for loops in Algorithms 1 and 2 are parallelizable, as coefficients in factorial basis are independent from one another. This is a particular advantage of working with this basis instead of the canonical one.*

B. A univariate example

We show an example with $w = 8$. As $w = 2^3$ is a power of 2, we know that the smallest degree of a *monic null polynomial* is $d_w = w + 2 = 10$. Consider the polynomial

$$\begin{aligned} P(x) = {} & 140x^{14} + 91x^{13} + 188x^{12} + 170x^{11} + 130x^{10} \\ & + 174x^9 + 176x^8 + 132x^7 + 19x^6 + 160x^5 \\ & + 143x^4 + 67x^3 + 112x^2 + 193x. \end{aligned}$$

The vector of coefficients of P in the canonical basis is

$$\begin{aligned} v_P = (&0, 193, 112, 67, 143, 160, 19, 132, \\ &176, 174, 130, 170, 188, 91, 140). \end{aligned}$$

In the notation of Algorithm 1, we have

$$\ell = \min(\deg(P) + 1, d_w) = \min(15, 10) = 10$$

and so we need to use the change of basis matrix

$$F[\ell, \deg(P) + 1] = F[10, 15].$$

The larger matrix $F[15, 15]$ can be found below on pg. 11. The other matrix needed is $C[\ell, \ell] = C[10, 10]$, see also pg. 11. Then, the vector of coefficients in the factorial basis is

$$u = F[10, 15] \cdot v = (0, 103, 30, 166, 162, 72, 51, 166, 60, 172).$$

We have to reduce the j^{th} coefficient of u modulo $c_j = 2^{\max(w-\nu_2(j!),0)}$. The vector of c_js equals

$$c = (256, 256, 128, 128, 32, 32, 16, 16, 2, 2).$$

After reducing u, we obtain

$$u' = (0, 103, 30, 38, 2, 8, 3, 6, 0, 0).$$

Finally, we multiply $C[10,10]\cdot u'$ to obtain the normalized polynomial

$$\tilde{P}(x) = 6x^7+133x^6+245x^5+119x^4+159x^3+16x^2+193x.$$

A short computation shows that indeed $P(a) \equiv \tilde{P}(a) \pmod{2^w}$ for all $a \in \mathbb{Z}_{2^w}$.

C. Multivariate algorithms

To complement the univariate algorithms, we explain how to adapt them to perform multivariate reduction and generation of equivalent polynomials in $\mathbb{Z}_{2^w}[x_1,\dots,x_k]$. We first define normalized polynomials.

Definition V.4. *Let $P(x) \in \mathbb{Z}_{2^w}[x_1,\dots,x_k]$ be a polynomial. The* normalization of P *is the unique polynomial $\tilde{P}$ such that*

- $P(a_1,\dots,a_k) \equiv \tilde{P}(a_1,\dots,a_k) \pmod{2^w}$ *for all* $a_1,\dots,a_k \in \mathbb{Z}_{2^w}^k$,
- $\tilde{P} = \sum_{\mathbf{j}\le(d,\dots,d)} b_{\mathbf{j}} x^{(\mathbf{j})}$ *in factorial basis, and*
- *if* $\mathbf{j} = (j_1,\dots,j_k)$ *is such that* $\sum_{i=1}^k \nu_2(j_i!) \ge w$, *then* $b_{\mathbf{j}} = 0$.
- *For every* $\mathbf{j}$ *with* $b_{\mathbf{j}} \neq 0$, *we have*

$$0 \le b_{\mathbf{j}} \le c_{\mathbf{j}} - 1 = 2^{w-\sum_{i=1}^k \nu_2(j_i!)} - 1.$$

The normalized polynomial $\tilde{P}$ exists and is unique by Theorem IV.1 and the fact that $\{x^{(\mathbf{d})}\}_{\mathbf{d}\in\mathbb{N}^k}$ is a basis.

According to Theorem IV.1 and the exposition around it, one can build the matrices of change of basis between $\{x^{\mathbf{d}}\}_{\mathbf{d}\in\mathbb{N}^k}$ and $\{x^{(\mathbf{d})}\}_{\mathbf{d}\in\mathbb{N}^k}$. In practice, it is ineffective to generate the full matrices, especially if we want to use them to alter *sparse* polynomials.

