The Multiplicative Complexity of Boolean Functions on Four and Five Variables

Meltem Sönmez Turan[†] [‡], René Peralta[†]

[†] National Institute of Standards and Technology, Gaithersburg, MD
meltem.turan@nist.gov, rene.peralta@nist.gov
[‡] Dakota Consulting Inc., Silver Spring, MD

Abstract. A generic way to design lightweight cryptographic primitives is to construct simple rounds using small nonlinear components such as 4x4 S-boxes and use these iteratively (e.g., PRESENT [1] and SPONGENT [2]). In order to efficiently implement the primitive, efficient implementations of its internal components are needed. Multiplicative complexity of a function is the minimum number of AND gates required to implement it by a circuit over the basis (AND, XOR, NOT). It is known that multiplicative complexity is exponential in the number of input bits n. Thus it came as a surprise that circuits for all 65 536 functions on four bits were found which used at most three AND gates [3]. In this paper, we verify this result and extend it to five-variable Boolean functions. We show that the multiplicative complexity of a Boolean function with five variables is at most four.

Keywords: Affine transformation, Boolean functions, Circuit complexity, Multiplicative complexity

1 Introduction

One of the important challenges in lightweight cryptography is to find efficient implementations of secure cryptographic primitives. Many attempts [4–7] have been done to improve the efficiency of the block cipher AES, in order to fit the implementations in resource-constrained devices. However, even the best implementations of AES are usually too big for constrained devices. A generic way to design dedicated lightweight cryptographic primitives is to construct simple rounds using small nonlinear components such as 4x4 S-boxes and iterate (this is done in, for example, PRESENT [1] and SPONGENT [2]). In order to efficiently implement the primitive, efficient circuits for the internal components are needed. Efficiency of the implementations can be assessed using different metrics such as area, power, and energy requirements. These metrics are strongly related to the number of logic gates used to implement the primitive.

Gate complexity is defined as the minimum number of 2-input logic gates required to implement the primitive in a circuit. *Multiplicative complexity* (MC) is another complexity measure, which is defined as the minimum number of AND gates required to implement the primitive by a circuit over the basis (AND, XOR, NOT), with an unlimited number of NOT and XOR gates. This is the same as the number of multiplications needed for straight-line programs that do arithmetic modulo 2.

Finding the gate complexity or the multiplicative complexity of a given function is computationally intractable, even for functions with a small number of variables. In 2006, Saarinen [8] published the gate complexity distribution of 4-variable Boolean functions. In 2010, Boyar et al. [9] proposed a two-stage heuristic method to minimize the gate complexity of Boolean circuits. In the first state, the heuristics minimizes the number of AND gates required to implement the circuit, 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).

Apart from possibly increasing the efficiency of the implementations, minimizing the number of AND gates provides a tool for the cryptanalysis of the primitives. For example, according to [10, 11], functions with low multiplicative complexity are more vulnerable to algebraic attacks than those with high multiplicative complexity. This is an important observation, as a function representation tends to hide its true multiplicative complexity (e.g. consider a random polynomial on five variables over $GF(2)$, it is hard to see how it could possibly be computed using only four multiplications).

Also, a relationship between collision resistance of a hash function and multiplicative complexity is provided in [12]. Courtois et al. [10] argued that minimizing the number of AND gates is important to prevent against side channel attacks such as differential power analysis. Finally, we point out that the number and position of AND gates in a circuit is the main determinant of whether the function can be used in the context of protocols that use homomorphic encryption.

Multiplicative complexity of a randomly selected n-variable Boolean function is at least $2^{n/2} - O(n)$ [13]. Exhaustive study of the distribution of multiplicative complexity can only be done for very small values of n. In 2013, we were surprised to find circuits for all 65 536 functions on four bits which used at most three AND gates [3]. Boyar et al. [12] conjecture that some five-bit Boolean functions have multiplicative complexity five. It is shown in [13] that there are at most $2^{k^2+2k+2kn+n+1}$ many n-variable Boolean functions that can be generated using k AND gates. This is a counting argument and it is not known how tight it is. Using this bound, it is easy to see that there exist eight-bit Boolean functions with multiplicative complexity of at least eight. So, it is an open question whether there exists n-bit Boolean functions with multiplicative complexity n, for $n = 5, 6, 7$. Lower bounds are extremely difficult: no specific n-variable function has yet been proven to have multiplicative complexity larger than $n - 1$ for any n.

In this paper, we focus on the multiplicative complexity of four and five-variable Boolean functions. Using the fact that multiplicative complexity is affine invariant, we first provide a succinct proof (i.e. one that does not list all circuits) that the multiplicative complexity of four-variable Boolean functions is at most three. Then, we extend the result for five-variable Boolean functions and show that the conjecture given in [12] is false: any five-bit Boolean functions can be implemented with at most four AND gates.

