Noname manuscript No.
(will be inserted by the editor)

The Number of Boolean Functions with Multiplicative Complexity 2

Magnus Gausdal Find · Daniel Smith-Tone · Meltem Sönmez Turan

Received: date / Accepted: date

Abstract Multiplicative complexity is a complexity measure, which is defined as the minimum number of AND gates required to implement a given primitive by a circuit over the basis (AND, XOR, NOT), with an unlimited number of NOT and XOR gates. Implementations of ciphers with a small number of AND gates are preferred in protocols for fully homomorphic encryption, multi-party computation and zero-knowledge proofs. In 2002, Fischer and Peralta [1] showed that the number of n-variable Boolean functions with multiplicative complexity one equals $2^{n+1}\binom{2^n-1}{2}$. In this paper, we study Boolean functions with multiplicative complexity 2. By characterizing the structure of these functions in terms of affine equivalence relations, we provide a closed form formula for the number of Boolean functions with multiplicative complexity 2.

Keywords Affine equivalence · Boolean functions · Cryptography · Multiplicative complexity · Self-mappings

Mathematics Subject Classification (2000) 94A60 · 06E30

1 Introduction

Multiplicative complexity is a complexity measure, which is defined as the minimum number of AND gates required to implement a given primitive

Magnus Gausdal Find
National Institute of Standards and Technology

Daniel Smith-Tone
National Institute of Standards and Technology &
University of Louisville

Meltem Sönmez Turan
National Institute of Standards and Technology &
Dakota Consulting Inc.

by a circuit over the basis (AND, XOR, NOT), with an unlimited number of NOT and XOR gates. In recent years, the relationships between multiplicative complexity and cryptography has been pointed out in several studies:

Multiplicative Complexity and Cryptography Many protocols for fully homomorphic encryption (e.g., [2]), multi-party computation (e.g., [3]) and zero-knowledge proofs of knowledge (e.g., [4]) all operate on the circuit representation of a function in a gate by gate manner. In these and many other protocols it is the case that processing AND gates is more expensive than processing XOR gates. We refer to references in [5] for a comprehensive list of examples. Courtois et al. [6] argued that minimizing the number of AND gates is important to prevent against side channel attacks such as differential power analysis. In Eurocrypt'15, Albrecht et al.[5] used this motivation to design the family of block ciphers LowMC.

On the other hand, having high multiplicative complexity is essential for security, e.g., Boyar et al. [7] showed that a cryptographic hash function must have a certain multiplicative complexity to be collision resistant.

Multiplicative Complexity and Circuit Design Determining the multiplicative complexity of a given function is computationally intractable, even for functions with a small number of variables. For general n, it is known that under standard cryptographic assumptions it is not possible to compute the multiplicative complexity in polynomial time in the length of the truth table [8]. The multiplicative complexity of a random n-variable Boolean function is at least $2^{n/2}-O(n)$ with high probability [9]. In 2010, Boyar et al. [10] proposed a two-stage heuristic method to minimize the gate complexity of Boolean circuits. In the first stage, the heuristic minimizes the number of AND gates required to implement the function, and then in the second stage, the linear components are optimized. Using this method, they constructed efficient circuits for the AES S-box over the basis (AND, XOR, NOT). In 2014, Turan and Peralta [11] studied the multiplicative complexity of five variable Boolean functions and showed that any five variable Boolean function can be implemented with at most four AND gates. Also in 2014, Zajac and Jókay [12] showed that any bijective 4×4 Sbox can be implemented with at most five AND gates.

This Paper In this work, we study the number of n-variable Boolean functions with multiplicative complexity M. Schnorr [13] gave a closed form for the number of n-variable *quadratic* functions with a given multiplicative complexity. In [9], it is shown that the number of functions with multiplicative complexity M is at most $2^{M^2+2M+2Mn+n+1}$. For large values of n and M, this bound is essentially tight [9], but it is unclear to what extent this is true for small constant values of M. In 2002, Fischer and Peralta [1] showed that there are precisely $2^{n+1}\binom{2^n-1}{2}$ Boolean functions on n variables with multiplicative complexity 1. Their result was based on properties on polynomial representations of such Boolean functions and the authors mention that this technique is unlikely to generalize to the case of even multiplicative complexity

2. In this work, we developed an alternative approach to count the number of Boolean functions with a given multiplicative complexity. Our approach relies on canonical circuits that compute functions of a certain multiplicative complexity. First, we count the number of such circuits, and then by solving a certain system of polynomial equations we obtain the number of functions from this. From a theoretical perspective this gives an algorithm, that given as input M, outputs a formula for the number of functions in n variables with multiplicative complexity M. Using this approach, we reprove the result of Fischer and Peralta [1], and extend the result to show that the number of Boolean functions with multiplicative complexity exactly 2 equals:

$$2^n(2^n-1)(2^n-2)(2^n-4)\left(\frac{2}{21}+\frac{2^n-8}{12}+\frac{2^n-8}{720}\right).$$

We remark that this is asymptotically a factor of $\frac{2^9 \cdot 720}{61} \approx 6043$ smaller than the bound from [9].

The organization of the paper is as follows. Section 2 gives definitions and some preliminary information about Boolean functions and multiplicative complexity. Section 3 discusses affine transformations and equivalence classes. Section 4 studies the Boolean functions with multiplicative complexity one. Section 5 provides the equivalence classes of Boolean functions with multiplicative complexity 2. Section 6 concludes the paper.

