Improved Zero-Sum Distinguisher for SPONGENT-88

Shuxian Fan[1, a] and Ming Duan[1,2,b]

[1]Institute of Foreign Language of Luoyang, Department of Language Engineering, China

[2]Institute of information engineering, Chinese academy of sciences, Beijing China

[a]fanshuxian0907@163.com, [b]duanming@iee.ac.cn

Keywords: Zero-sum distinguisher, SPONGENT, higher-order differentials, PRESENT-type.

Abstract. SPONGENT is a hash function based on the sponge construction and uses the PRESENT-type permutation as the internal permutation. The lightweight edition of SPONGENT has sizes of 88, 128, 160, 224 and 256 bits and we focus on SPONGENT-88. In this paper, we combine the higher order differential attack and integral attack to construct an improved zero-sum distinguisher for 14-rounds SPONGENT-88 in computational complexity 280. The encouraging results can provide technical reference for further cryptanalysis of PRESENT-type algorithms.

Introduction

With the development and application of the AES, the need for new block cipher has been diminished. However, the excellent cipher is not quite suitable in the extremely constrained environment, such as the sensor networks and RIDF which require both security and hardware efficiency equally. Once this need appeared, the cryptographic community designed plenty of custom-built lightweight cryptographic algorithms to satisfy this need. For example, stream ciphers like Trivium[5], and Grain[7] as well as block ciphers like HIGHT[8], SPONGENT[3] and PRESENT[2] are all designed to meet the needs of lightweight cipher.

Since the end of the 20th century the integral attack has been an important cryptanalysis method. Its first target is an algorithm called Square which plays a very important role in the analysis of the AES. The higher-order differential analysis was proposed by Lai in 1994 and Knudsen presented the concept of high order differential attack systematically in 1995 and used it to analyze the Feistel type ciphers. Although the two methods have different theoretical basis, they both use a group of chosen plaintext to build a distinguisher and obtain a balanced state at the end of the distinguisher. To be more specific, they both firstly find some active bytes or bits and take all the possible values while the other bits stay constant and finally make all the ciphertexts or some bits xor be zero.

In this paper we combine the two methods to construct an improved zero-sum distinguisher for 14 rounds PONGENT-88 in computational complexity 2^{80}, which is 2^4 less than that of Dong etc.[6] and we believe that this method can also be implied in the other PRESENT-type ciphers.

The reminder of the paper is organized as follows. Section 2 describes the related basic knowledge of the analysis. Section 3 introduces the construction of SPONGENT-88 . Section 4 describes the steps of constructing the zero-sum distinguisher of the SPONGENT-88 in detail. In section 5, we conclude and point out the study direction in the future.

Preliminaries

Higher-Order Derivatives. The higher-order derivatives was presented by Lai in[11] and applied to attack KN cipher by Knudsen in [9]. We recall some basic notions needed in the analysis process.

Definition 1. Let $(S, +)$ and $(T, +)$ be *Abelian* groups. For a function $f: S \to T$, the derivative at a point $a_1 \in S$ is defined as

$$\Delta_{(a_1)} f(y) = f(y + a_1) - f(y).$$

The i-th derivative of f at $(a_1, a_2, ..., a_i)$ is then recursively defined as

$$\Delta_{(a_1,...,a_i)} f(y) = \Delta_{(a_i)}(\Delta_{(a_1,...,a_{i-1})} f(y)).$$

The high-order differential attack takes advantages of those properties to analyze the algorithms. One can calculate the upper bound of the algebraic degree of the attacked function denoted as d and choose d+1 bits as the active bits while the other bits are constant, then the balanced state would be gotten at the end of output round if the active bits take all the possible values.

Zero-Sum Distinguisher. The zero-sum property is a new type of distinguishing property which has been recently presented by Aumasson and Meier[1]. For a given function f(x), a zero-sum is that the set of inputs sum to zero as well as their images by f(x) also sum to zero. Such zero-sum properties can be seen as a generalization of multiset properties (as known as integral properties)[4,10]. This method has been used in the cryptanalysis of the SHA-3 candidates, such as Luffa, Hamsi and Keccak.