The organization of this paper is as follows. Section 2 provides definitions and some known facts about Boolean functions, affine equivalence, and multiplicative complexity. Section 2 focuses on affine invariance of multiplicative complexity and provides results for four and five-variable Boolean functions. Section 4 concludes the paper.

2 Preliminaries

2.1 Boolean Functions

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 B_n be the set of n-variable Boolean functions. Note that $|B_n| = 2^{2^n}$. Boolean functions have various representations, some are canonical, some not so. The list of output values for each n-bit input $T_f = (f(0, \ldots, 0), f(0, \ldots, 0, 1), \ldots, f(1, \ldots, 1))$ is called the *truth table* representation of f. The number of ones in T_f is called the *Hamming weight* of f, denoted $wt(f)$. The *Hamming distance* between two Boolean functions $f, g \in B_n$, denoted $d(f, g)$, is $wt(f + g)$. That is, the Hamming distance is the cardinality of the set $\{x \in \mathbb{F}_2^n | f(x) \neq g(x)\}$.

Boolean functions are also represented uniquely by the multivariate polynomial called *algebraic normal form* (ANF)

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

where $x^u = x_1^{u_1} x_2^{u_2} \cdots x_n^{u_n}$ is a *monomial* composed of the variables for which $u_i = 1$ and $a_u \in \mathbb{F}_2$. This is also called the *Zhegalkin polynomial* of f. The degree of a Boolean function, denoted d_f, is the degree of the highest-degree monomial in its ANF representation.

Let A_n be the set of n-variable affine functions, i.e., functions having degree at most one. The nonlinearity of a Boolean function f, denoted N_f, is the minimum Hamming distance of f to all affine functions, that is $N_f = \min_{g \in A_n} d(f, g)$. The nonlinearity of an n-variable Boolean function is upper bounded by $2^{n-1} - 2^{n/2-1}$.

2.2 Affine Equivalence

Affine transformations of a function f allow for linear operations to inputs and output of f. There are several definitions in the literature, and they are not all equivalent to each other. Here, we will use a definition from Berlekamp and Welch [14].

Definition 1. *An* affine transformation *from g to f in B_n is a mapping of the form $f(x) = g(Ax + a) + b \cdot x + c$, where*

- *A is a non-singular $n \times n$ matrix over $\mathbb{F}_2$;*
- *x, a are column vectors over $\mathbb{F}_2$;*
- *b is a row vector over $\mathbb{F}_2$; and*
- *$c \in F_2$.*

It is not hard to prove that this defines an equivalence relation on the set of n-variable Boolean functions. Two functions f, g are *affine equivalent* if there exist affine transformations between them. Affine equivalent Boolean functions are said to be in the same equivalence class.

An algorithm to check whether two functions are equivalent is given in [15]. This algorithm also outputs an affine transformation between the input functions, if one exists. The equivalence classes can be constructed using exhaustive search for small values of n. For example, the Boolean functions with three variables can be partitioned into three equivalence classes, and a representative from each class can be given as $\{x_1, x_1x_2$ and $x_1x_2x_3\}$. The classification of five-variable Boolean functions was done in 1972 by Berlekamp and Welch [14]. Maiorana [16] proved that the number of classes in B_6 is 150 357. This was independently verified by Fuller [15] and by Braeken et al. [17]. For $n = 7$, Hou [18] determined that there are 63 379 147 320 777 408 548 equivalence classes. Since the size of B_n is doubly exponential, and each equivalence class can contain only an exponential number of functions, the number of equivalence classes is asymptotically exponential in n (see Table 1).

Table 1. Number of equivalence classes for $n \leq 7$

n	$\|B_n\|$	# of Equivalence Classes
3	2^8	3
4	2^{16}	8
5	2^{32}	48
6	2^{64}	150 357
7	2^{128}	63 379 147 320 777 408 548 (this is between 2^{65} and 2^{66})

Some cryptographic measures such as nonlinearity, algebraic degree, and algebraic immunity remain unchanged after applying an affine transformation. Such measures are said to be *affine invariant* [19]. Braeken et al. [17] studied the classification of Boolean functions with respect to various cryptographic properties. Uyan [20] analyzed the Boolean functions with respect to the Walsh Spectrum using equivalence classes.

2.3 Multiplicative Complexity

The *multiplicative complexity* $C_\wedge(f)$ of a Boolean function is the minimum number of multiplications (AND-$\wedge$ gates) that are sufficient to evaluate the function over the basis (AND, XOR, NOT). The multiplicative complexity of functions having degree d is at least $d-1$ [21]. This bound is called the *degree bound*. Calculating the multiplicative complexity of a randomly selected Boolean function is hard even for small values of n [1].