2 Preliminaries

Let $\mathbb{F}_2$ be the binary field. An n-variable Boolean function f is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Let $\mathcal{B}_n$ be the set of n-variable Boolean functions. A Boolean function $f \in \mathcal{B}_n$ can be represented uniquely by the list of output values for each input $T_f = (f(0,\ldots,0), f(0,\ldots,0,1),\ldots,f(1,\ldots,1))$. This list is called the *truth table* (representation) of f. Since the truth table has length 2^n and there are two possibilities for each, $|\mathcal{B}_n| = 2^{2^n}$. Another way of representing a Boolean function $f \in \mathcal{B}_n$ is by the unique multilinear polynomial called the *algebraic normal form* (ANF)

$$f(x_1,\ldots,x_n) = \sum_{u \in \mathbb{F}_2^n} a_u x^u, \tag{1}$$

where $a_u \in \mathbb{F}_2$ and $x^u = x_1^{u_1} x_2^{u_2} \cdots x_n^{u_n}$ is a *monomial* containing the variables x_i where $u_i = 1$. The degree of the monomial x^u is the number of variables appearing in x^u. The degree of a Boolean function, denoted d_f, is the highest degree of monomials occurring in its ANF. Functions with degree 2 are called quadratic and functions with degree 1 are called affine.

The *multiplicative complexity* of a Boolean function f is the minimum number of AND gates (multiplications in $\mathbb{F}_2$) that are sufficient to evaluate the function over the basis (AND, XOR, NOT) where all gates have fanin 2. It is known that a function with degree d has multiplicative complexity at least $d-1$ [14]. This bound is called the *degree bound*.

3 Affine Transformations and Equivalence Classes

Definition 1 [15] A map $S\colon \mathcal{B}_n \to \mathcal{B}_n$ is called an *affine transformation* if $g \overset{S}{\mapsto} f$ is defined by

$$f(\mathbf{x}) = g(A\mathbf{x} + \mathbf{a}) + \mathbf{b}^\top \mathbf{x} + c, \text{ for all } \mathbf{x},$$

where A is a non-singular $n \times n$ matrix over $\mathbb{F}_2$; $\mathbf{a}, \mathbf{b}, \mathbf{x}$ are column vectors in $\mathbb{F}_2^n$ and $c \in \mathbb{F}_2$.

An affine transformation can be characterized by the values of $A, \mathbf{a}, \mathbf{b}, c$. Directly from the definition of an affine transformation, it follows that the relation

$$R = \{(f, g) | \ \exists \text{ an affine transformation from } f \text{ to } g\},$$

is an equivalence relation on $\mathcal{B}_n$. This relation imposes equivalence classes on $\mathcal{B}_n$, and two functions in the same class are said to be *affine equivalent*. An algorithm to determine whether two functions are equivalent is given in [16].

Let $[f]$ denote the equivalence class containing the function f. For brevity, we refer to the function $f \in \mathcal{B}_n$ by its algebraic normal form. For example we will refer to the n-variable function $f(\mathbf{x}) = x_1x_2x_3$ as simply $x_1x_2x_3$, while $[x_1x_2x_3]$ refers to the equivalence class containing f.

By counting the number of choices of the A, $\mathbf{a}$, $\mathbf{b}$, and c from Definition 1, we get that for all $n \in \mathbb{N}$, the total number of distinct affine transformations applicable to any given function $f \in \mathcal{B}_n$ is

$$\tau_n = 2^{2n+1} \prod_{i=0}^{n-1} (2^n - 2^i).$$

It was shown in 1972 by Berlekamp and Welch that $\mathcal{B}_5$ has 48 equivalence classes [15]. Maiorana [17] proved that $\mathcal{B}_6$ has 150 357 equivalence classes. This was independently verified by Fuller [16] and Braeken et al. [18]. It was shown by Hou [19] that $\mathcal{B}_7$ has 63 379 147 320 777 408 548($\approx 2^{65.78}$) classes. See Table 1 for the equivalence classes with n=2,3,4 variables.

It should be noted that multiplicative complexity is *affine invariant*, i.e., the multiplicative complexity of a Boolean function does not change after applying an affine transformation to the function. Hence functions in the same equivalence class all have the same mulitiplicative complexity.

3.1 Properties of Affine Transformations

For the purposes of the rest of the paper, we represent the affine transformation from $f(\mathbf{x})$ to $f(A\mathbf{x} + \mathbf{a}) + \mathbf{b}^\top \mathbf{x} + c$ using the tuple $S = (A, \mathbf{a}, \mathbf{b}, c)$. The collection of affine transformations $(A, \mathbf{a}, \mathbf{b}, c)$ forms a group $\mathcal{A}_n$ under the operation $\otimes$ defined by

$$(A_1, \mathbf{a}_1, \mathbf{b}_1, c_1) \otimes (A_2, \mathbf{a}_2, \mathbf{b}_2, c_2) = (A_2A_1, A_2\mathbf{a}_1 + \mathbf{a}_2, A_1^\top \mathbf{b_2} + \mathbf{b}_1, \mathbf{b}_2^\top \mathbf{a}_1 + c_1 + c_2).$$