Definition 2. Let F be a function from F_2^n into F_2^m. A zero-sum for F of size K is a subset $\{x_1,...,x_k\} \subset F_2^n$ of elements which sum to zero and for which the corresponding images by F also sum to zero.

For example: $\sum_{i=1}^{K} x_i = \sum_{i=1}^{K} F(x_i) = 0$

SPONGENT

SPONGENT, proposed in the CHES 2011[3], is a hash function based on the sponge construction and uses the PRESENT type permutation as the internal permutation. The lightweight edition of SPONGENT has sizes of 88, 128, 160, 224 and 256 bits and we focus on SPONGENT-88.

Sponge construction is a new choice for the constructing of hash functions, which can be split into two parts: the absorbing and the squeezing. In the absorbing phase, r bits input message are xored into the first r bits of the state and then enter into a large b-bit permutation π_b until all the input message are processed. In the squeezing phase, the first r bits of the state are returned as the output and then enter into the same b-bit permutation until all the n needed bits are returned. The specific construction is showed in the figure 1.

r

c

absorbing squeezing

Figure 1

The permutation π_b of SPONGENT uses PRESENT structure which is a variant of SPN network.The S-box and permutation used in SPONGENT-88 are listed in table 1 and table2.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	E	D	B	0	2	1	4	F	7	A	8	5	9	C	3	6

Table 1

0	22	44	66	1	23	45	67	2	24	46
68	3	25	47	69	4	26	48	70	5	27
49	71	6	28	50	72	7	29	51	73	8
30	52	74	9	31	53	75	10	32	54	76
11	33	55	77	12	34	56	78	13	35	57
79	14	36	58	80	15	37	59	81	16	38
60	82	17	39	61	83	18	40	62	84	19
41	63	85	20	42	64	86	21	43	65	87

Table 2.

Improved Attack on SPONGENT-88

Firstly, we calculate the algebraic normal form of the *S*-box of SPONGENT-88. We use x_1, x_2, x_3, x_4 to denote the four input bits and y_1, y_2, y_3, y_4 to denote the output bits. The algebraic normal forms of the S-box of forward and backward direction are shown in table 3.

Output	Forward	Backward
y_1	$x_1x_2x_4 + x_1x_3x_4 + x_1x_3 + x_1x_4 + x_3x_4 + x_1 + x_2 + 1$	$x_2x_3 + x_2x_4 + x_3x_4 + x_1$
y_2	$x_1x_2x_3 + x_1x_4 + x_2 + x_3 + 1$	$x_2x_3x_4 + x_1x_3 + x_2x_3 + x_3x_4 + x_2 + x_3 + x_4$
y_3	$x_1x_2x_3 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_4 + 1$	$x_2x_3x_4 + x_1x_2 + x_2x_3 + x_2x_4 + x_3 + x_4 + 1$
y_4	$x_2x_3 + x_1 + x_3 + x_4$	$x_1x_2x_4 + x_1x_3x_4 + x_2x_4 + x_1 + x_2 + x_3 + 1$

Table 3

Then we build a 14 rounds zero-sum distinguisher of the SPONGENT-88 using the methods mentioned in section 2 to obtain the balanced state at the both ends of the rounds. We set the 7th round as the initial state so the 1st to the 6th rounds is the backward direction while the 7th to the 14th rounds is the forward direction.

Forward Rounds. We construct a 14 rounds Zero-Sum distinguisher and set the 7th round as the beginning round.