3 Multiplicative Complexity of Boolean Functions

Multiplicative complexity is invariant under affine transformations. Thus, to bound the multiplicative complexity distribution of n-bit Boolean functions, it is enough to bound the multiplicative complexity of a single function from each equivalence class. Fuller presents an algorithm to find a representative from each equivalence class [15]. Her method is practical for values of n up to six.

In order to find a circuit for $f \in B_n$ with a small number of AND gates, we use the following approach.

1. Precomputation
   (a) Using the algorithm given in [15], find a "simple" (e.g., one that has a small number of monomials in its ANF) representative for each equivalence class in B_n.
   (b) For each representative, find a circuit which is efficient with respect to multiplicative complexity.
2. Find the equivalence class C_f of f.
3. Find the affine transformation from f^*, the representative of C_f, to f using the algorithm given in [15].
4. Apply the affine transformation to the circuit for f^*. This yields a circuit for f. The circuit will be efficient with respect to multiplicative complexity. Note that, if the circuit found for f^* is not optimal, this still yields an upper bound on the multiplicative complexity of f.

In the following subsections, we used this approach to bound the multiplicative complexity of all Boolean functions on four and on five variables. The approach becomes impractical as the number of variables increases due to the following reasons; (i) the number of equivalence classes increases exponentially with the number of variables; (ii) finding an affine transformation from f^* to f gets harder; and (iii) constructing circuits that are optimal with respect to multiplicative complexity gets harder.

[1] Lest the reader think this easy, he/she may attempt to compute the function $f(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3x_4x_5 + x_1x_2x_3 + x_1x_2x_4 + x_2x_3x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_4 + x_3x_4$ using only four AND gates.

3.1 $n = 4$

There are eight equivalence classes of B_4, with representatives $\{x_1,\ x_1x_2,\ x_1x_2 + x_3x_4,\ x_1x_2x_3,\ x_1x_2x_3+x_1x_4,\ x_1x_2x_3x_4,\ x_1x_2x_3x_4+x_1x_2,\ x_1x_2x_3x_4+x_1x_2+x_3x_4\}$. The representatives are simple enough that optimal (with respect to multiplicative complexity) circuits can be easily constructed. Table 2 provides a circuit with optimal number of AND gates for each of these representatives. The optimality of the circuits follows from the degree bound for seven (out of eight) of the equivalence classes. For example, according to the degree bound, the multiplicative complexity of $x_1x_2x_3x_4 + x_1x_2$, is at least three. Optimality of the third class, which is a sum of quadratic functions, cannot be verified by the degree bound. That two AND gates are needed seems obvious, as the two quadratic terms have no common variables. For a formal proof, see Mirwald and Schnorr [22], which shows that the multiplicative complexity of a quadratic function of the form $\sum_{i=1}^{k} x_{2i-1}x_{2i}$ is k.

It is easy to find the equivalence class of a given function $f \in B_4$ by checking the nonlinearity and degree of f, since the nonlinearity and the degree pair (N_f, d_f) of the representatives are distinct (See Table 2).

Table 2. Equivalence classes of B_4

Class	Representatives	Implementation	MC	(N_f, d_f)	# Functions
1	x_1	$f = x_1$	0	(0,1)	32
2	x_1x_2	$t_1 = x_1 \wedge x_2$	1	(4,2)	1120
3	$x_1x_2 + x_3x_4$	$t_1 = x_1 \wedge x_2$ $t_2 = x_3 \wedge x_4$ $f = t_1 \oplus t_2$	2	(6,2)	896
4	$x_1x_2x_3$	$t_1 = x_1 \wedge x_2$ $f = t_1 \wedge x_3$	2	(2,3)	3840
5	$x_1x_2x_3 + x_1x_4$	$t_1 = x_2 \wedge x_3$ $t_2 = t_1 \oplus x_4$ $f = t_2 \wedge x_1$	2	(4,3)	26880
6	$x_1x_2x_3x_4$	$t_1 = x_1 \wedge x_2$ $t_2 = t_1 \wedge x_3$ $f = t_2 \wedge x_4$	3	(1,4)	512
7	$x_1x_2x_3x_4 + x_1x_2$	$t_1 = x_1 \wedge x_2$ $t_2 = t_1 \wedge x_3$ $t_3 = t_2 \wedge x_4$ $f = t_3 \oplus t_1$	3	(3,4)	17920
8	$x_1x_2x_3x_4 + x_1x_2 + x_3x_4$	$t_1 = x_1 \wedge x_2$ $t_2 = x_3 \wedge x_4$ $t_3 = t_1 \wedge t_2$ $t_4 = t_3 \oplus t_1$ $f = t_4 \oplus t_2$	3	(5,4)	14336