n	Equivalence Class $[f]$	$\lvert[f]\rvert$	$\lvert\Theta(f)\rvert$	dimension k	τ_n
2	$[x_1]$	8	24	1	192
	$[x_1x_2]$	8	24	2	
3	$[x_1]$	16	1344	1	21504
	$[x_1x_2]$	112	192	2	
	$[x_1x_2x_3]$	128	168	3	
4	$[x_1]$	32	322560	1	10321920
	$[x_1x_2]$	1120	9216	2	
	$[x_1x_2x_3]$	3840	2688	3	
	$[x_1x_2 + x_3x_4]$	896	11520	4	
	$[x_1x_2x_3 + x_1x_4]$	26880	384	4	
	$[x_1x_2x_3x_4]$	512	20160	4	
	$[x_1x_2x_3x_4 + x_1x_2]$	17920	579	4	
	$[x_1x_2x_3x_4 + x_1x_2 + x_3x_4]$	14336	720	4	

Table 1 Equivalence classes for $n = 2, 3, 4$.

Note that this group operation corresponds to applying the affine transformation $(A_2, \mathbf{a}_2, \mathbf{b}_2, c_2)$, followed by $(A_1, \mathbf{a}_1, \mathbf{b}_1, c_1)$.

Let $f \in \mathcal{B}_n$ and let L_f be the number of distinct input variables appearing in the ANF of f. For example, for $f(x_1, \ldots, x_6) = x_1x_2x_3 + x_3x_4$, L_f is 4. It is easy to see that (i) the dimension of $[f]$ is at least the degree of f, and (ii) L_f is not affine invariant.

Definition 2 The equivalence class $[f]$ has *dimension* k $(0 \leq k \leq n)$, if the smallest L_g for $g \in [f]$ is k, i.e., $k = \min_{g\in[f]} L_g$. Overloading the definition, any function $g \in [f]$ is also said to have *dimension* k.

For a function $g \in \mathcal{B}_k$, the *embedding* of g in $\mathcal{B}_n$ is the function $g_n(\mathbf{x}) = g(x_1, \ldots, x_k)$. When certain inputs do not affect the value of the output of a function we denote them with $*$ or when clear from context ignore them altogether. Thus $g_n(\mathbf{x}) = g_n(x_1, \ldots, x_k, *, \ldots, *) = g(x_1, \ldots, x_k)$. We say that the function $g \in \mathcal{B}_k$ is a k-dimensional representative of $f \in \mathcal{B}_n$ if the embedding of g in $\mathcal{B}_n$ is affine equivalent to f. Note that if the dimension of $[f]$ is k then there exist ℓ-dimensional representatives of f for all $\ell \geq k$.

Next we derive a useful computational result; namely, affine equivalence can be tested with low-dimensional representatives.

Lemma 1 *Let $f, g \in \mathcal{B}_n$ be of dimension at most k. Let f_k and g_k be k-dimensional representatives of f and g, respectively. Then $g \in [f]$ if and only if $g_k \in [f_k]$.*

Proof Suppose that f and g are affine equivalent. Then there exists an affine transformation $S \in \mathcal{A}_n$ such that $S(f) = g$. Let f_n and g_n be the embeddings of f_k and g_k in $\mathcal{B}_n$. By definition there exist affine transformations $T, U \in \mathcal{A}_n$ such that $T(f_n) = f$ and $U(g_n) = g$. Thus $U^{-1} \otimes S \otimes T(f_n) = g_n$, and so f_n and g_n are affine equivalent. We may write

$$U^{-1} \otimes S \otimes T = \left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}, \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}, \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{bmatrix}, c \right),$$

where A is $k \times k$, B is $k \times n-k$, C is $n-k \times k$, D is $n-k \times n-k$, $\mathbf{a_1}$ and $\mathbf{b_1}$ are k-dimensional, and $\mathbf{a_2}$ and $\mathbf{b_2}$ are $n-k$-dimensional. Let $\mathbf{x_1} = \begin{bmatrix} x_1 & \cdots & x_k \end{bmatrix}^\top$ and $\mathbf{x_2} = \begin{bmatrix} x_{k+1} & \cdots & x_n \end{bmatrix}^\top$. Since the variables $x_{k+1}, \ldots, x_n$ do not occur in the ANF of g_n, we find that

$$f_n\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}\right)$$

is linear in $\mathbf{x_2}$. Thus

$$f_n\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}\right) + f_n\left(\begin{bmatrix} A & 0 \\ C & D \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{0} & \mathbf{b_2'} \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix}.$$

Adding $\begin{bmatrix} \mathbf{b_1} & \mathbf{b_2} \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + c$ to both sides of this equation we obtain on the left hand side g_n and on the right hand side

$$f_n\left(\begin{bmatrix} A & 0 \\ C & D \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}\right) + \begin{bmatrix} \mathbf{b_1}^\top & \mathbf{b_2} + \mathbf{b_2'}^\top \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + c.$$

Since the output of f_n doesn't involve the variables $x_{k+1}, \ldots, x_n$, we learn that $\mathbf{b_2'} = \mathbf{b_2}$. Clearly A is of full rank. Then for all $\mathbf{x} = \mathbf{x_1}||\mathbf{x_2}$,

$$\begin{aligned} g_k(\mathbf{x_1}) = g_n(\mathbf{x_1}||\mathbf{x_2}) &= f_n\left(\begin{bmatrix} A & 0 \\ C & D \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}\right) + \begin{bmatrix} \mathbf{b_1}^\top & \mathbf{0}^\top \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + c \\ &= f_n([A\mathbf{x_1} + \mathbf{a_1}]||[C\mathbf{x_1} + D\mathbf{x_2}]) + \mathbf{b_1}^\top\mathbf{x_1} + c \\ &= f_k(A\mathbf{x_1} + \mathbf{a_1}) + \mathbf{b_1}^\top\mathbf{x_1} + c. \end{aligned}$$

Thus f_k and g_k are affine equivalent.