Instead, we do as follows. Fix a positive integer d and let $\mathbf{d} = (d,\dots,d) \in \mathbb{N}^k$. We know that $\{x_1^{r_1}\cdots x_r^{r_k}\}_{r_i\le d}$ is a basis of $\mathbb{Z}_{2^w}[x_1,\dots,x_k]_{\mathbf{d}}$. On the other hand, a basis of $\mathbb{Z}_{2^w}[x_1]_d \otimes_{\mathbb{Z}_{2^w}} \cdots \otimes_{\mathbb{Z}_{2^w}} \mathbb{Z}_{2^w}[x_k]_d$ is given by $\{x_1^{r_1}\otimes\cdots\otimes x_r^{r_k}\}_{r_i\le d}$. We have an isomorphism of $\mathbb{Z}_{2^w}$-modules

$$\mathbb{Z}_{2^w}[x_1]_d \otimes_{\mathbb{Z}_{2^w}} \cdots \otimes_{\mathbb{Z}_{2^w}} \mathbb{Z}_{2^w}[x_k]_d \simeq \mathbb{Z}_{2^w}[x_1,\dots,x_k]_{\mathbf{d}}$$

which consists in mapping $x_1^{r_1}\otimes\cdots\otimes x_r^{r_k}$ to $x_1^{r_1}\cdots x_r^{r_k}$. Therefore, this isomorphism maps

$$x_1^{(j_1)}\otimes\cdots\otimes x_r^{(j_k)} \mapsto x_1^{(j_1)}\cdots x_r^{(j_k)}.$$

At this point we need to introduce the Kronecker product of two matrices: if A is an $m\times n$ matrix and B is a $p\times q$ matrix, then its *Kronecker product* is given by

$$A\otimes B = \begin{pmatrix} a_{00}B & \cdots & a_{0(n-1)}B \\ \vdots & \ddots & \vdots \\ a_{(m-1)0}B & \cdots & a_{(m-1)(n-1)}B \end{pmatrix}.$$

The reason to index matrices from 0 will be clear shortly. The entry in row r and column c of $A\otimes B$ is computed by

$$(A\otimes B)_{r,c} = a_{[r/m],[c/n]}\cdot b_{r \bmod p, c \bmod q} \tag{1}$$

The matrices changing between the bases $\{x^{\mathbf{d}'}\}_{\mathbf{d}'\le\mathbf{d}}$ and $\{x^{(\mathbf{d}')}\}_{\mathbf{d}'\le\mathbf{d}}$ are then given by the Kronecker product matrices

$$F^{\otimes k} := F\otimes\overset{k}{\cdots}\otimes F,\ C^{\otimes k} := C\otimes\overset{k}{\cdots}\otimes C.$$

This corresponds to changing the basis in each component of the tensor product $\mathbb{Z}_{2^w}[x_1]_d\otimes_{\mathbb{Z}_{2^w}}\cdots\otimes_{\mathbb{Z}_{2^w}}\mathbb{Z}_{2^w}[x_k]_d$. To make sense of the orderings of the bases, one needs to take the *inverse lexicographic* monomial ordering.

With this, we do not need to generate the matrices for changing bases in $\mathbb{Z}_{2^w}[x_1,\dots,x_k]_{\mathbf{d}}$, it is enough to compute our original F and C to change bases in $\mathbb{Z}_{2^w}[x]_d$. However, this is not significantly faster than generating the full matrices $F^{\otimes k}$, $C^{\otimes k}$, it just makes the process tidier. The actual improvement comes from computing only portions of these matrices.

Say we want to compute the coefficients in the factorial basis of the monomial $x_1^{c_1}\cdots x_k^{c_k}$. In the inverse lexicographic ordering of $\mathbb{Z}_{2^w}[x_1,\dots,x_k]_{\mathbf{d}}$, this monomial lies in position

$$c = \sum_{i=1}^k c_i (d+1)^{i-1} \tag{2}$$

(observe that we are labeling positions in the basis from 0 to $(d+1)^k-1$, this also makes matrix indexing work). Hence, if a polynomial $P(x)$ has nonzero coefficient in the monomial $x_1^{c_1}\cdots x_k^{c_k}$, then we need to know the c^{th} column of the matrix $F^{\otimes k}$. We do this for every monomial appearing in P in order to express P in factorial basis. A similar reasoning shows us how to go from factorial to canonical basis by indexing columns of the matrix $C^{\otimes k}$.