Example 1. Let $f = 1 + x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_4 + x_3x_4 + x_1x_2x_3 + x_2x_3x_4 + x_1x_3x_4 + x_1x_2x_3x_4$. In order to find a circuit for f with minimum number of AND gates, we

first need to find its equivalence class. Since (N_f, degree) = (3,4), f belongs to the seventh equivalence class with representative $f^* = x_1x_2x_3x_4 + x_1x_2$. Then, we need to obtain the affine transformation between f and f^*. Using the algorithm in [15], the transformation is obtained as

$$f(x) = f^*(\begin{pmatrix} 1\,0\,1\,1 \\ 1\,1\,1\,1 \\ 0\,1\,1\,0 \\ 1\,0\,1\,0 \end{pmatrix} x \oplus (0\ 0\ 0\ 0)) \oplus (0\ 1\ 1\ 0)x \oplus 1. \tag{2}$$

According to this transformation, input variables are transformed as follows; $x_1 \to x_1 + x_3 + x_4$, $x_2 \to x_1 + x_2 + x_3 + x_4$, $x_3 \to x_2 + x_3$, and $x_4 \to x_1 + x_3$, and the affine shift is equivalent to XORing $x_2 + x_3 + 1$ to f. Given an efficient circuit for $x_1x_2x_3x_4 + x_1x_2$, an efficient circuit for f can be found as is shown in Table 3. The first four equations in the implementation of f are due to the linear transformations of input variables, whereas the last three equations corresponds to the affine shift.

Table 3. Optimal circuit for f in terms of number of AND gates.

f^*	f
	$t_1 = x_1 \oplus x_3$
	$t_2 = t_1 \oplus x_4$
	$t_3 = t_2 \oplus x_2$
	$t_4 = x_2 \oplus x_3$
$t_1 = x_1 \wedge x_2$	$t_5 = t_2 \wedge t_3$
$t_2 = t_1 \wedge x_3$	$t_6 = t_5 \wedge t_4$
$t_3 = t_2 \wedge x_4$	$t_7 = t_6 \wedge x_3$
$f* = t_3 \oplus t_1$	$t_8 = t_7 \oplus t_5$
	$t_9 = t_8 \oplus t_3$
	$t_{10} = t_9 \oplus t_4$
	$f = t_{10} \oplus 1$

□

3.2 $n = 5$

Berlekamp and Welsh [14] provided the representatives of the 48 equivalence classes for $n = 5$. Table 5 in the Appendix provides the circuits to implement the representatives of each equivalence class. Most of the representatives are simple enough that the optimal circuits are found trivially. The circuits corresponding to the representatives of the classes 14, 18, 26, 37, 44, 45, 46, 47, 48 are obtained using the heuristic provided in [23]. Optimality of thirty circuits (out of 48) can be verified using the degree bound.

To find the equivalence class of a given function $f \in B_5$, the nonlinearity and degree of f can be utilized. Table 2 provides a classification of the equivalence classes based on the nonlinearity of degrees. As seen from the table, for 10 of the equivalence classes, the degree and nonlinearity uniquely determines the class representative. Moreover, checking degree and nonlinearity of an input function significantly reduces the possible number of equivalence classes.

Table 4. The distribution of equivalence classes of B_5 according to degree and nonlinearity. The functions are written in an abbreviated notation. For example, 123+145 indicates the representative of the form $x_1x_2x_3 + x_1x_4x_5$.

	d_f				
N_f	1	2	3	4	5
0	1	-	-	-	-
1	-	-	-	-	12345
2	-	-	-	2345	-
3	-	-	-	-	12345+123
4	-	-	123	2345+123	-
5	-	-	-	-	12345+123+12 12345+123+145+12
6	-	-	123+145	2345+23 2345+123+12 2345+123+145+45	-
7	-	-	-	-	12345+12 12345+123+14 12345+123+145 12345+123+145+23 12345+123+145+45+23
8	-	12	123+14 123+145+23	2345+12 2345+123+24 2345+123+14 2345+123+145 2345+123+12+45	-
9	-	-	-	-	12345+123+45 12345+123+12+45 12345+123+12+34 12345+123+145+24 12345+123+145+24+23 12345+123+145+35+24
10	-	-	123+45 123+145+24	2345+23+45 2345+12+34 2345+123+45 2345+123+12+34 2345+123+14+35 2345+123+145+24+45 2345+123+145+35+24	-
11	-	-	-	-	12345+12+34 12345+123+14+25 12345+123+145+35+24+23 12345+123+145+45+35+24+23
12	-	12+34	123+14+25 123+145+23+24+35	2345+123+24+35	-

4 Conclusion

We studied the multiplicative complexity of Boolean functions with four and five variables. For four variables, we confirmed that the multiplicative complexity is at most three by producing circuits for a representative of each of the eight equivalence classes. We knew this was true because one of us has posted circuits for all 2^{16}, each using at most three AND gates [3]. Those circuits are also optimized for total number of gates: it turns out that no more than seven XOR gates are needed by AND-optimal circuits.

