Automatic Differential Analysis of ARX Block Ciphers: with Application to SPECK and LEA

Ling Song[1,2], Zhangjie Huang[1,2], Qianqian Yang[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing 100093, China
{songling,hulei,sunsiwei,yangqianqian}@iie.ac.cn

Abstract. In this paper, we focus on the automatic differential cryptanalysis of ARX block ciphers with respect to XOR-difference, and develop Mouha et al.'s framework of finding differential characteristics by adding a new method for constructing long characteristics from short ones. The new method reduces the searching time a lot and makes it possible to search differential characteristics for ARX block ciphers with large word sizes. Also, we take the differential effect into consideration and find that the differential probability increases by a factor of $4 \sim 2^{10}$ when multiple characteristics are counted in. The efficiency of our method is demonstrated by improved attacks of SPECK and LEA, which cover 1, 1, 4 and 5 more rounds of SPECK48, SPECK64, SPECK96 and SPECK128, respectively, and 2 more rounds of LEA-128.

Key words: Differential cryptanalysis, automatic search, ARX, SPECK, LEA

1 Introduction

ARX ciphers are a broad class of symmetric-key cryptographic algorithms that only consists of three operations: additions modulo 2^n, bit rotations and X-ORs. Some examples of ARX ciphers are: the block cipher FEAL [15], TEA [18], SPECK [4], LEA [11], the stream cipher Salsa20 [5], the SHA-3 finalists Skein [9] and Blake [3]. To evaluate the security of an ARX cipher, differential cryptanalysis [6] is one of the most important attacks that should be considered.

Even though ARX ciphers have a long history for use, their security analysis are lagging behind. For S-box based symmetric-key ciphers, their security against differential cryptanalysis is measured by the number of active S-boxes. On the contrary, there is no rigorous security proof of ARX ciphers against differential cryptanalysis in existing literature, so searching optimal differentials becomes the only way for evaluation. In 2013, Mouha et al. introduce a framework [14] for searching optimal differential characteristics of ARX ciphers, assuming all the operations in the cipher are independent. From the application to Salsa20, the assumption is shown to be invalid sometimes. In [2, 1], Biryukov et al. extend

Matsui's algorithm and based on partial difference distribution tables, a tool is proposed for automatically searching differential characteristics in ARX ciphers. This tool is applicable to differential search with respect to both XOR- and ADD- differences. However, the searching results are not applicable for large word sizes such as $n = 48, 96$.

In this paper, we focus on the automatic differential cryptanalysis of ARX block ciphers with respect to XOR-difference. apply Mouha et al.'s framework of finding differential characteristics to ARX block ciphers where the assumption of independent additions holds, and develop the framework by adding a new method for constructing long characteristics from short ones. This new method reduces the searching time, especially for large word size. Also, we take the differential effect into consideration and find that the differential probability increases by a factor of $4 \sim 2^{10}$ when multiple characteristics are counted in. The efficiency of our new method can be demonstrated by the application to two block ciphers: SPECK and LEA, in which better differentials are found and differential attacks against them are improved. The results are summarized in Table 1 and compared with the best ones of previous works. As can be seen, for SPECK we reduce the complexities of differential attack on SPECK32/64, and attack 1, 1, 4 and 5 more rounds against SPECK48, SPECK64, SPECK96 and SPECK128, respectively; for LEA, except the attacks in the specification we provide the first differential analysis for it and attack 13, 13 and 15 rounds of LEA-128, LEA-192 and LEA-256, respectively.

The rest of this paper is organized as follows. Section 2 provides a background of differential cryptanalysis; Section 3 elaborates on searching method developed in this paper; Section 4 briefly describes the two block ciphers SPECK and LEA; in Section 5 we provide the searching results for differentials of SPECK and LEA, on which attacks are launched; Section 6 is a short discussion; and finally, the last section is the conclusion.

A few words on notations: differences here are expressed using XOR; values for differences are represented in hexadecimal.

2 Background

This section briefly reviews the differential cryptanalysis and differential properties of addition. In the end, the assumption we take in this paper is clarified.

2.1 Differential cryptanalysis

Differential cryptanalysis was introduced by Biham and Shamir in [6]. For block ciphers, it is used to analyze how input differences lead to output differences. If certain input/output difference happens in a non-random way, it can be used to build a distinguisher or even to recover keys.

To consider the security of iterated block ciphers against differential cryptanalysis, Lai et al. first introduced the theory of Markov ciphers and made a

Table 1. Previous attacks and our new attacks on SPECK.