round	the upper bound algebraic degree of each bit
9	(1,0,0,0),(0,0,0,1),(0,0,0,0),(0,0,0,0),(0,0,1,0),(0,0,1,0),(0,0,0,0),(0,1,0,0), (0,0,0,0),(0,0,0,0),(1,0,0,0),(1,0,0,0) (0,0,0,1),(0,0,0,0),(0,0,0,0),(0,0,1,0), (0,0,1,0),(0,0,0,0)(0,1,0,0),(0,0,0,0),(0,0,0,0),(1,0,0,0)
10	(1,1,0,0),(1,1,0,1),(0,0,1,1),(1,0,0,1),(1,0,1,0),(0,1,1,1),(0,0,1,1),(0,1,0,0), (1,1,1,0),(0,1,1,0),(1,0,0,1),(1,1,0,0)(1,1,0,1),(0,0,1,1),(1,0,0,1),(1,0,1,0), (0,1,1,1),(0,0,1,1)(0,1,0,0),(1,1,1,0),(0,1,1,0),(1,0,0,1)
11	(2,3,2,2),(2,2,2,1),(2,1,2,2),(3,2,2,2),(2,2,1,2),(1,2,2,2),(1,2,2,2),(1,1,3,2), (2,2,2,1),(2,2,2,1),(1,3,2,2),(2,2,1,2),(2,2,1,1),(3,2,2,2),(2,1,2,2),(2,1,1,3), (2,2,1,1),(1,1,1,2),(1,1,2,2),(1,1,1,1),(1,1,2,1),(1,2,2,1)
12	(7,5,6,7),(6,5,5,6),(5,5,6,6),(5,7,6,6),(5,4,4,3),(4,4,7,6),(5,7,5,5),(5,5,6,6), (6,5,5,7),(5,4,5,3),(4,3,4,5),(7,6,5,7),(5,5,5,5),(6,6,6,5),(5,7,5,4),(5,3,4,3), (4,5,5,4),(3,4,3,4),(4,4,4,4),(5,3,3,4),(3,2,3,2),(3,2,3,4）
13	(12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12)(12,11,12,12), (8,10,12,12),(12,12,12,12),(12,12,12,12),(11,12,12,12),(12,12,12,10), (12,11, 8, 8), (12,12,12,12),(12,12,12,12),(12,12,11,12),(12,12,12,12), (12,10,12,11),(8, 8,11,10),(11,12, 8,11),(12,11,10, 9),(7,11,10,12), (12,7,10,7),(8, 6, 5, 5)
14	(12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12), (12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12), (12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12), (12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12),(12,12,12,12), (12,12,12,12),(12,12,12,11)

Table 4. The upper bound algebraic degree of the bits of the 9th to the 14th rounds

(1) The initial state. We select S_1,S_8,S_{19} as the active S boxes, which including 12 active bits and the other bits in those three S boxes stay constant. The S box is a substitution, permutation only changes the relative position of the bit and the Add Round Constant can't influence the property of the active bits. So those active bits still stay active and independent after translating from the S box.

(2) The ninth round. The 12 active bits' positions have been changed by the permutation layer. Those 12 active bits enter into 12 different S-boxes of the next round, thus there are still 10 S-boxes that have no active bit.

(3) The tenth round. The 4 variables- x_1, x_2, x_3, x_4 all appear at the right of the equations, so all the bits that translating from the S-box will be effected by those variables. As a result, even though one S box has only one active bit, the other 3 bits will be effected by it and their algebraic degree may be changed from zero to one. So the outcome of all the active S boxes will be active and their algebraic degree's upper bound might become (1,1,1,1). And the other bits' upper bound degree is still (0,0,0,0). After the permutation, the result of this round is showed in the 3rd row of table 4.

(4) The eleventh round. The output of the tenth round enter into the S boxes again and we calculate the algebraic degree of those bits by the assistance of the equations. We chose the first S box as an example, the bits whose algebraic degree is (1,1,0,0) enter the first S box of the eleventh round. According to the equation $y_1=x_1x_2x_4+x_1x_3x_4+x_1x_3+x_1x_4+x_3x_4+x_1+x_2+1$ the highest power is $x_1x_2x_4$ and $x_1x_3x_4$ and the algebraic degree of two variables x_3 and x_4 are 0.We can calculate that the upper bound algebraic degrees of $x_1x_2x_4$, $x_1x_3x_4$ are separately 2 and 1. In conclusion, the upper bound algebraic degree of y_1 is 2. We can use the same method calculate the upper bound algebraic degrees of the other three output bits which are 2,2,1. So after the influence of the S box the upper bound algebraic degrees of the 4 bits have been changed from (1,1,0,0) to(2,2,2,1). Then we use the permutation change the position of the output bits and we can find the final results in the fourth row of the table 4.