It remains to see how to compute a particular column of $F^{\otimes k}$. In fact, it is possible to index the entry in the r^{th} row and c^{th} column of $F^{\otimes k}$, we denote this number by $F^{\otimes k}_{r,c}$. We do this by producing the base-d expansion of r and c,

$$r = \sum_{i=1}^k r_i (d+1)^{i-1},\ c = \sum_{i=1}^k c_i (d+1)^{i-1},$$

with $0\le r_i, c_i \le d$, and then we obtain

$$F^{\otimes k}_{r,c} = \prod_{i=1}^k F_{r_i,c_i}.$$

This generalizes Equation (1) for the case of tensor powers of square matrices. Iterating over every $0\le r\le (d+1)^k-1$ we obtain the necessary column of $F^{\otimes k}$.

To put the procedure described in algorithm form, we first let d_P be the maximum degree in any variable of a polynomial $P\in\mathbb{Z}_{2^w}[x_1,\dots,x_k]$. We assign to P a vector $v_P\in\mathbb{Z}_{2^w}^{(d_P+1)^k}$, where the c^{th} entry corresponds to the coefficient of $x_1^{c_i}\cdots x_k^{c_k}$ according to the base-(d_P+1) expression for c in (2). Conversely, every vector in $\mathbb{Z}_{2^w}^{(d_P+1)^k}$ can be interpreted as a polynomial in $\mathbb{Z}_{2^w}[x_1,\dots,x_k]$ via the same expression.

Once we have the coordinates of a multivariate polynomial in *factorial basis*, we can perform reduction as in Algorithm 1 according to Theorem IV.1.

Proposition V.5. *Algorithm 3 computes the normalization $\tilde{P}$ of P correctly in $O((d_P+1)^{2k})$ operations in $\mathbb{Z}_{2^w}$. If P has n nonzero coefficients in canonical basis, then the algorithm runs in $O(n(d_P+1)^k + d_w^{2k})$ operations in $\mathbb{Z}_{2^w}$.*

Proof: The proof is similar to Proposition V.2. ∎

Algorithm 3 Reduction of a multivariate polynomial

function NORMALIZEPOLYNOMIAL(v_P)
  $d_P \leftarrow$ maximum degree of P in any variable
  For each nonzero entry $v_P[c]$, compute the c^{th} column of $F[d_P+1, d_P+1]^{\otimes k}$
  $u \leftarrow F[d_P+1, d_P+1]^{\otimes k} \cdot v_P$ (sparse multiplication)
  for $j \leftarrow 0, \ldots, (d_P+1)^k$ **do**
    Let $j_1, \ldots, j_k$ be the base-(d_P+1) expansion of j
    $u[j] \leftarrow u[j] \ \% \ c_{j_1,\ldots,j_k}$
  end for
  remove all rightmost zeros from u
  $\tilde{v} \leftarrow C[d_w, d_w]^{\otimes k} \cdot u$
  return $\tilde{v}$
end function

We remark that it is possible to optimize Algorithm 3 whenever $d_P \geq d_w$. It is also easy to combine Algorithms 2 and 3 to create polynomials defining equal functions $\mathbb{Z}_{2^w}^k \to \mathbb{Z}_{2^w}$.