Variant	Rounds attacked/ Total rounds	Time	Data (CP)	Memory	Reference
SPECK32/64	14/22	2^{63}	2^{31}	2^{22}	[7]
	14/22	$\mathbf{2^{61.41}}$	$\mathbf{2^{29.41}}$	$\mathbf{2^{22}}$	**This paper**
SPECK48/72	14/22	2^{65}	2^{41}	2^{22}	[7]
	15/22	$\mathbf{2^{68.31}}$	$\mathbf{2^{44.31}}$	$\mathbf{2^{22}}$	This paper
SPECK48/96	15/23	2^{89}	2^{41}	2^{22}	[7]
	16/23	$\mathbf{2^{92.31}}$	$\mathbf{2^{44.31}}$	$\mathbf{2^{22}}$	**This paper**
SPECK64/96	18/26	2^{93}	2^{61}	2^{22}	[7]
	19/26	$\mathbf{2^{92.56}}$	$\mathbf{2^{60.56}}$	$\mathbf{2^{22}}$	**This paper**
SPECK64/128	19/27	2^{125}	2^{61}	2^{22}	[7]
	20/27	$\mathbf{2^{124.56}}$	$\mathbf{2^{60.56}}$	$\mathbf{2^{22}}$	**This paper**
SPECK96/96	16/28	2^{85}	2^{85}	2^{22}	[7]
	18/28	$\mathbf{2^{85}}$	$\mathbf{2^{85}}$	$\mathbf{2^{22}}$	**This paper**
	20/28	$\mathbf{2^{94.94}}$	$\mathbf{2^{94.94}}$	$\mathbf{2^{22}}$	**This paper**
SPECK96/144	17/29	2^{133}	2^{85}	2^{22}	[7]
	19/29	$\mathbf{2^{133}}$	$\mathbf{2^{133}}$	$\mathbf{2^{22}}$	**This paper**
	21/29	$\mathbf{2^{142.94}}$	$\mathbf{2^{94.94}}$	$\mathbf{2^{22}}$	**This paper**
SPECK128/128	17/32	2^{113}	2^{113}	2^{22}	[7]
	22/32	$\mathbf{2^{124.70}}$	$\mathbf{2^{124.70}}$	$\mathbf{2^{22}}$	**This paper**
SPECK128/192	18/33	2^{177}	2^{113}	2^{22}	[7]
	23/33	$\mathbf{2^{188.70}}$	$\mathbf{2^{124.70}}$	$\mathbf{2^{22}}$	**This paper**
SPECK128/256	19/34	2^{241}	2^{113}	2^{22}	[7]
	24/34	$\mathbf{2^{252.70}}$	$\mathbf{2^{124.70}}$	$\mathbf{2^{22}}$	**This paper**
LEA-128	12/24	2^{84}	2^{100}	2^{76}	[11]
LEA-128	**14/24**	$\mathbf{2^{124.02}}$	$\mathbf{2^{124.02}}$	$\mathbf{2^{22}}$	**This paper**
LEA-192	**14/28**	$\mathbf{2^{124.02}}$	$\mathbf{2^{124.02}}$	$\mathbf{2^{22}}$	**This paper**
LEA-256	**15/32**	$\mathbf{2^{252.02}}$	$\mathbf{2^{124.02}}$	$\mathbf{2^{22}}$	**This paper**

distinction between a differential and a differential characteristic [12]. A differential is a difference propagation from an input difference to an output difference, while a differential characteristic specifies not only the input/output difference, but also all the internal differences after each round. For a Markov cipher, the probability of a differential characteristic is the multiplication of difference transition probabilities of each round, and the probability of a differential is equal to the sum of the probabilities of all differential characteristic which correspond to the differential. Practically, an iterated block cipher is taken as a Markov cipher when the key schedule generates (almost) random round keys.

In this paper, we consider XOR-differences, and assume the round keys are random. If the round keys are xored in the encryption, the key part can be neglected in differential searching.