(5) The twelfth and thirteenth rounds. We can calculate the change of upper bound algebraic degree of the bits in twelfth and thirteenth rounds. Because there are only twelve variables involved in the whole process, the highest algebraic degree of those bits can't be more than 12. And we can find that the upper bound algebraic degrees of the last four bits are (8,6,5,5).

(6) The fourteenth round. According to the function $y_4=x_2x_3+x_1+x_3+x_4$ and the value of the last four bits , we can calculate that the value of the upper bound of x_2x_3, x_1, x_3, x_4 are 11,8,5,5. So the upper bound algebraic degree of y_4 is 11 which is less than the maximum--12.

So we select 12 bits in the ninth bound and try all the possible 2^{12} values and at the same time all the other bits are constant and we can get a balanced state at the end of the 14th round.

Backward Rounds. The higher-order differential path of the first to the fourth rounds of the distinguisher is the backward rounds. We set the fourth round of the forward direction as the first round of the reverse direction and calculate the upper bound algebraic degree by using the functions of the reverse of S-box.

round	the upper bound algebraic degree of each bit
4	(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1), (0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0), (0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0)
3	(1,1,2,2),(1,1,2,2),(1,1,2,2),(1,1,2,2),(1,1,2,2),(1,1,2,2),(1,1,2,2),(1,1,2,2), (1,1,2,2),(1,1,2,2),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1), (1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1),(1,1,1,1)
2	(3,4,4,4),(3,4,4,4),(2,3,3,4),(2,3,3,4),(3,4,4,4),(3,4,4,4),(2,3,3,4),(2,3,3,4), (3,4,4,4),(3,4,4,4),(2,3,3,4),(2,3,3,4),(3,4,4,4),(3,4,4,4),(2,3,3,4),(2,3,3,4), (3,4,4,4),(3,4,4,4),(2,3,3,4),(2,3,3,4),(2,3,3,3),(2,3,3,3)
1	(8,8,8,8),(8,8,8,8),(6,8,8,8),(8,8,8,8),(7,8,8,8),(8,8,8,8),(6,8,8,8),(7,8,8,8), (6,8,8,8),(8,8,8,8),(7,8,8,8),(8,8,8,8),(7,8,8,8),(8,8,8,8),(6,8,8,8),(8,8,8,8), (7,8,8,8),(7,8,8,8),(5,7,7,8),(7,8,8,8),(6,8,8,8),(8,8,8,8)

Table 5. The upper bound algebraic degree of the 4th to the 1st rounds

(1) The initial state. We chose the first eight bits of the input bits of the fourth round of the forward direction as the active bits which can be denoted by $x_1,x_2,\ldots x_8$. The 8 active bits enter into 8 different S-boxes and the other bits are all constant. We can see from the right of those functions that all the four variables appear. So even though there is only one active bit enter into the S box, all the 4 bits will be influenced by it and their upper bound algebraic degree might become 1, which can be seen

from the second row of the table 5. So the first 8 bits' algebraic degree will become 1 and the other bits that are not influenced by the active bits still are constant and their algebraic degree is 0.

(2) The third round. Firstly, all the bits enter into the permutation layer and their position have been changed. The upper bound power of bits that enter into the first ten S boxes are (1,1,0,0) and that enter into the last twelve S boxes are (1,0,0,0). Then, we can calculate the change of those bits' algebraic degree according to the functions that maintained early. The result is showed in the third row of table 5.

(3)The second and the first round. Using the same method we can get the algebraic degree of the second and the first round and we can get all the 88 bits' algebraic degree, showing in the fifth row of table 4. We can conclude from the table 5 that there are still fifteen bits whose algebraic degree haven't achieve the maximum eight. So it is a balanced state, too.

So if we select the first 8 bits of the fourth round as the active bits, set the fourth round as the initial state and calculate all the 2^8 values while the other bits stay constant, we will have a balanced state at the first round.