For five variables, we disproved the conjecture that there exists Boolean functions with multiplicative complexity five. We are in the process of extending this work to six-variable Boolean functions. This will most likely require a computer proof, as there are 150 357 equivalence classes.

Acknowledgments

We thank Çağdaş Çalık, Joan Boyar, and Magnus Find for helpful discussions and suggestions. We also thank our colleagues Yi-Kai Liu, Ray Perlner, Lily Chen, and the anonymous reviewers for their useful comments.

References

1. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
2. Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. SPONGENT: The Design Space of Lightweight Cryptographic Hashing. *IEEE Trans. Computers*, 62(10):2041–2053, 2013.
3. René Peralta. Circuit minimization work, http://cs-www.cs.yale.edu/homes/peralta/circuitstuff/cmt.html, january 2014.
4. Martin Feldhofer, Johannes Wolkerstorfer, and Vincent Rijmen. AES implementation on a grain of sand. *Information Security, IEE Proceedings*, 152(1):13 – 20, 2005.
5. Panu Hamalainen, Timo Alho, Marko Hannikainen, and Timo D. Hamalainen. Design and implementation of low-area and low-power AES encryption hardware core. In *Proceedings of the 9th EUROMICRO Conference on Digital System Design*, DSD '06, pages 577–583, Washington, DC, USA, 2006. IEEE Computer Society.
6. Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6632 of *Lecture Notes in Computer Science*, page 69. Springer, 2011.
7. Joan Boyar and Rene Peralta. A small depth-16 circuit for the AES S-Box. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer Berlin Heidelberg, 2012.
8. Markku-Juhani Olavi Saarinen. Chosen-IV statistical attacks on estream ciphers. In Manu Malek, Eduardo Fernández-Medina, and Javier Hernando, editors, *SECRYPT*, pages 260–266. INSTICC Press, 2006.

9. 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.
10. Nicolas Courtois, Daniel Hulme, and Theodosis Mourouzis. Solving circuit optimisation problems in cryptography and cryptanalysis, 2011.
11. Nicolas Courtois, Daniel Hulme, and Theodosis Mourouzis. Multiplicative complexity and solving generalized brent equations with SAT solvers. *In COMPUTATION TOOLS 2012, The Third International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking*, pages 22–27, 2012.
12. 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.
13. 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.
14. 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.
15. Joanne Elizabeth Fuller. *Analysis of affine equivalent boolean functions for cryptography.* PhD thesis, Queensland University of Technology, 2003.
16. James A. Maiorana. A classification of the cosets of the Reed-Muller code R(1,6). *Mathematics of Computation*, 57(195):403–414, 1991.
17. 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.
18. Xiang-Dong Hou. AGL (m, 2) acting on R (r, m)/R (s, m). *Journal of Algebra*, 171(3):927–938, 1995.
19. Claude Carlet. Boolean functions for cryptography and error correcting codes. In Y. Crama and P.L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science and Engineering*, chapter 8. Cambridge Univ. Press, Cambridge, UK, 2010.
20. Erdener Uyan. *Analysis of Boolean Functions with respect to Walsh Spectrum.* PhD thesis, Middle East Technical University, 2013.
21. Claus-Peter Schnorr. The multiplicative complexity of Boolean functions. In *AAECC*, pages 45–58, 1988.
22. Roland Mirwald and Claus-Peter Schnorr. The multiplicative complexity of quadratic Boolean forms. *Theor. Comput. Sci.*, 102(2):307–328, 1992.
23. Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *J. Cryptology*, 26(2):280–312, 2013.

Appendix

Table 5: Circuits for $n = 5$.