2.2 Estimating differential probabilities for ARX ciphers

For ARX block ciphers, only additions modulo 2^n are non-linear operations and propagate differences indefinitely. So we focus on calculating differential probability of addition. In [13], Lipmaa and Moriai study the differential properties of addition. Let $\mathrm{xdp}^+(\alpha, \beta \to \gamma)$ be the XOR-differential probability of addition modulo 2^n, with input difference α, β and output difference γ. The authors prove that the differential $(\alpha, \beta \to \gamma)$ is valid if and only if

$$\mathrm{eq}(\alpha \ll 1, \beta \ll 1, \gamma \ll 1) \wedge (\alpha \oplus \beta \oplus \gamma \oplus (\beta \ll 1)) = 0, \tag{1}$$

where

$$\mathrm{eq}(x, y, z) := (\neg x \oplus y) \wedge (\neg x \oplus z). \tag{2}$$

For every valid differential $(\alpha, \beta \to \gamma)$, the weight $w(\alpha, \beta \to \gamma)$ is defined as follows:

$$w(\alpha, \beta \to \gamma) := -\log_2(\mathrm{xdp}^+(\alpha, \beta \to \gamma)).$$

The weight of a valid differential can be calculated as:

$$w(\alpha, \beta \to \gamma) = h(\neg \mathrm{eq}(\alpha, \beta, \gamma)), \tag{3}$$

where $h(x)$ denotes the number of non-zero bits in x except the most significant bit.

Assumption of independent additions In this paper, we assume that additions in the block cipher are independent of each other with regard to XOR-difference due to the use of round keys. Under this assumption, the probability of a differential characteristic is equal to the multiplication of the probabilities of all addition operations. Specifically, we calculate the weight of a differential characteristic as the sum of the weights of all addition operations.

3 Automatic Search for Characteristics and Differentials in ARX Block ciphers

In this section we elaborates on the searching method used in this paper.

3.1 Mouha's framework to search differential characteristics of ARX ciphers

In [14], Mouha and Preneel construct a framework to search for optimal differential characteristics of ARX ciphers and apply it to Salsa20. In their framework, a typical Satisfiability Modulo Theory (SMT) solver STP [10] is used. STP is built upon a SAT solver. Since many word-wise operations are included in its input language, it is suitable for searching problems of ARX ciphers.

In the framework, they find differential characteristics up to a certain weight W with STP as follows. First, they write simple equations in respect of XOR-difference for every addition, rotation and XOR of the ARX cipher as follows.

- Use n-bit variables to represent input difference words.
- Introduce additional n-bit variables to represent the differences after the addition, XOR, and rotation operations when required.
- Use Equation (1(and (2) for every addition modulo 2^n of the ARX cipher to ensure that the input and output differences correspond to a valid differential of the addition operation.
- Include Equation (3) to calculate the weight of each addition operation, and represent the sum of weights of all additions with W, which corresponds to the weight of the differential characteristics under consideration.
- Specify that input difference is non-zero and restrict W to a maximum of a certain number.

Second, they feed the equations generated into STP. STP converts these equations into formulae of conjunctive normal form (CNF), and then invokes an underlying SAT solver to find solutions.

Although Mouha's framework that multiplies the differential probabilities of all additions was originally applied to a stream cipher, it is more suitable for ARX block ciphers where a round key is XORed each round. The reason is that additions in an ARX stream ciphe are usually not independent, while additions in an ARX block cipher may be independent due to the use of round keys.

3.2 Obtaining a long characteristic from two short ones

Mouha's framework can be applied directly to ARX block ciphers where additions are independent with regard to XOR-difference. However, due to the limitation of computation power, it takes too much time to find a long characteristic. In this paper, we introduce a method to obtain a long characteristic from two short ones. The method lies in searching differential characteristics from an internal difference which has only one active bit. This idea for searching long characteristics was inspired by the phenomenon that many searched characteristics with best probabilities have a special internal difference with only one active bit which usually leads to a differential transition of the nearest round with probability 1.

The method for obtaining long characteristics is illustrated in Fig. 1. First, we set an internal difference after some rounds D to be a value where only

one bit is nonzero, and then search forward and backward independently to get two short characteristics. After that we combine these two short characteristics together to get a long one. Since either the input or output difference is fixed, two short characteristics with best probability can be easily searched. Note that this method saves much time for searching long characteristics, but doesn't always guarantee best characteristics.

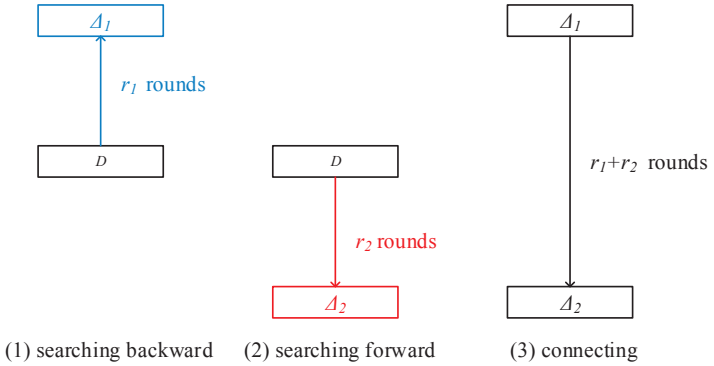

Fig. 1. Obtaining a longer characteristic from two shorter ones.