D. A multivariate example

Consider the following polynomial in $\mathbb{Z}_{2^8}[x_1, x_2]$, we have $k = 2$ variables.

$$\begin{aligned} P(x_1,x_2) = {} & x_1^3x_2^8 + 228x_1^3x_2^7 + 253x_1^2x_2^8 + 66x_1^3x_2^6 \\ & + 84x_1^2x_2^7 + 2x_1x_2^8 + 4x_1^4x_2^4 + 88x_1^3x_2^5 + 58x_1^2x_2^6 \\ & + 200x_1x_2^7 + 232x_1^4x_2^3 + 89x_1^3x_2^4 + 248x_1^2x_2^5 \\ & + 132x_1x_2^6 + 44x_1^4x_2^2 + 68x_1^3x_2^3 + 217x_1^2x_2^4 \\ & + 176x_1x_2^5 + 232x_1^4x_2 + 4x_1^3x_2^2 + 220x_1^2x_2^3 \\ & + 202x_1x_2^4 + 224x_1^3x_2 + 193x_1^2x_2^2 + 248x_1x_2^3 \\ & + 8x_1^2x_2 + 16x_1x_2^2 + 48x_1x_2. \end{aligned}$$

The maximum degree in x_2 is $d_P = 8$, so we need the matrix $F[9,9]^{\otimes 2}$. Computing only the appropriate columns of this matrix, we obtain the factorial basis expression of P,

$$P(x_1,x_2) = x^{(1,1)} + x^{(2,1)} + x^{(1,2)} + x^{(2,2)} + 4x^{(4,4)} + x^{(3,8)}.$$

To perform the reduction of coefficients, we find the coefficients $c_{\mathbf{j}}$,

$$\begin{aligned} c_{(1,1)} &= 2^{8-\nu_2(1!)-\nu_2(1!)} = 2^8 = 256 \\ c_{(2,1)} &= c_{(1,2)} = 2^{8-\nu_2(2!)-\nu_2(1!)} = 2^7 = 128 \\ c_{(2,2)} &= 2^{8-\nu_2(2!)-\nu_2(2!)} = 2^6 = 64 \\ c_{(4,4)} &= 2^{8-\nu_2(4!)-\nu_2(4!)} = 2^2 = 4 \\ c_{(3,8)} &= 2^{8-\nu_2(3!)-\nu_2(8!)} = 2^0 = 1. \end{aligned}$$

Hence, the normalization of P in factorial basis reads

$$\tilde{P}(x_1,x_2) = x^{(1,1)} + x^{(2,1)} + x^{(1,2)} + x^{(2,2)}.$$

Using the inverse lexicographic ordering, the coefficients of $\tilde{P}$ in the factorial basis $\{x^{(i,j)}\}_{0\leq i,j\leq 2}$ of $\mathbb{Z}_{2^8}[x_1,x_2]_{(2,2)}$ are

$$u = (0,0,0,0,1,1,0,1,1),$$

according to the rule that $x^{(i,j)}$ corresponds to the entry $i+3j$.

The matrix $C[3,3]^{\otimes 2} = C[3,3] \otimes C[3,3]$ equals

$$\left(\begin{array}{ccc|ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 255 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 255 & 0 & 255 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 255 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 255 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right).$$

Hence we have

$$\tilde{v} = C[3,3]^{\otimes 2} \cdot u = (0,0,0,0,0,0,0,0,1)$$

which corresponds to $\tilde{P}(x_1,x_2) = x_1^2x_2^2$.

We stress that, in practice, we would never compute the full matrices $C[3,3]^{\otimes 2}$ or $F[9,9]^{\otimes 2}$, and the previous is displayed only for illustrative purposes.

VI. LIMITATIONS AND FUTURE WORK

One of the biggest limitations when reverse engineering obfuscated code is determining its boundaries and structure in the first place. Thus, the issue of locating and extracting binary polynomials that could be reduced also applies if we analyze an executable program, especially if the binary polynomial semantics are mixed within other data flow and control flow obfuscation transformations.

In general, the normalized polynomial of a sparse polynomial will not necessarily be a sparse polynomial itself. This gets more noticeable with an increasing number of variables. If the underlying application, approach or tooling is more sensitive to a larger number of terms than a greater overall degree, this behavior might not be desired.

We have already mentioned some straightforward optimizations for the algorithms: parallelization of the for loops in both univariate and multivariate algorithms, and bounded matrix terms in the multivariate case when $d_P \geq d_w$. An extensive discussion of their scope and implications, from the perspective of algorithmic complexity and practical efficiency improvements, is left out from this paper due to space limitations, and might be included in a future extended revision of this study.