Combination of the Two Differential Paths. In this chapter we will combine the higher-order differential attack and the integral attack to build a fourteen rounds high order differential path and get a fourteen round zero-sum distinguisher.

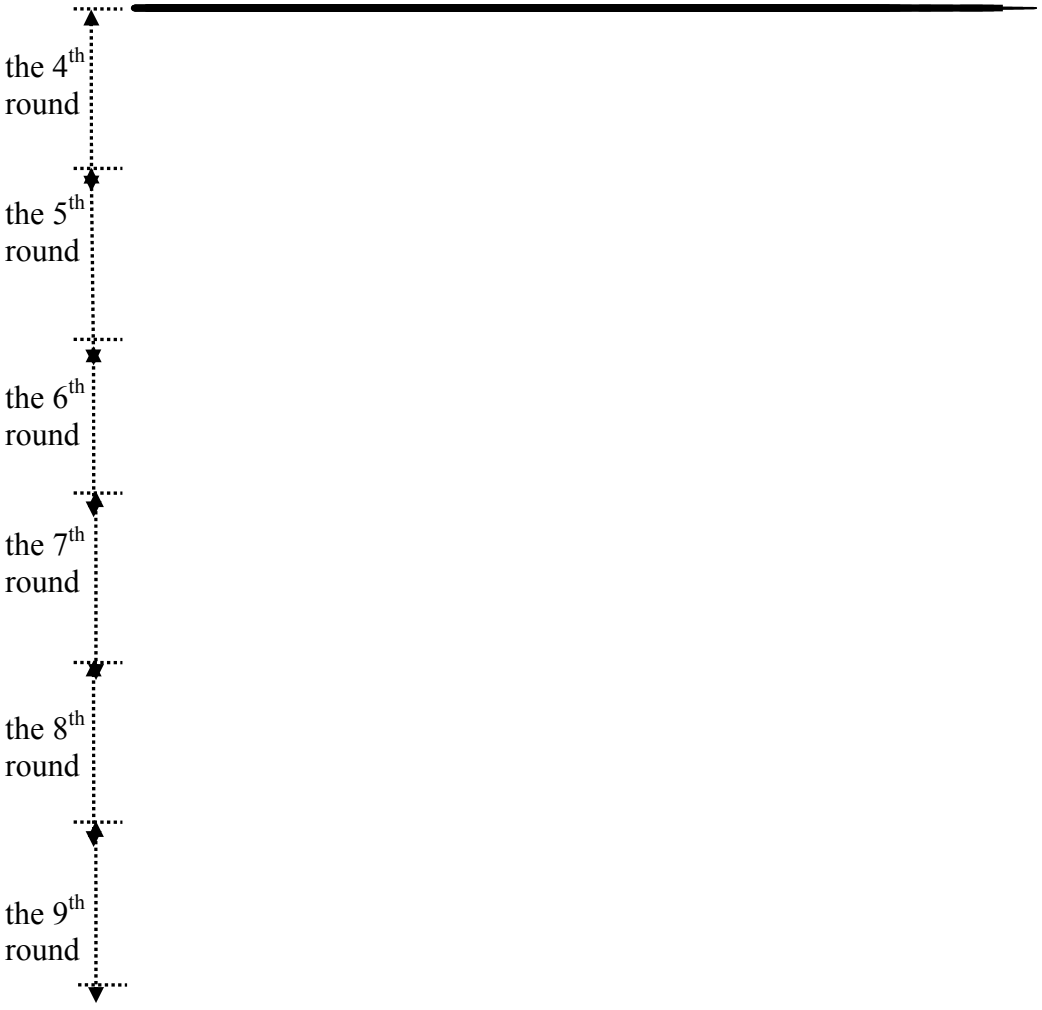

Figure 2. The integral of 4^{th} to 9^{th} rounds

In the integral attack, for a integral attack if we can find a state that all the active and independent bits contain the original active and independent bits of the initial state after the changing of a round, then we can attach this round to head of the primary integral attack path. So in this way we can get a new integral attack path which is one round more than the original one. If the original integral path is

balanced, the new path is also balanced. So we can use this method to combine the forward and backward high order differential paths that we get in the former two chapters.