This method in differential attack resembles the one used in boomerang attack [17]. However, conditions for the two short characteristics are different. Suppose the probabilities of the two short characteristics are p and q respectively, and the block size is N. For differential attack, $pq > 2^{-N}$ and the two short characteristics must be connected, while for boomerang attack $pq > 2^{-N/2+1}$ but the two short characteristics are independent.

3.3 Characteristics to differentials

For ARX ciphers, the probability of one characteristic cannot well approximate the probability of the corresponding differential because of a strong differential effect, that is, between the input difference and the output difference there are many characteristics.

To calculate the differential probability as accurately as possible, more characteristics sharing the same input and output difference should be counted in. After a good characteristic is obtained, we fix the input and output difference, and search all characteristics with probability less or equal than that of the one obtained. Specifically, if the characteristic obtained has a weight W, we search all characteristics with the same input and output difference where the weight is $W, W+1, W+2, \cdots$, and add the probabilities of all these characteristics together. Note that STP just outputs one solution. To find all solutions, the user can tell STP to generate the CNF formulae and exit. A special SAT solver, such as CryptoMiniSat [16], can then be used to get all solutions.

4 Description of SPECK and LEA

4.1 SPECK

SPECK is family of lightweight block cipher designed by researchers from the U.S. National Security Agency (NSA)[4]. It contains 10 variants, each of which is characterized by its block size $2n$ and key size mn. For example, SPECK32/64 refers to the SPECK block cipher with block size 32 bits and key size 64 bits. The parameters of SPECK are listed in Table 2.

Table 2. The SPECK parameters.

Block Size $2n$	Key Size mn	Word Size n	Key Words m	Rounds T	α	β
32	64	16	4	22	7	2
48	72	24	3	22	8	3
	96		4	23	8	3
64	96	32	3	26	8	3
	128		4	27	8	3
96	96	48	2	28	8	3
	144		3	29	8	3
128	128	64	2	32	8	3
	192		3	33	8	3
	256		4	34	8	3

The SPECK2n encryption maps a plaintext of two n-bit words (x_0, y_0) into a ciphertext (x_T, y_T), using a sequence of T rounds. The key-dependent round function is defined as

$$R^k(x, y) = (((x \ggg \alpha) \boxplus y) \oplus k, (y \lll \beta) \oplus ((x \ggg \alpha) \boxplus y) \oplus k),$$

where k is the round key, and rotation constants α and β are given in Table 2.

The SPECK key schedule reuses the round function to generate the round keys $k_0, \cdots, k_T$. The m-word master key $K = (l_{m-2}, \cdots, l_0, k_0)$ are used as follow:

$$l_{i+m-1} = (k_i \boxplus (l_i \ggg \alpha)) \oplus i$$
$$k_{i+1} = (k_i \lll \beta) \oplus l_{i+m-1}.$$

Figure 2 provides a schematic view on the round function and the key schedule of SPECK.

4.2 LEA

LEA is an ARX block cipher designed by Hong et al.[11] and provides a high-speed software encryption on general-purpose processors. It has the block size of 128 bits and the key size of 128, 192, or 256 bits. We denote the algorithms with 128-bit, 192-bit, and 256-bit keys by LEA-128, LEA-192, and LEA-256, respectively.

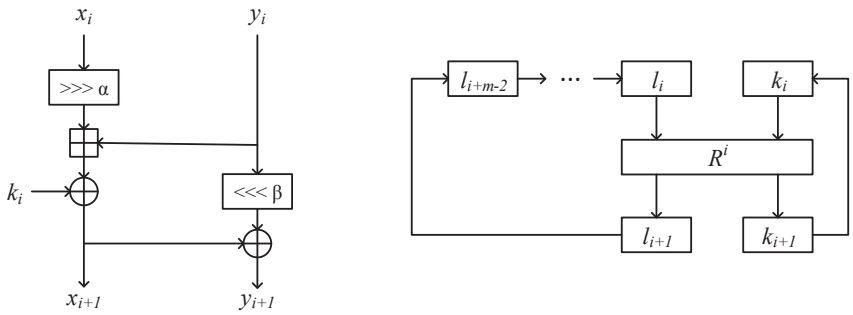

Fig. 2. The round function and the key schedule of SPECK. R^i is the SPECK round function with i acting as the round key.