Class	Representative	Circuit	MC
1	2345	$t_1 = 2 \wedge 3$, $t_2 = 4 \wedge 5$, $f = t_1 \wedge t_2$	3
2	2345⊕12	$t_1 = 3 \wedge 4$, $t_2 = t_1 \wedge 5$, $t_3 = t_2 \oplus 1$, $f = 2 \wedge t_3$	3
3	2345⊕23	$t_1 = 2 \wedge 3$, $t_2 = t_1 \wedge 4$, $t_3 = t_2 \wedge 5$, $f = t_1 \oplus t_3$	3
4	2345⊕23 ⊕ 45	$t_1 = 2 \wedge 3$, $t_2 = 4 \wedge 5$, $t_3 = t_1 \wedge t_2$, $t_4 = t_3 \oplus t_1$ $f = t_4 \oplus t_2$	3
5	2345⊕12⊕34	$t_1 = 3 \wedge 4$, $t_2 = t_1 \wedge 5$, $t_3 = t_2 \oplus 1$, $t_4 = 2 \wedge t_3$, $f = t_4 \oplus t_1$	3
6	2345⊕123	$t_1 = 2 \wedge 3$, $t_2 = 4 \wedge 5$, $t_3 = t_2 \oplus 1$, $f = t_3 \wedge t_1$	3
7	2345⊕123⊕12	$t_1 = 4 \wedge 5$, $t_2 = 1 \oplus t_1$, $t_3 = 3 \wedge t_2$, $t_4 = 1 \oplus t_3$ $f = 2 \wedge t_4$	3
8	2345⊕123⊕24	$t_1 = 4 \wedge 5$, $t_2 = 1 \oplus t_1$, $t_3 = 3 \wedge t_2$, $t_4 = 4 \oplus t_3$ $f = 2 \wedge t_4$	3
9	2345⊕123⊕14	$t_1 = 2 \wedge 3$, $t_2 = 4 \wedge 5$, $t_3 = 1 \oplus t_2$, $t_4 = t_1 \wedge t_3$ $f = t_4 \oplus t_2$	3
10	2345⊕123⊕45	$t_1 = 2\wedge3$,$t_2 = 4\wedge5$, $t_3 = t_2\oplus1$, $t_4 = t_3 \wedge t_1$, $f = t_4 \oplus t_2$	3
11	2345⊕123⊕12⊕34	$t_1 = 2\wedge4$, $t_2 = t_1\wedge5$, $t_3 = 1\wedge2$, $t_4 = t_2 \oplus t_3$, $t_5 = t_4\oplus4$,$t_6 = t_5\wedge3$, $f = t_6 \oplus t_3$	≤ 4
12	2345⊕123⊕14⊕35	$t_1 = 4 \wedge 5$, $t_2 = 1 \oplus t_1$, $t_3 = 2 \wedge t_2$, $t_4 = 5 \oplus t_3$, $t_5 = 3 \wedge t_4$, t_6=$1\wedge4$, $f = t_5 \oplus t_6$	≤ 4
13	2345⊕123⊕12⊕45	$t_1 = 2 \wedge 3$, $t_2 = 1 \wedge 2$, $t_3 = 4 \wedge 5$, $t_4 = t_2 \oplus t_3$ $t_5 = t_1 \wedge t_4$, $f = t_5 \oplus t_4$	≤ 4
14	2345⊕123⊕24⊕35	$t_1 = 4 \wedge 5$, $t_2 = 1 \oplus t_1$, $t_3 = 2 \oplus 3$, $t_4 = 1 \oplus t_3$, $t_5 = t_4 \oplus t_1$, $t_6 = t_2 \wedge t_5$, $t_7 = 4 \oplus t_6$, $t_8 = 2 \wedge t_7$, $t_9 = 3 \wedge 5$, $f = t_8 \oplus t_9$	≤ 4
15	2345⊕123⊕145	$t_1 = 2 \wedge 3$, $t_2 = 1 \oplus t_1$, $t_3 = 4 \wedge 5$ $t_4 = 1 \oplus t_3$ $t_5 = t_2 \wedge t_4$, $f = 1 \oplus t_5$	3
16	2345⊕123⊕145⊕45	$t_1 = 2 \wedge 3$, $t_2 = 1 \oplus t_1$, $t_3 = 4 \wedge 5$, $t_4 = 1 \oplus t_3$ $t_5 = t_2 \wedge t_4$, $f = t_5 \oplus t_4$	3
17	2345⊕123⊕145⊕24⊕45	$t_1 = 4 \wedge 5$, $t_2 = 1 \oplus t_1$, $t_3 = 2 \wedge 3$, $t_4 = t_3 \oplus t_1$, $t_5 = t_2 \wedge t_4$, $t_6 = 2 \wedge 4$, $f = t_5 \oplus t_6$	≤ 4
18	2345⊕123⊕145⊕35⊕24	$t_1 = 2 \wedge 3$, $t_2 = 1 \oplus t_1$, $t_3 = t_4 \wedge t_5$, $t_4 = t_1 \oplus t_3$, $t_5 = t_2 \wedge t_4$, $t_6 = 2 \oplus 5$, $t_7 = 3 \oplus 4$, $t_8 = t_6 \wedge t_7$, t_9 =$t_3 \oplus t_8$, $f = t_5 \oplus t_9$	≤ 4
19	123	$t_1 = 1\wedge2$, $f = t_1\wedge3$	2
20	123⊕45	$t_1 = 1\wedge2$, $t_2 = t_1\wedge3$, $t_3 = 4\wedge5$, $f = t_2 \oplus t_3$	3
21	123⊕14	$t_1 = 2\wedge3$, $t_2 = t_1\oplus4$, $f = 1 \wedge t_2$	2
22	123⊕14⊕25	$t_1 = 2\wedge3$, $t_2 = t_1\oplus4$,$t_3 = t_2\wedge1$, $t_4 = 2\wedge5$ $f = t_3 \oplus t_4$	3
23	123⊕145	$t_1 = 2\wedge3$, $t_2 = 4\wedge5$, $t_3 = t_1 \oplus t_2$, $f = 1 \wedge t_3$	3