As shown in figure 2., in order to guarantee the 12output bits of S_1,S_8,S_1 to be independent and active, we need S_1,S_2,S_3,S_4,S_7,S_8,S_9,S_{10} output bits be active and independent. So we need the S_1-S_{18} and S_{20} output bits of the 7^{th} round be active and independent. While in the backward direction, if we want the input bits of S_1,$S_2$to be active and independent, we need the output bits of S_1,S_{12},S_{17},S_{18} of the 6^{th} round be active and independent. And in order to keep those S boxes be active and independent we need keep the outputs of S_1,S_2,S_3,S_5,S_6,S_7,S_8,S_9,S_{10},S_{12},S_{13},S_{14},S_{16},S_{17},S_{18},S_{20},S_{21} of the 7^{th} round active and independent. So we chose union set of the S boxes in the 7^{th} round that the two directions both need as the initial state. So we select S_1-S_{18},S_{20},S_{21}as active S-boxes, we will construct a fourteen rounds zero-sum distinguisher which can obtain the balanced state at the both ends, whose computational complexity is 2^{80} .

Conclusion

In this paper, we introduce an improvement on the 14 rounds zero-sum distinguisher of the SPONGENT-88 and improve the computational complexity from 2^{84} to 2^{80}. And in the future, we will focus on the combining of the integral attack, high-order differential attack and the zero-sum properties to find more connections of those methods. What's more, we can apply the thoughts of this paper to other PRESETN-type algorithms and other variations of the SPONGENT.

References

[1] Aumasson, J.-P. and Meier, W. Zero-sum distinguishers for reduced KECCAK-f and for the core functions of Luffa and Hamsi. Presented at the rump session of Cryptographic Hardware and Embedded System-CHES 2009, 2009.

[2] Bogdanov, A.,Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES. Lecture Notes in Computer Science, vol.4727, pp.450-466. Springer (2007).

[3] Bogdanov A, Knežević M, Leander G, Toz D, Varici K, Verbauwhede I. SPONGENT: A Lightweight Hash Function. Proceedings of the International Workshop of Cryptographic Hardware and Embedded System (CHES 2011). Nara, Japan, 2011: 312-325.

[4] Daemen, J., Knudsen, L.R., Rijmen, V. The block cipher Square. In Fast Software encryption-FSE' 97, vol 1267 of Lecture Notes in Computer Science, pages 149-165. Springer-Verlag, 1997.

[5] De Cannière, C. Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In: Katsikas, S.K., Lopez, J., Backes, M.,Gritzalis, S., Preneel, B. (eds.) ISC. Lecture Notes in Computer Science, vol. 4176, pp.171-186. Springer (2006)

[6] Dong Le, Wu Wen-Ling, Wu Shuang, Zou Jian. Another Look at the Integral Attack by the Higher-Order Differential Attack. CHINESE JOURNAL OF COMPUTERS, 2012, vol. 35 No. 9: 1906-1917.(in Chinese)

[7] Hell, M., Johansson, T., Maximov, A., Meier, W. The Grain Family of Stream Ciphers. In: Robshaw and Billet [25], pp. 179-190.

[8] Hong, D., Sung, J., Hong S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (ed.) CHES. Lecture Notes in Computer Science, vol.4249, pp. 46-59. Springer (2006)

[9] Knudsen L. Truncated and higher order differentials. Proceedings of the International Workshop on Fast Software Encryption (FSE 1994). Leuven, Belgium, 1994: 196-211.

[10] Knudsen, L.R., Wagner, D. Integral cryptanalysis. In Fast Software Encryption- FSE 2002, vol 2365 of Lecture Notes in Computer Science, pages 112-127. Springer-Verlag, 2002.

[11] Lai Xue-Jia. Higher order derivatives and differential cryptanalysis. Proceedings of the Symposium on Communication, Coding and Cryptography. Ascona, Switzerland, 1994: 227-233.