The encryption of LEA maps a plaintext of four 32-bit words $(x_0^0, x_1^0, x_2^0, x_3^0)$ into a ciphertext $(x_0^r, x_1^r, x_2^r, x_3^r)$ using a sequence of r rounds, where $r = 24$ for LEA-128, $r = 28$ for LEA-192 and $r = 32$ for LEA-256. The round function for round $i, 0 \leq i < r$ is defined as follows:

$$\begin{aligned}
x_0^{i+1} &\leftarrow ((x_0^i \oplus rk_0^i) \boxplus (x_1^i \oplus rk_1^i)) \lll 9,\\
x_1^{i+1} &\leftarrow ((x_1^i \oplus rk_2^i) \boxplus (x_2^i \oplus rk_3^i)) \ggg 5,\\
x_2^{i+1} &\leftarrow ((x_2^i \oplus rk_4^i) \boxplus (x_3^i \oplus rk_5^i)) \ggg 3,\\
x_3^{i+1} &\leftarrow x_0^i.
\end{aligned}$$

where $rk^i = (rk_0^i, rk_1^i, rk_2^i, rk_3^i, rk_4^i, rk_5^i)$ is the round key, which is generated by a key schedule. We take LEA-128 as an example. Let $K = (k_0, k_1, k_2, k_3)$ be a 128-bit key. We set $t_i^0 = k_i$ for $0 \leq i < 4$. For round $i, 0 \leq i < r$, rk^i is produced through following relations:

$$\begin{aligned}
t_0^{i+1} &\leftarrow (t_0^i \boxplus (\delta^i \lll i)) \lll 1,\\
t_1^{i+1} &\leftarrow (t_1^i \boxplus (\delta^i \lll i+1)) \lll 3,\\
t_2^{i+1} &\leftarrow (t_2^i \boxplus (\delta^i \lll i+2)) \lll 6,\\
t_3^{i+1} &\leftarrow (t_3^i \boxplus (\delta^i \lll i+3)) \lll 11,\\
rk^i &\leftarrow (t_0^{i+1}, t_1^{i+1}, t_2^{i+1}, t_1^{i+1}, t_3^{i+1}, t_1^{i+1}).
\end{aligned}$$

where δ^i is the constant for round i. Figure 3 provides a schematic view on the round function of LEA and the key schedule of LEA-128. For key schedules of LEA-192 and LEA-256, please refer to appendix.

5 Searching Results and Attacks of SPECK and LEA

In this section we apply the searching method explained in Section 3 to SPECK and LEA. For ten version of SPECK and LEA, we would like to find the longest characteristics. To this goal, we need to find the minimal weight of differential

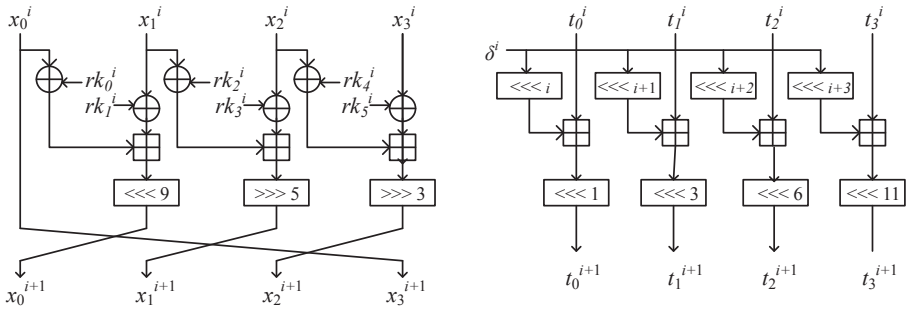

Fig. 3. The round function of LEA and the key schedule of LEA-128.

characteristics with given number of rounds. Suppose the block size is N. If the weight of an r-round differential characteristic is less than N, then the corresponding differential characteristic can be used to build a distinguisher or to recover the key. On the contrary, if the minimal weight of all r-round differential characteristics is no less than N, then no useful differential characteristic exists for that r-round cipher. Note that even though the weight of some characteristics is larger than N, the weight of the corresponding differential may be less than N. Therefore, we also evaluate the probability of the corresponding differential by counting in more characteristics which share the same input and output difference. For a differential, as long as its weight is less than N, it is useful, and our attacks in this paper are mounted based on differentials. Note that all of the characteristics are searched with STP2.0 on a 3.4GHz Itel Core i7-2600 processor, and CryptoMiniSat4 is used as the underlying solver of STP.