VII. CONCLUSION

This paper presents a thorough mathematical study of binary polynomial properties and the characterization of equivalent instances, both in the univariate and multivariate cases. This theoretical foundation supports the introduction of explicit algorithms to generate arbitrary equivalent binary polynomials that define the same function. In particular, we provide computationally efficient algorithms to reduce binary polynomials to a normalized form that uniquely determines the underlying function. These results are very encouraging and might lead to the development of more generic algorithms for equivalence generation and normalized reduction of polynomials modulo powers of prime numbers or an arbitrary integer.

ACKNOWLEDGMENT

Enric Florit was partially supported by the Spanish Ministry of Universities (FPU20/05059).

References

[1] M. F. Atiyah and I. G. Macdonald, *Introduction to commutative algebra* (Addison-Wesley Series in Mathematics), economy. Westview Press, Boulder, CO, 2016, pp. ix+128, For the 1969 original see [MR0242802].

[2] S. Banescu and A. Pretschner, "Chapter five - a tutorial on software obfuscation," in ser. Advances in Computers, A. M. Memon, Ed., vol. 108, Elsevier, 2018, pp. 283–353.

[3] L. Barhelemy, N. Eyrolles, G. Renault, and R. Roblin, "Binary permutation polynomial inversion and application to obfuscation techniques," ser. SPRO '16, Vienna, Austria: Association for Computing Machinery, 2016, pp. 51–59.

[4] L. Barthelemy, D. Kahrobaei, G. Renault, and Z. Šunić, "Quadratic time algorithm for inversion of binary permutation polynomials," in *Mathematical Software – ICMS 2018*, ser. LNCS, vol. 10931, Cham: Springer International Publishing, 2018, pp. 19–27.

[5] M. Brain, "Further steps down the wrong path: Improving the bit-blasting of multiplication," in *International Workshop on Satisfiability Modulo Theories*, 2021.

[6] L. Carlitz, "Functions and polynomials $(\mathrm{mod}\ p^n)$," eng, *Acta Arithmetica*, vol. 9, no. 1, pp. 67–78, 1964.

[7] C. Collberg and J. Nagra, *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*, 1st. Addison-Wesley Professional, 2009.

[8] T. Drane and G. Constantinides, "Leap in the formal verification of datapath," *DAC Knowledge Center*, 2012.

[9] N. Eyrolles, "Obfuscation with Mixed Boolean-Arithmetic Expressions : reconstruction, analysis and simplification tools," Theses, Université Paris-Saclay, Jun. 2017.

[10] G. Keller and F. R. Olson, "Counting polynomial functions $(\mathrm{mod}\ p^n)$," *Duke Mathematical Journal*, vol. 35, no. 4, pp. 835–838, 1968.

[11] R. Lidl and G. L. Mullen, "When does a polynomial over a finite field permute the elements of the field?" *The American Mathematical Monthly*, vol. 95, no. 3, pp. 243–246, 1988.

[12] R. Lidl and G. L. Mullen, "When does a polynomial over a finite field permute the elements of the field?, ii," *The American Mathematical Monthly*, vol. 100, no. 1, pp. 71–74, 1993.

[13] R. Lidl and W. B. Müller, "Permutation polynomials in RSA-cryptosystems," in *Annual International Cryptology Conference*, 1983.

[14] G. Mullen and H. Stevens, "Polynomial functions $(\mathrm{mod}\ m)$," *Acta Mathematica Hungarica*, vol. 44, no. 3, pp. 237–241, Sep. 1, 1984.

[15] R. L. Rivest, "Permutation polynomials modulo 2w," *Finite Fields and Their Applications*, vol. 7, no. 2, pp. 287–292, 2001.

[16] R. L. Rivest, M. Robshaw, R. Sidney, and Y. Yin, *The RC6 block cipher*, version 1.1, 1998.

$$F[15,15] = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 3 & 7 & 15 & 31 & 63 & 127 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\
0 & 0 & 0 & 1 & 6 & 25 & 90 & 45 & 198 & 209 & 114 & 85 & 254 & 249 & 234 & 234 \\
0 & 0 & 0 & 0 & 1 & 10 & 65 & 94 & 165 & 90 & 57 & 86 & 173 & 178 & 193 & 193 \\
0 & 0 & 0 & 0 & 0 & 1 & 15 & 140 & 26 & 39 & 29 & 202 & 72 & 21 & 27 & 27 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 21 & 10 & 86 & 43 & 31 & 132 & 96 & 85 & 85 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 28 & 206 & 248 & 243 & 196 & 224 & 128 & 128 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 36 & 238 & 104 & 51 & 92 & 192 & 192 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 45 & 131 & 3 & 78 & 26 & 26 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 55 & 169 & 157 & 112 & 112 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 66 & 127 & 18 & 18 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 78 & 39 & 39 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 91 & 91 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