Continued on next page

5 – continued from previous page

Class	Representative	Circuit	MC
24	123⊕145⊕23	$t_1 = 2{\wedge}3$, $t_2 = 4{\wedge}5$, $t_3 = t_1 \oplus t_2$, $t_4 = 1 \wedge t_3$ $f = t_4 \oplus t_1$	3
25	123⊕145⊕24	$t_1 = 2{\wedge}3$, $t_2 = 4{\wedge}5$,$t_3 = t_1 \oplus t_2$,$t_4 = 1 \wedge t_3$ $t_5 = 2{\wedge}4$,$f = t_4 \oplus t_5$	≤ 4
26	123⊕145⊕23⊕24⊕35	$t_1 = 1 \wedge 5$, $t_2 = 2 \oplus t_1$, $t_3 = 1 \oplus 3$, $t_4 = 3 \wedge t_3$ $t_5 = 4 \oplus t_4$, $t_6 = t_2 \wedge t_5$, $t_7 = 3 \wedge 5$, $f = t_6 \oplus t_7$	≤ 4
27	12	$f = 1{\wedge}2$	1
28	12⊕34	$t_1 = 1{\wedge}2$, t_2=$3{\wedge}4$, $f = t_1 \oplus t_2$	2
29	1	f =1	0
30	12345	$t_1 = 1{\wedge}2$, $t_2 = t_1{\wedge}3$,$t_3 = t_2{\wedge}4$, $f = t_3{\wedge}5$	4
31	12345⊕12	$t_1 = 1{\wedge}2$, $t_2 = t_1{\wedge}3$, $t_3 = t_2{\wedge}4$,$t_4 = t_3{\wedge}5$ $f = t_4 \oplus t_1$	4
32	12345⊕12⊕34	$t_1 = 1{\wedge}2$, t_2=$3{\wedge}4$, $t_3 = t_1 \wedge t_2$, $t_4 = t_3{\wedge}5$ $t_5 = t_4 \oplus t_1$, $f = t_5 \oplus t_2$	4
33	12345⊕123	$t_1 = 1{\wedge}2$, $t_2 = t_1{\wedge}3$, $t_3 = t_2{\wedge}4$, $t_4 = t_3{\wedge}5$ $f = t_4 \oplus t_2$	4
34	12345⊕123⊕12	$t_1 = 1{\wedge}2$, $t_2 = t_1{\wedge}3$, $t_3 = t_2{\wedge}4$, $t_4 = t_3{\wedge}5$ $t_5 = t_4 \oplus t_1$, $f = t_5 \oplus t_2$	4
35	12345⊕123⊕14	$t_1 = 2{\wedge}3$, $t_2 = 1{\wedge}4$,$t_3 = t_2{\wedge}5$, $t_4 = t_3{\oplus}1$ $t_5 = t_1 \wedge t_4$, $f = t_2 \oplus t_5$	4
36	12345⊕123⊕45	$t_1 = 1{\wedge}2$, $t_2 = t_1{\wedge}3$,$t_3 = 4{\wedge}5$, $t_4 = t_2 \wedge t_3$ $t_5 = t_4 \oplus t_2$, $f = t_5 \oplus t_3$	4
37	12345⊕123⊕14⊕25	$t_1 = 1 \oplus 4$, $t_2 = 3 \wedge t_1$, $t_3 = 4 \oplus t_2$, $t_4 = 3 \oplus 4$ $t_5 = 2 \oplus t_4$, $t_6 = 2 \wedge 5$, $t_7 = 3 \oplus t_6$, $t_8 = t_5 \wedge t_7$ $t_9 = 1 \oplus t_8$, $t_{10} = t_3 \wedge t_9$, $f = t_{10} \oplus t_6$	4
38	12345⊕123⊕12⊕45	$t_1 = 1{\wedge}2$, $t_2 = t_1{\wedge}3$, $t_3 = 4{\wedge}5$,$t_4 = t_2 \wedge t_3$ $t_5 = t_4 \oplus t_1$, $t_6 = t_5 \oplus t_2$, $f = t_6 \oplus t_3$	4
39	12345⊕123⊕12⊕34	$t_1 = 1{\wedge}2$, $t_2 = 4{\wedge}5$, $t_3 = t_1 \wedge t_2$, $t_4 = t_3 \oplus t_1$ $t_5 = t_4{\oplus}4$, t_6=$3{\wedge}t_5$, $f = t_1 \oplus t_6$	4
40	12345⊕123⊕145	$t_1 = 2{\wedge}3$, $t_2 = 4{\wedge}5$, $t_3 = t_1 \wedge t_2$, $t_4 = t_1 \oplus t_2$ $t_5 = t_4 \oplus t_3$, $f = 1 \wedge t_5$	4
41	12345⊕123⊕145⊕12	$t_1 = 2{\wedge}3$, $t_2 = 4{\wedge}5$, $t_3 = t_1 \wedge t_2$,$t_4 = t_1 \oplus t_2$ $t_5 = t_4{\oplus}2$, $t_6 = t_5 \oplus t_3$, $f = 1 \wedge t_6$	4
42	12345⊕123⊕145⊕23	$t_1 = 2{\wedge}3$, $t_2 = 4{\wedge}5$, $t_3 = t_1 \wedge t_2$, $t_4 = t_3 \oplus t_1$ $t_5 = t_4 \oplus t_2$, $t_6 = 1 \wedge t_5$, $f = t_6 \oplus t_1$	4
43	12345⊕123⊕145⊕45 ⊕23	$t_1 = 2{\wedge}3$, $t_2 = 4{\wedge}5$, $t_3 = t_1 \wedge t_2$, $t_4 = t_3 \oplus t_1$ $t_5 = t_4 \oplus t_2$, $t_6 = 1 \wedge t_5$, $t_7 = t_6 \oplus t_1$, $f = t_7 \oplus t_2$	4
44	12345⊕123⊕145⊕24	$t_1 = 2 \oplus 3$, $t_2 = 1 \oplus t_1$, $t_3 = 3 \wedge t_2$, $t_4 = 4 \oplus t_3$ $t_5 = 1 \wedge 4$, $t_6 = 5 \wedge t_5$, $t_7 = 2 \oplus t_6$, $f = t_4 \wedge t_7$	4
45	12345⊕123⊕145⊕24 ⊕23	$t_1 = 2 \wedge 3$, $t_2 = 3 \oplus 4$, $t_3 = 1 \oplus t_2$, $t_4 = 1 \wedge 5$ $t_5 = 2 \oplus t_4$, $t_6 = 4 \wedge t_5$, $t_7 = t_3 \oplus t_6$, $t_8 = t_1 \wedge t_7$ f= $t_8 \oplus t_6$	4
46	12345⊕123⊕145⊕35⊕24	$t_1 = 4 \oplus 5$, $t_2 = 1 \oplus t_1$, $t_3 = 5 \wedge t_2$, $t_4 = 3 \oplus t_3$ $t_5 = 2 \oplus 3$, $t_6 = 4 \wedge 5$, $t_7 = 1 \oplus t_6$, $t_8 = 2 \wedge t_7$	4