To prove the converse, assume that f_k and g_k are affine equivalent. Then there exists an affine transformation S such that $S(f_k) = g_k$. We may write $S = (A, \mathbf{a}, \mathbf{b}, c)$. Consider the affine transformation

$$\tilde{S} = \left(\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}, c\right).$$

It is obvious that for all $\mathbf{x} \in \mathbb{F}_2^n$ with $\mathbf{x} = \mathbf{x_1}||\mathbf{x_2}$ where $\mathbf{x_1} \in \mathbb{F}_2^k$ and $\mathbf{x_2} \in \mathbb{F}_2^{n-k}$ that

$$\begin{aligned} \tilde{S}(f_n)(\mathbf{x}) &= f_n\left(\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + \begin{bmatrix} \mathbf{a} \\ \mathbf{0} \end{bmatrix}\right) + \begin{bmatrix} \mathbf{b} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{bmatrix} + c \\ &= f_n(A\mathbf{x_1} + \mathbf{a}||\mathbf{x_2}) + \mathbf{b}^\top\mathbf{x_1} + c \\ &= f_k(A\mathbf{x_1} + \mathbf{a}) + \mathbf{b}^\top\mathbf{x_1} + c \\ &= g_k(\mathbf{x_1}) \\ &= g_n(\mathbf{x}). \end{aligned}$$

Thus f_n and g_n are affine equivalent, and by transitivity, f and g are affine equivalent. Thus the proof is complete. □

3.2 Self-Mappings

In this section we establish a few facts on a particular kind of affine transformations, called self-mappings.

Definition 3 [16] A self mapping of $f \in \mathcal{B}_n$ is an affine transformation such that $f(\mathbf{x}) = f(A\mathbf{x}+\mathbf{a})+\mathbf{b}^\top\mathbf{x}+c$, where A is a non-singular $n\times n$ matrix over $\mathbb{F}_2$; $\mathbf{a},\mathbf{b},\mathbf{x}$ are column vectors in $\mathbb{F}_2^n$ and $c\in\mathbb{F}_2$.

Let $f \in \mathcal{B}_n$, and $\Theta(f)$ be the set of self-mappings of f. We first remark that $\Theta(f)$ is closed under the group operation $\otimes$, since for every $\mathbf{x}\in\mathbb{F}_2^n$ and every pair of self-mappings (S_1, S_2), defined by the tuples $S_1 = (A_1, \mathbf{a}_1, \mathbf{b}_1, c_1)$ and $S_2 = (A_2, \mathbf{a}_2, \mathbf{b}_2, c_2)$, we have:

$$\begin{aligned}(S_1\otimes S_2)(f)(\mathbf{x}) &= f(A_2A_1\mathbf{x}+A_2\mathbf{a}_1+\mathbf{a}_2)+(A_1^\top\mathbf{b}_2+\mathbf{b}_1)^\top\mathbf{x}+\mathbf{b}_2^\top\mathbf{a}_1+c_1+c_2\\ &= f(A_2A_1\mathbf{x}+A_2\mathbf{a}_1+\mathbf{a}_2)+\mathbf{b}_2^\top A_1\mathbf{x}+\mathbf{b}_1^\top\mathbf{x}+\mathbf{b}_2^\top\mathbf{a}_1+c_1+c_2\\ &= f(A_2(A_1\mathbf{x}+\mathbf{a}_1)+\mathbf{a}_2)+\mathbf{b}_2^\top(A_1\mathbf{x}+\mathbf{a}_1)+c_2+\mathbf{b}_1^\top\mathbf{x}+c_1.\end{aligned} \tag{2}$$

Now since $f(A_2\mathbf{x}+\mathbf{a}_2)+\mathbf{b}_2^\top\mathbf{x}+c_2 = f(\mathbf{x})$ for all $\mathbf{x}$ and since $A_1\mathbf{x}+\mathbf{a}_1$ is a permutation, we have that

$$f(A_2(A_1\mathbf{x}+\mathbf{a}_1)+\mathbf{a}_2)+\mathbf{b}_2^\top(A_1\mathbf{x}+\mathbf{a}_1)+c_2 = f(A_1\mathbf{x}+\mathbf{a}_1), \tag{3}$$

for all $\mathbf{x}$. Thus from equations (2) and (3) we obtain:

$$\begin{aligned}(S_1\otimes S_2)(f)(\mathbf{x}) &= f(A_2(A_1\mathbf{x}+\mathbf{a}_1)+\mathbf{a}_2)+\mathbf{b}_2^\top(A_1\mathbf{x}+\mathbf{a}_1)+c_2+\mathbf{b}_1^\top\mathbf{x}+c_1\\ &= f(A_1\mathbf{x}+\mathbf{a}_1)+\mathbf{b}_1^\top\mathbf{x}+c_1\\ &= f(\mathbf{x}).\end{aligned}$$

Since $\Theta(f)$ is finite, $\Theta(f)$ forms a subgroup of $\mathcal{A}_n$. By the Orbit-Stabilizer Theorem, we have

$$|[f]| = \frac{\tau_n}{|\Theta(f)|}.$$

In the following lemma, we let σ be the number of ways of choosing $k+1$ n-dimensional affine forms, $t_1,\ldots,t_{k+1}$, over $x_1,\ldots,x_n$ the first k of which are linearly independent.