Fig. 1. Matrix of change of basis for $\mathbb{Z}_{2^8}[x]_{14}$, from canonical to factorial.

$$C[10,10] = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 255 & 2 & 250 & 24 & 136 & 208 & 80 & 128 \\
0 & 0 & 1 & 253 & 11 & 206 & 18 & 28 & 12 & 240 \\
0 & 0 & 0 & 1 & 250 & 35 & 31 & 88 & 180 & 108 \\
0 & 0 & 0 & 0 & 1 & 246 & 85 & 33 & 113 & 44 \\
0 & 0 & 0 & 0 & 0 & 1 & 241 & 175 & 88 & 177 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 235 & 66 & 72 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 228 & 34 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 220 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Fig. 2. Matrix of change of basis for $\mathbb{Z}_{2^8}[x]_9$, from factorial to canonical.

[17] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communication of the ACM*, vol. 26, pp. 96–99, 1978.
[18] J. H. Ryu, "Permutation polynomial based interleavers for turbo codes over integer rings: Theory and applications," Ph.D. dissertation, The Ohio State University, The Ohio State University, 2007.
[19] J. Ryu and O. Takeshita, "On quadratic inverses for quadratic permutation polynomials over integer rings," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1254–1260, 2006.
[20] J. Ryu and O. Takeshita, "On quadratic inverses for quadratic permutation polynomials over integer rings," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1254–1260, 2006.
[21] T. Seed, C. Coppins, A. King, and N. Evans, "Polynomial analysis of modular arithmetic," in *Static Analysis*, M. V. Hermenegildo and J. F. Morales, Eds., Cham: Springer Nature Switzerland, 2023, pp. 508–539.
[22] N. Shekhar, P. Kalla, M. B. Meredith, and F. Enescu, "Simulation bounds for equivalence verification of polynomial datapaths using finite ring algebra," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 4, pp. 376–387, 2008.
[23] D. Singmaster, "On polynomial functions $\pmod m$," *Journal of Number Theory*, vol. 6, no. 5, pp. 345–352, 1974.
[24] L. Trifina and D. Tarniceriu, "Analysis of cubic permutation polynomials for turbo codes," *Wireless Personal Communications*, vol. 69, no. 1, pp. 1–22, 2013.
[25] L. Trifina and D. Tarniceriu, "On the equivalence of cubic permutation polynomial and arp interleavers for turbo codes," *IEEE Transactions on Communications*, vol. 65, no. 2, pp. 473–485, 2017.
[26] H. Zhao, P. Fan, and V. Tarokh, "On the equivalence of interleavers for turbo codes using quadratic permutation polynomials over integer rings," *IEEE Communications Letters*, vol. 14, no. 3, pp. 236–238, 2010.
[27] Y. Zhou and A. Main, "Diversity via code transformations: A solution for NGNA renewable security.," The NCTA Technical Papers, Tech. Rep., 2006.
[28] Y. Zhou, A. Main, Y. X. Gu, and H. Johnson, "Information hiding in software with Mixed Boolean-Arithmetic transforms," in *Information Security Applications*, S. Kim, M. Yung, and H.-W. Lee, Eds., ser. LNCS, vol. 4867, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 61–75.