5.1 Characteristics and Differentials of SPECK

Characteristics of SPECK32 and SPECK48. We directly apply Mouha et al.'s framework to two smallest versions of SPECK. For SPECK32, the best 9-round characteristic obtained has a weight of 30, which coincides with that of [2]. We provides the source code for searching 9-round characteristics of SPECK32 in Appendix D for verification. In addition, from a 10-round characteristic with weight 35 as shown in Table 6, we get the corresponding differential $(0040, 2040) \rightarrow (A840, 0800)$ with weight 31. The probability calculation of the differential is displayed in Table 3. This 10-round characteristic is the longest distinguisher for SPECK32 in the literature. For SPECK48, our computer takes 12.5 days to find a 11-round characteristic with weight 46, and the corresponding differential has a weight of 43.31.

Characteristics of SPECK64, SPECK96 and SPECK128. We construct long characteristics for these three lager versions of SPECK from two short ones. Take SPECK64 as an example. We set an internal difference to be $(00000080, 00000000)$ and search forward and backward independently. According to the experiments, a 4-round forward characteristic with weight 9 and an

Table 3. A 10-round differential of SPECK32 with $(\Delta x_0, \Delta y_0) = (2040, 0040)$, $(\Delta x_{10}, \Delta y_{10}) = (0800, A840)$.

weight	#sol.	$\log_2$Pr.	$\log_2 \Sigma_{acc}$
35	2	-34	-34
36	4	-34	-33
37	6	-34.42	-32.54
38	16	-34	-32.09
39	38	-33.75	-31.70
40	50	-34.36	-31.48
41	88	-34.54	-31.32
42	92	-35.48	-31.24
43	180	-35.51	-31.17
44	226	-36.18	-31.12
45	284	-36.85	-31.10
46	502	-37.03	-31.07
47	802	-37.35	-31.06
48	1296	-37.66	-31.04
49	2044	-38.00	-31.03
50	3646	-38.17	-31.02
51	5974	-38.46	-31.0109
52	10272	-38.67	-31.0038

11-round backward characteristic with weight 53 can be combined to get a 15-round characteristic of weight 62. The corresponding differential has a weight less than 59.56. For SPECK96 and SPECK128, the searching works similarly. However, for both of them, differentials are derived from characteristics with weight larger or equal than the block size. Specifically, from a 17-round characteristic of SPECK96 with weight 96 we get a 17-round differential with weight less than 93.94; from a 19-round characteristic of SPECK128 with weight 129, a 19-round differential with weight less than 123.70. The times for searching long characteristics for SPECK64, SPECK96 and SPECK128 are 0.9 hour, 11.3 hours and 5.2 hours respectively, which are much less compared with the time for directly searching characteristics of SPECK48.

Comparison. Table 5 compares the differentials we find with the differentials of SPECK in the literature. For SPECK32, we find a 10-round differential with probability $2^{-31.02}$, the best distinguisher of SPECK32 to date. We also find a 9-round characteristic of SPECK32 that coincides with that of [2] but has a tighter estimation of differential probability. For SPECK48, we obtain an 11-round characteristic with a better weight. For SPECK64, SPECK96 and SPECK128, the characteristics which we find cover 1, 4 and 4 more round(s) than previous works.

5.2 Characteristics and Differentials of LEA

Characteristics of LEA. We construct long characteristics for LEA from two short ones. We set an internal difference to be (00000100, 00000000, 00000000, 000

Table 4. Comparison of our differentials of SPECK with previous ones.