Lemma 2 *Let $f\in\mathcal{B}_n$, and $g\in\mathcal{B}_k$ be a k-dimensional representative of f. Let g_n be the embedding of g in $\mathcal{B}_n$ and let ρ be the number of choices of affine forms $r_1,\ldots,r_{k+1}$ such that*

$$g_n(t_1,\ldots,t_k,*,\ldots,*)+t_{k+1} = g_n(r_1,\ldots,r_k,*,\ldots,*)+r_{k+1} \text{ for all } \mathbf{x}\in\mathbb{F}_2^n.$$

Then $|[f]| = |[g_n]| = \frac{\sigma}{\rho}$.

Proof Let $\mathcal{I}$ be the collection of self mappings $(A, \mathbf{a}, \mathbf{b}, c)$ with $\mathbf{b} = \mathbf{0}$ and $c = 0$ that satisfies the system of linear equations

$$A \cdot (x_1, \ldots, x_n)^T + \mathbf{a} = (x_1, \ldots, x_k, *, \ldots, *)^T.$$

Since $\mathcal{I}$ is a subgroup of $\Theta(g_n)$, the cosets of $\mathcal{I}$ form a refinement of the partition of $\mathcal{A}_n$ formed by the cosets of $\Theta(g_n)$. Hence $|\mathcal{A}_n : \mathcal{I}| = |\mathcal{A}_n : \Theta(g_n)||\Theta(g_n) : \mathcal{I}|$.

Any affine mapping in $\mathcal{I}$ has the form:

$$\left(\begin{bmatrix} I & 0 \\ A & B \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ \mathbf{a} \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, 0 \right),$$

where I is the $k \times k$ identity matrix, A is an $(n-k) \times k$ matrix, B is an $(n-k) \times (n-k)$ matrix, and $\mathbf{a}$ is an $n-k$-dimensional column vector. Consider an arbitrary left coset of $\mathcal{I}$. That is, a left coset generated by

$$S = \left(\begin{bmatrix} C & D \\ E & F \end{bmatrix}, \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}, \begin{bmatrix} \mathbf{e} \\ \mathbf{f} \end{bmatrix}, h \right),$$

where each of the block matrices and block vectors are of the appropriate dimension. Any member of the left coset $S \otimes \mathcal{I}$ is given by

$$\left(\begin{bmatrix} C & D \\ AC + BE & AD + BF \end{bmatrix}, \begin{bmatrix} \mathbf{c} \\ A\mathbf{c} + B\mathbf{d} + \mathbf{a} \end{bmatrix}, \begin{bmatrix} \mathbf{e} \\ \mathbf{f} \end{bmatrix}, h \right).$$

Here, the first k rows of the first two coordinates as well as the last two coordinates of this element are identical to S; therefore any two elements in $S \otimes \mathcal{I}$ share these $k+1$ affine forms (the first k of which are necessarily independent) in common. On the other hand, since g_n is the embedding of a k-dimensional simple representative of f, different values of the last two coordinates of S give rise to different functions; thus, since each such coset of $\mathcal{I}$ is determined by the first k rows of the first two coordinates of S along with the last two coordinates of S, the cosets of $\mathcal{I}$ are uniquely determined by such choices of $k+1$ affine forms with the first k linearly independent. Therefore $|\mathcal{A}_n : \mathcal{I}| = \sigma$ and $|\Theta(f) : \mathcal{I}| = \rho$. Thus $|[f]| = |[g_n]| = \frac{|\mathcal{A}_n : \mathcal{I}|}{|\Theta(f) : \mathcal{I}|} = \frac{\sigma}{\rho}$. □

A particular consequence of the lemma is the following tool for computationally determining the number of self-mappings of a Boolean function by computing on a much smaller search space.

Corollary 1 *Let $f \in \mathcal{B}_n$ be of dimension k. Let $g \in \mathcal{B}_k$ be a k-dimensional representative of f and let g_n be the embedding of g in $\mathcal{B}_n$. The number of self-mappings θ of f is*

$$\theta = 2^{n-k} \prod_{i=k}^{n-1} (2^n - 2^i)\rho,$$

where ρ is the number of choices of affine forms $r_1, \ldots, r_{k+1}$ such that

$$g_n(t_1, \ldots, t_k, *, \ldots, *) + t_{k+1} = g_n(r_1, \ldots, r_k, *, \ldots, *) + r_{k+1} \text{ for all } \mathbf{x} \in \mathbb{F}_2^n,$$

as in Lemma 2.