APPENDIX A
A PROOF OF THEOREM IV.3

Recall from Section IV that we defined the indices

$$\mathcal{J} = \{(j_1,\dots,j_k) \in 2\mathbb{N}^k \mid 0 < \sum_{i=1}^{k} \nu_2(j_i!) < w\},$$

$$\mathcal{J}^m = \left\{ \mathbf{j} + 2\mathbf{e}_i \;\middle|\; \begin{array}{l} \mathbf{j} + 2\mathbf{e}_i \notin \mathcal{J}; \\ \mathbf{j} + 2\mathbf{e}_i - 2\mathbf{e}_{i'} \in \mathcal{J}\ \forall i' = 1,\dots,k \end{array} \right\}.$$

With this, we considered sets of polynomials

$$\mathcal{B}_1 = \{G_{\mathbf{j}} \mid \mathbf{j} \in \mathcal{J}\},$$
$$\mathcal{B}_2 = \{G_{\mathbf{j}} \mid \mathbf{j} \in \mathcal{J}^m\}.$$

We need the following simple lemma.

Lemma A.1. *The following properties hold:*

1) $\mathcal{J} \cap \mathcal{J}^m = \emptyset$.
2) *For every* $\mathbf{j} \in \mathcal{J}^m$, $G_{\mathbf{j}}$ *is monic.*
3) *If* $\mathbf{j}, \mathbf{j}' \in 2\mathbb{N}^k$ *satisfy* $\mathbf{0} \neq \mathbf{j} \leq \mathbf{j}'$ *and* $\mathbf{j}' \in \mathcal{J}$, *then* $\mathbf{j} \in \mathcal{J}$.

Proof: (1) Holds by definition of $\mathcal{J}^m$. For (2), note that every $\mathbf{j} \in \mathcal{J}^m$ satisfies $\sum_{i=1}^{k} \nu_2(j_i!) \geq w$, and therefore $c_{\mathbf{j}} = 2^0 = 1$. Finally, (3) holds because the inequality $\mathbf{0} \neq \mathbf{j} \leq \mathbf{j}'$ (and the entries of the vectors being even) implies

$$0 < \sum_{i=1}^{k} \nu_2(j_i!) \leq \sum_{i=1}^{k} \nu_2(j_i'!) < w.$$

■

We are ready to prove Theorem IV.3.

Theorem. *Let* $\mathbf{d} = (d_1,\dots,d_k) \in \mathbb{N}^k$. *If* $\sum_{i=1}^{k} \nu_2(d_i!) \geq w$, *then*

$$(1 + x_i)\mathcal{Z}_{\mathbf{d}}^{w,k} = \mathcal{Z}_{\mathbf{d}+\mathbf{e}_i}^{w,k}$$

for every $i = 1,\dots,k$. *Moreover,* $\mathcal{Z}^{w,k}$ *is generated as a* $\mathbb{Z}_{2^w}[x_1,\dots,x_k]$*-ideal by the minimal set* $\mathcal{B} := \mathcal{B}_1 \cup \mathcal{B}_2$.

Proof: For the first statement, it is enough to see the inclusion $\mathcal{Z}_{\mathbf{d}+\mathbf{e}_i}^{w,k} \subseteq (1 + x_i)\mathcal{Z}_{\mathbf{d}}^{w,k}$. The proof is identical to the univariate case, where $(1 + x)\mathcal{Z}_d^{w,1} = \mathcal{Z}_{d+1}^{w,1}$ for $d \geq d_w$ (see the proof of Theorem III.10).

Let us show that $\mathcal{B}$ is a generating set. By Theorem IV.1, it is enough to see that for every $\mathbf{j} = (j_1,\dots,j_k) \in \mathbb{N}^k$, $G_{\mathbf{j}}$ is a $\mathbb{Z}_{2^w}[x_1,\dots,x_k]$-combination of polynomials in $\mathcal{B}$.

Recall from Definition III.5 and Lemma III.6 that d_w is the smallest positive integer such that $\mathcal{Z}_{d_w}^{w,1}$ contains a monic polynomial. If $\mathbf{j}$ is such that $j_i > d_w$ for some $i = 1,\dots,k$, then $G_{\mathbf{j}}$ is a $\mathbb{Z}_{2^w}[x_1,\dots,x_k]$-combination of polynomials $G_{\mathbf{j}'}$ with $\mathbf{j}' \lneq \mathbf{j}$. Therefore, we may assume $\mathbf{j} \leq (d_w,\dots,d_w)$. Moreover, by Proposition IV.2 we may assume $j_1,\dots,j_k$ are even. Finally, we assume the induction hypothesis that for every $\mathbf{j}' \lneq \mathbf{j}$, $G_{\mathbf{j}'}$ is a $\mathbb{Z}_{2^w}[x_1,\dots,x_k]$-combination of elements in $\mathcal{B}$.