Block Size	Rounds r	Prob.	Input Difference	Output Difference	Reference
32	9	2^{-30}	(8054, A900)	(0040, 0542)	[2]
	9	$\mathbf{2^{-28.41}}$	(8054, A900)	(0040, 0542)	**This paper**
	10	$\mathbf{2^{-31.01}}$	(0040, 2040)	(A840, 0800)	**This paper**
48	11	$2^{-46.48}$	(202040, 082921)	(808424, 84A905)	[2]
	11	$\mathbf{2^{-43.31}}$	(504200, 004240)	(202001, 202000)	**This paper**
64	14	$2^{-59.02}$	(00000009, 01000000)	(00040024, 04200D01)	[2]
	15	$\mathbf{2^{-59.56}}$	(04092400, 20040104)	(808080A0, A08481A4)	**This paper**
96	13	2^{-84}	(2A20200800A2, 322320680801)	(1008004C804, C0180228C60)	[8]
	15	$\mathbf{2^{-84}}$	(000900000000, 000001000000)	(A0A000008880, 81A02004C88C)	**This paper**
	17	$\mathbf{2^{-93.94}}$	(240004000009, 010420040000)	(A0A000008880, 81A02004C88C)	**This paper**
128	15	$2^{-117.28}$	(0640240804002440, 6004400C20040004)	(828028080A080888, E88C81A4A0924B2C)	[8]
	18	$\mathbf{2^{-117.75}}$	(0202000000000080, 8012020000000480)	(0800002080820808, 48080124A0924A08)	**This paper**
	19	$\mathbf{2^{-123.70}}$	(4000400000000012, 1042004000000080)	(0800002080820808, 48080124A0924A08)	**This paper**

00000) and search forward and backward independently. A 12-round characteristic of weight 112 can be constructed by combining two short ones of 6 forward rounds and 6 backward rounds respectively. From this characteristic we derive a 12-round differential

$$(10401080, 0A001080, 02041208, 00049228)$$
$$\rightarrow(88008008, 88A2A00A, 22020060, 00000010)$$

with weight less than 101.71. Also, a 13-round characteristic can be constructed by connecting two short ones of 6 forward rounds and 7 backward rounds and its weight is 134. From this characteristic a 13-round differential

$$(00049018, 40049000, 10220041, 00028001)$$
$$\rightarrow(88008008, 88A2A00A, 22020060, 00000010)$$

of weight less than 123.02 is derived.

The details of these two characteristics are shown in Table 8.

5.3 Differential attacks on SPECK and LEA

Differential attacks on SPECK. In [7] Itai proposed an enumeration technique for key recovery in differential attacks against SPECK. Given a differential characteristic for SPECK$2n/mn$ that covers r rounds of the cipher with probability $p > 2 \cdot 2^{-2n}$, the enumeration technique can be used to recover the key of

Table 5. Comparison of our differentials of LEA with previous ones.

#Rounds	Prob.	Reference
11	2^{-98}	[11]
12	2^{-128}	[11]
12	$\mathbf{2^{-101.71}}$	**This paper**
13	$\mathbf{2^{-123.02}}$	**This paper**

a variant with $r+m$ rounds with $2 \cdot p^{-1}$ chosen plaintexts, in an averaged time complexity of $2 \cdot p^{-1} \cdot 2^{(m-2)n}$ encryptions. The required memory is constant for all versions of SPECK, which is 2^{22} bytes, i.e. only a few megabytes. Appendix C provides more information about Itai's enumeration technique.

Adding one round for free. We use the r-round differential $(\alpha \rightarrow \beta)$ over rounds $2 \sim (r+1)$, and choose pairs of plaintexts such that their difference after the first round is α. In this way, one more round can be extended for free. This idea was also adopted by Abed et al. in [8]. Consequently, given a r-round differential, the attack can cover $(r+m+1)$ rounds.

For SPECK32/64, we use the same 9-round differential as in [8, 2]. According to our experiments, the differential holds with probability at least $2^{-28.41}$, which is much larger than 2^{-30}, the probability of the best characteristic of the differential. This indicates that the complexities of the attack can be reduced with a tighter estimation of the probability of the differential. Combined with Itai's enumeration technique for key recovery, the differential can be used to attack a 14-round SPECK32/64 at a cost of $2 \cdot 2^{28.41} = 2^{29.41}$ plaintexts and $2 \cdot 2^{28.41} \cdot 2^{32} = 2^{61.41}$ encryptions.

Differential attacks for the rest variants are similar to that of SPECK32/64, so we omit the details on calculation of the complexities. The attacks are mounted based on the differentials in Table 5 and the results are summarized in Table 1. Compared with the previous works, the attacks on SPECK48, SPECK64, SPCKE96, SPECK128 extend 1, 1, 4 and 5 more round(s) respectively.

Differential attacks on LEA. Since the differential equations of addition in the key recovery of LEA are similar to that of SPECK, Itai's enumeration technique can be adapted to LEA. Given an r-round differential characteristic of LEA with probability $p > 2 \cdot 2^{-N}$ where N is the block size, for LEA-128 and LEA-192, the attack recovers the key of a variant of $(r+1)$ rounds with $2 \cdot p^{-1}$ plaintexts, in expected time complexity of $2 \cdot p^{-1}$ encryptions, while $(r+2)$ rounds of LEA-256 can be attacked with $2 \cdot p^{-1}$ plaintexts and $2 \cdot p^{-1} \cdot 2^N$ encryptions in average. The attacks are summarized in Table.