Proof When $n = k$ we have $\theta = \rho$ in the formula. If $k < n$, by Lemma 2, we have that $\frac{\sigma}{\rho} = |[g_n]| = \frac{\tau_n}{\theta}$, and thus $\theta = \frac{\tau_n \rho}{\sigma}$. Since $\sigma = 2^{n+k+1} \prod_{i=0}^{k-1} (2^n - 2^i)$, and since $|\Theta(f)| = |\Theta(g_n)|$, we have the result. □

We will count the number of functions with multiplicative complexity M using the following strategy. First show that all functions on n variables with multiplicative complexity M can be partitioned into equivalence classes, each having a representative of a dimension *independent of* n. Now the size of each of these equivalence classes (as a function of n) can be determined directly using Corollary 1 as τ_n/θ. The total number of functions is then the sum of the sizes of these equivalence classes.

4 Boolean Functions with Multiplicative Complexity One

Fischer and Peralta [1] provided the number of Boolean functions with multiplicative complexity one. In this section, we reprove their result using an alternative method which can easily be extended for multiplicative complexity two.

Proposition 1 *Let f be an n-variable Boolean function with multiplicative complexity 1. f is affine equivalent to $x_1 \cdot x_2$.*

Proof Let $f \in \mathcal{B}_n$ have multiplicative complexity exactly 1. Then f is ~~on~~ of the form

$$f(\mathbf{x}) = (\mathbf{a}^\top \mathbf{x}) \cdot (\mathbf{b}^\top \mathbf{x}) + \mathbf{c}^\top \mathbf{x} + d,$$

where $\mathbf{a}$ and $\mathbf{b}$ are distinct and nonzero. Note that when $\mathbf{a}$ and $\mathbf{b}$ do not satisfy these conditions, the multiplicative complexity of f is 0.

Define the function $g \in \mathcal{B}_n$ by

$$g(x_1, \ldots, x_n) = x_1 \cdot x_2.$$

We want to show that f and g are affine equivalent. It suffices to show that there exists an invertible A such that

$$f(\mathbf{x}) = g(A\mathbf{x}) + \mathbf{c}^\top \mathbf{x} + d.$$

One can let the first row of A be $\mathbf{a}^\top$, the second row be $\mathbf{b}^\top$. The last $n - 2$ vectors can be chosen arbitrarily under the condition that they together with $\mathbf{a}, \mathbf{b}$ form a basis of $\mathbb{F}_2^n$. This is possible since $\mathbf{a}, \mathbf{b}$ are distinct and therefore linearly independent. □

Thus it suffices to count the number of functions in the equivalence class from Proposition 1. To this end, we use Lemma 2.

Theorem 1 *The number of n-variable Boolean functions with multiplicative complexity 1 is exactly*

$$2^{n+1}(2^n - 1)(2^n - 2)/6.$$

Proof First we count the number of circuits with exactly one AND gate, then, compute the number of functions with multiplicative complexity 1. Written as a formula, such a circuit is of the form

$$C(x) = t_1(\mathbf{x}) \cdot t_2(\mathbf{x}) + t_3(\mathbf{x}),$$

where t_1, t_2 and t_3 are affine forms on $\mathbb{F}_2$. Since t_1, t_2 must be linearly independent, the number of such circuits is $4 \cdot 2^n \cdot (2^n - 1) \cdot 2^{n+1}$. Now we want to use Corollary 1. For this, we determine ρ, number of such triples of affine forms satisfying the above equation. An exhaustive search (we used a computer program but the system is small enough that one can do this by hand) found that $\rho = 24$. We get that the size of the equivalence class x_1x_2, the number of functions with multiplicative complexity 1 is

$$\frac{4 \cdot 2^n \cdot (2^n - 1) \cdot 2^{n+1}}{24},$$

which is what we wanted. □

5 Boolean Functions with Multiplicative Complexity Two

In this section, we generalize the proof technique from the previous section to count the number of functions with multiplicative complexity 2. We start by showing that there exist exactly three equivalence classes with multiplicative complexity 2.

Theorem 2 *Let f be an n-variable Boolean function with multiplicative complexity 2. Then f is affine equivalent to exactly one of the following three functions:*

1. $x_1x_2x_3$
2. $x_1x_2x_3 + x_1x_4$
3. $x_1x_2 + x_3x_4$

Proof First we show that each function with multiplicative complexity 2 falls in one of three classes. Consider a circuit, C, containing exactly two AND gates computing a function with multiplicative complexity two. Suppose first that there is no directed path from either of the AND gate to the other. In this case there exists affine forms $r_1, , \ldots, r_5$ such that the circuit computes the function

$$C(x) = r_1(x) \cdot r_2(x) + r_3(x) \cdot r_4(x) + r_5(x).$$

This function is affine equivalent to $x_1x_2 + x_3x_4$ via an affine transformation similar to the one demonstrated in the proof of Theorem 1.

Now suppose that there exist a directed path from one of the AND gates to the other. Call the functions computed by the two AND gates f_{A_1}, f_{A_2}, respectively. Suppose the topologically minimal AND gate computes the function $f_{A_1}(x) = r_1(x)r_2(x)$ for suitably chosen affine forms r_1, r_2.