Given the above simplifications, assume further that $\mathbf{j}$ is such that $G_{\mathbf{j}} \notin \mathcal{B}$. Then we know that

- $\sum_{i=1}^{k} \nu_2(j_i!) \geq w$, and
- there is some $i_0 = 1,\dots,k$ such that $\mathbf{j} - 2\mathbf{e}_{i_0} \notin \mathcal{J}$.

If we now consider the multi-index $\mathbf{j} - 2\mathbf{e}_{i_0}$, we are one step closer to finding some multi-index in $\mathcal{J}^m$. In fact, by applying these conditions repeatedly, we arrive to some vector $\mathbf{f} = (f_1,\dots,f_k) \in \mathbb{N}^k$ with even components such that

- $\mathbf{f} \lneq \mathbf{j}$,
- $\sum_{i=1}^{k} \nu_2(f_i!) \geq w$, and

- for every $i = 1, \ldots, k$, $\mathbf{f} - 2\mathbf{e}_i \in \mathcal{J}$.

In other words, $\mathbf{f} \in \mathcal{J}^m$. Therefore, we find that $G_{\mathbf{f}} \in \mathcal{B}_2$ and is monic. In particular $G_{\mathbf{j}} - x^{\mathbf{j}-\mathbf{f}} G_{\mathbf{f}}$ can be expressed as a $\mathbb{Z}_{2^w}$-combination of polynomials $G_{\mathbf{j}'}$ with $\mathbf{j}' \lneqq \mathbf{j}$, and we are done by the induction hypothesis.

It remains to show that $\mathcal{B}$ is a minimal generating set, in the sense that no $G_{\mathbf{j}} \in \mathcal{B}$ can be written as a $\mathbb{Z}_{2^w}[x_1, \ldots, x_k]$-combination of other elements in $\mathcal{B}$. Suppose otherwise that

$$G_{\mathbf{j}} = \sum_{\substack{\mathbf{j}' \in \mathcal{J} \cup \mathcal{J}^m \\ \mathbf{j}' \neq \mathbf{j}}} p_{\mathbf{j}'} G_{\mathbf{j}'}$$

for polynomials $p_{\mathbf{j}'} \in \mathbb{Z}_{2^w}[x_1, \ldots, x_k]$. For every $\mathbf{j}'$ with $p_{\mathbf{j}'} \neq 0$, we necessarily have $\mathbf{j}' \lneqq \mathbf{j}$. If $\mathbf{j} \in \mathcal{J}$, then every $\mathbf{j}' \in \mathcal{J}$ by Lemma A.1. If we choose some $\mathbf{j}'_0$ such that $p_{\mathbf{j}'_0} \neq 0$ and $\nu_2(c_{\mathbf{j}'_0})$ is minimal, then $c_{\mathbf{j}}$ is a multiple of $c_{\mathbf{j}'_0}$. But $\mathbf{j}'_0 \lneqq \mathbf{j}$ implies $\nu_2(c_{\mathbf{j}'_0}) < \nu_2(c_{\mathbf{j}})$, which is a contradiction.

If on the other hand $\mathbf{j} \in \mathcal{J}^m$, then there must be some $\mathbf{j}'_0 = (j'_1, \ldots, j'_k) \in \mathcal{J}^m$ with $p_{\mathbf{j}'_0} \neq 0$. Because $\mathbf{j}'_0 \lneqq \mathbf{j}$, there must be some $i = 1, \ldots, k$ such that $j'_i < j_i$. Therefore

$$\mathbf{j}'_0 \leq \mathbf{j} - 2\mathbf{e}_i,$$

and the latter is an element of $\mathcal{J}$. Therefore we also have $\mathbf{j}'_0 \in \mathcal{J}$, which is again a contradiction. It follows that no element $G_{\mathbf{j}} \in \mathcal{B}$ is a $\mathbb{Z}_{2^w}[x_1, \ldots, x_k]$-combination of other elements, and therefore $\mathcal{B}$ is a minimal generating set. ■