Continued on next page

5 – continued from previous page

Class	Representative	Circuit	MC
		$t_9 = t_1 \oplus t_8$, $t_{10} = t_4 \wedge t_9$, $t_{11} = t_6 \oplus t_3$ $f = t_{10} \oplus t_{11}$	
47	12345⊕123⊕145⊕35⊕24 ⊕23	$t_1 = 1 \oplus 4$, $t_2 = 1 \wedge t_1$, $t_3 = 2 \oplus t_2$, $t_4 = 4 \oplus 5$ $t_5 = 1 \oplus 3$, $t_6 = 2 \oplus 3$, $t_7 = 1 \oplus t_6$, $t_8 = 5 \wedge t_7$ $t_9 = t_5 \oplus t_8$, $t_{10} = 3 \wedge t_9$, $t_{11} = t_4 \oplus t_{10}$ $t_{12} = t_3 \wedge t_{11}$, $f = t_{12} \oplus t_8$	4
48	12345⊕123⊕145⊕45⊕35 ⊕24⊕23	$t_1 = 2 \oplus 4$, $t_2 = 2 \oplus 3$, $t_3 = 2 \oplus 5$, $t_4 = t_2 \wedge t_3$ $t_5 = t_1 \oplus t_4$, $t_6 = 1 \oplus t_3$, $t_7 = 4 \oplus 5$, $t_8 = t_2 \oplus t_7$ $t_9 = 1 \oplus t_8$, $t_{10} = 1 \wedge 4$, $t_{11} = t_2 \oplus t_{10}$, $t_{12} = t_9 \wedge t_{11}$ $t_{13} = t_6 \oplus t_{12}$, $t_{14} = t_5 \wedge t_{13}$, $f = t_{14} \oplus t_{10}$	4