We now claim that there exist affine forms r_3, r_4, r_5 such that the circuit computes the function

$$(r_1 \cdot r_2 + r_3) \cdot r_4 + r_5.$$

First, we can assume that f_{A_1} occurs only in one of the two inputs of f_{A_2}. To see this, notice that

$$(f_{A_1} + r_3) \cdot (f_{A_1} + r_4) = (f_{A_1} + r_3) \cdot (r_4 + r_3 + 1).$$

Therefore the topologically last AND gate, f_{A_2}, computes the function

$$f_{A_2} = (f_{A_1} + r_3) \cdot r_4.$$

So we have that the output of the circuit is either $f_{A_2} + r_5$, or $f_{A_2} + f_{A_1} + r_5$. for some affine function L_5.

We claim that the output can be assumed to be on the first form. Suppose $A_2 = (A_1 + r_3) \cdot r_4$, and let $A_2' = (A_1 + r_3) \cdot (r_4 + 1)$, $r_5' = r_5 + r_3$

$$A_2' + r_5' = (A_1 + r_3) \cdot (1 + r_4) + r_3 + r_5 = A_2 + A_1 + r_5.$$

This proves the claim. Now suppose that $r_3 \in \text{span}_{\mathbb{F}_2}\{r_1, r_2, r_4\}$, and let

$$r_3 = c_1 r_1 + c_2 r_2 + c_4 r_4.$$

By adding constants to r_1, r_2 we can assume that $c_1 = c_2 = 0$. That is we have that the function computed is:

$$(r_1 r_2 + c_4 r_4) r_4 + r_5,$$

again we can assume that $c_4 = 0$ since otherwise

$$(r_1 r_2 + r_4) r_4 + r_5 = r_1 r_2 r_4 + r_4 + r_5.$$

We observe that the affine functions r_1, r_2, r_4 must be linearly independent (otherwise the function has multiplicative complexity at most 1). We conclude that this function is affine equivelant to $x_1 x_2 x_3$.

Now suppose that r_3 is linearly independent of r_1, r_2, r_4. In this case the function computed is

$$r_1 r_2 r_4 + r_3 r_4 + r_5,$$

for linearly independent functions r_1, r_2, r_3, r_4. This function is clearly linearly equivalent to the function $x_1 x_2 x_3 + x_1 x_4$.

Finally we notice that these three functions indeed belong to three distinct equivalence classes. The function (3) has degree two, and is therefore not affine equivalent to a function of degree 3. Furthermore, by Lemma 1 a simple calculation shows that $[x_1 x_2 x_3]$ has dimension 3 whereas $[x_1 x_2 x_3 + x_1 x_4]$ has dimension 4. Thus these equivalence classes are distinct. □

The result readily implies that there are three different circuit types computing functions with multiplicative complexity 2. These are shown in Figure 1.

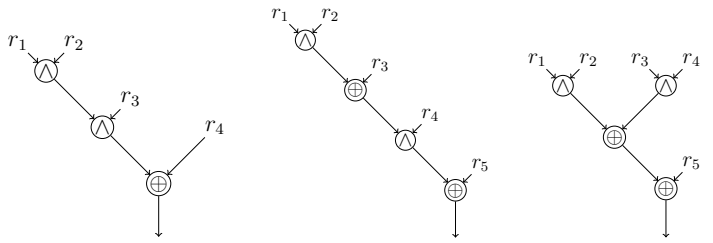

Fig. 1 The three different circuit types for the three equivalence classes.r_i denotes an affine form on the input variables. Every function with multiplicative complexity 2 can be computed by exactly one of the three displayed circuits.

Theorem 3 *The number of n-variable Boolean functions with multiplicative complexity 2 is exactly*

$$2^n(2^n-1)(2^n-2)(2^n-4)\cdot\left(\frac{2}{21}+\frac{2^n-8}{12}+\frac{2^n-8}{720}\right).$$

Proof By Theorem 2 it suffices to count the number of functions in each of the following three equivalence classes. By the proof of Theorem 2 there is a natural type of circuit associated with each equivalence class.

1. $[x_1x_2x_3]$
2. $[x_1x_2x_3 + x_1x_4]$
3. $[x_1x_2 + x_3x_4]$

We will count the number of functions in each class in a way similar in spirit, but slightly more complicated than what was done in the proof of Theorem 1: We will count the number of circuit layouts for each type. Then we will compute the value of ρ from Corollary 1 to obtain the actual number of functions.

Type 1: We want to determine the value ρ from Corollary 1. That is, count the number of choices of affine forms (r_1, r_2, r_3, r_4) such that for all $\mathbf{x}$

$$t_1(\mathbf{x})\cdot t_2(\mathbf{x})\cdot t_3(\mathbf{x}) + t_4(\mathbf{x}) = r_1(\mathbf{x})\cdot r_2(\mathbf{x})\cdot r_3(\mathbf{x}) + r_4(\mathbf{x}). \tag{4}$$

By affine equivalence, the number of solutions (r_1, r_2, r_3, r_4) does not depend on the actual choice of (t_1, t_2, t_3, t_4). By going through all the possibly choices one can verify that the number of solutions 168 times. By Corollary 1, the number of functions in this equivalence class is

$$\frac{(2^n-1)(2^n-2)(2^n-4)2^{n+1}}{21}.$$

Type 2: Again here we can count the number of ways of choosing $(r_1, \ldots, r_5)$ in such that for all $\mathbf{x}$

$$r_1(\mathbf{x})r_2(\mathbf{x})r_3(\mathbf{x}) + r_1(\mathbf{x})r_4(\mathbf{x}) + r_5(\mathbf{x}) = t_1(\mathbf{x})t_2(\mathbf{x})t_3(\mathbf{x}) + t_1(\mathbf{x})t_4(\mathbf{x}) + t_5(\mathbf{x}).$$

Using a computer search one can verify that this number is 384, so by Corollary 1 the number of functions computable by this type of circuit is.

$$\frac{2^n(2^n-1)(2^n-2)(2^n-4)(2^n-8)}{12}.$$

Type 3: Again, a computer search can verify that for this type we have that $\rho = 23040$. That is, the number of distinct functions computable by circuits of this type is

$$\frac{(2^n-1)(2^n-2)(2^n-4)(2^n-8)2^n}{720}.$$

We conclude that the total number of functions in $\mathcal{B}_n$ with multiplicative precisely 2 is:

$$2^n(2^n-1)(2^n-2)(2^n-4) \cdot \left(\frac{2}{21} + \frac{2^n-8}{12} + \frac{2^n-8}{720}\right)$$

□

6 Conclusion

One can count the number of Boolean functions with multiplicative complexity M by exhaustively listing the equivalence classes with multiplicative complexity M and finding the size of each class. However, already when multiplicative complexity is three, it is hard to list the equivalence classes exhaustively. For functions on $n = 4$, one can show that $[x_1x_2x_3x_4]$,$[x_1x_2x_3x_4 + x_1x_2]$ and $[x_1x_2x_3x_4 + x_1x_2 + x_3x_4]$ are the only equivalence classes with multiplicative complexity 3. For $n = 5$, the exhaustive list of classes with multiplicative complexity 3 is not known. Turan and Peralta [11] showed that the number of such classes is between 16 and 24. For $n = 6$, there are 2497 equivalence classes having degree at most 4. This provides an upper bound on the number of equivalence classes that can be computed by circuits with three AND gates, since some of these might have multiplicative complexity 4 or more.

References

1. Michael J. Fischer and René Peralta. Counting predicates of conjunctive complexity one. *Yale Technical Report 1222*, February 2002.
2. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325. ACM, 2012.

3. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.
4. Joan Boyar, Ivan Damgård, and René Peralta. Short non-interactive cryptographic proofs. *J. Cryptology*, 13(4):449–472, 2000.
5. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.
6. Nicolas Courtois, Daniel Hulme, and Theodosis Mourouzis. Solving circuit optimisation problems in cryptography and cryptanalysis, 2011.
7. Joan Boyar, Magnus Find, and René Peralta. Four measures of nonlinearity. In Paul G. Spirakis and Maria J. Serna, editors, *CIAC*, volume 7878 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 2013.
8. Magnus Gausdal Find. On the complexity of computing two nonlinearity measures. In Edward A. Hirsch, Sergei O. Kuznetsov, Jean-Éric Pin, and Nikolay K. Vereshchagin, editors, *Computer Science - Theory and Applications - 9th International Computer Science Symposium in Russia, CSR 2014, Moscow, Russia, June 7-11, 2014. Proceedings*, volume 8476 of *Lecture Notes in Computer Science*, pages 167–175. Springer, 2014.
9. Joan Boyar, René Peralta, and Denis Pochuev. On the multiplicative complexity of Boolean functions over the basis ($\wedge$, $\oplus$, 1). *Theor. Comput. Sci.*, 235(1):43–57, 2000.
10. Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In Paola Festa, editor, *SEA*, volume 6049 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 2010.
11. Meltem Sönmez Turan and René Peralta. The multiplicative complexity of boolean functions on four and five variables. In Thomas Eisenbarth and Erdinç Öztürk, editors, *Lightweight Cryptography for Security and Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers*, volume 8898 of *Lecture Notes in Computer Science*, pages 21–33. Springer, 2014.
12. Pavol Zajac and Matus Jokay. Multiplicative complexity of bijective 4 x 4 s-boxes. *Cryptography and Communications*, 6(3):255–277, 2014.
13. Roland Mirwald and Claus-Peter Schnorr. The multiplicative complexity of quadratic Boolean forms. *Theor. Comput. Sci.*, 102(2):307–328, 1992.
14. Claus-Peter Schnorr. The multiplicative complexity of Boolean functions. In *AAECC*, pages 45–58, 1988.
15. Elwyn R. Berlekamp and Lloyd R. Welch. Weight distributions of the cosets of the (32, 6) Reed-Muller code. *IEEE Transactions on Information Theory*, 18(1):203–207, 1972.
16. Joanne Elizabeth Fuller. *Analysis of affine equivalent boolean functions for cryptography*. PhD thesis, Queensland University of Technology, 2003.
17. James A. Maiorana. A classification of the cosets of the Reed-Muller code R(1,6). *Mathematics of Computation*, 57(195):403–414, 1991.
18. An Braeken, Yuri L. Borissov, Svetla Nikova, and Bart Preneel. Classification of Boolean functions of 6 variables or less with respect to some cryptographic properties. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 324–334. Springer, 2005.
19. Xiang-Dong Hou. AGL (m, 2) acting on R (r, m)/R (s, m). *Journal of Algebra*, 171(3):927–938, 1995.