6 Discussion

Differential effect. Experiment results demonstrate the strong differential effect of ARX block ciphers. When the characteristics sharing the same input and output difference are counted in, the differential probability increases by a factor of $4 \sim 8$ for SPECK and by a factor more than 2^{10} for LEA. Due to this

differential effect, the probability of a characteristic shouldn't be simply taken as the differential probability for these ARX block ciphers.

Limitation of our searching method. The searching method discussed in this paper takes the assumption of independent additions with respect of XOR-difference. However, additions are dependent in most ARX block ciphers, such as TEA, Chaskey and RAIDEN, to which our searching method can not be applied directly. One of our future work is to deal with the dependency among additions.

7 Conclusion

In this paper, we apply Mouha et al.'s framework of finding differential characteristics to ARX block ciphers where the additions are independent with respect to XOR differences, and develop this framework by adding a new method for constructing long characteristics from short ones. This new method reduces the searching time a lot and makes it possible to search differential characteristics for ARX block ciphers with large word size. Also, we take the differential effect into consideration and the results show the probability of a characteristic shouldn't be simply taken as the differential probability for these ARX block ciphers. The efficiency of our method is demonstrated by improved attacks of SPECK and LEA. One of our future work is to deal with the dependency among additions that are common in most ARX ciphers.

Acknowledgement: The authors would like to thank Jian Guo for his valuable suggestions.

References

1. Vesselin Velichkov Alex Biryukov. Automatic search for differential trails in arx ciphers. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 227–250. Springer International Publishing, 2014.
2. Vesselin Velichkov Alex Biryukov, Arnab Roy. Differential analysis of block ciphers simon and speck. In Cid Carlos and Rechberger Christian, editors, *Fast Software Encryption - FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 546–570. Springer Berlin Heidelberg, 2014.
3. Henzen L. Meier W. Phan R.C.-W. Aumasson, J.-P. Sha-3 proposal blake. Technical report, Submission to the NIST SHA-3 Competition (Round 2), 2008.
4. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck families of lightweight block ciphers. Technical report, Cryptology ePrint Archive, Report 2013/404, 2013.
5. Daniel J. Bernstein. *New Stream Cipher Designs: The eSTREAM Finalists*, chapter The Salsa20 Family of Stream Ciphers, pages 84–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
6. Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

7. Itai Dinur. Improved differential cryptanalysis of round-reduced speck. In Antoine Joux and Amr Youssef, editors, *Selected Areas in Cryptography – SAC 2014*, volume 8781 of *Lecture Notes in Computer Science*, pages 147–164. Springer International Publishing, 2014.
8. Stefan Lucks Jakob Wenzel Farzaneh Abed, Eik List. Differential cryptanalysis of round-reduced simom and speck. In Cid Carlos and Rechberger Christian, editors, *Fast Software Encryption - FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 525–545. Springer Berlin Heidelberg, 2014.
9. Lucks S. Schneier B. Whiting D. Bellare-M. Kohno T. Callas J. Walker J. Ferguson, N. The skein hash function family. Technical report, Submission to the NIST SHA-3 Competition (Round 2), 2009.
10. Vijay Ganesh and David L. Dill. A decision procedure for bit-vectors and arrays. In *Proceedings of the 19th International Conference on Computer Aided Verification*, CAV'07, pages 519–531, Berlin, Heidelberg, 2007. Springer-Verlag.
11. Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Dong-Geon Lee. *Information Security Applications: 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers*, chapter LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors, pages 3–27. Springer International Publishing, Cham, 2014.
12. Xuejia Lai, JamesL. Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer Berlin Heidelberg, 1991.
13. Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In Mitsuru Matsui, editor, *Fast Software Encryption*, volume 2355 of *Lecture Notes in Computer Science*, pages 336–350. Springer Berlin Heidelberg, 2002.
14. Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for arx: Application to salsa20. Technical report, Cryptology ePrint Archive, Report 2013/328, 2013.
15. Akihiro Shimizu and Shoji Miyaguchi. *Advances in Cryptology — EUROCRYPT' 87: Workshop on the Theory and Application of Cryptographic Techniques Amsterdam, The Netherlands, April 13–15, 1987 Proceedings*, chapter Fast Data Encipherment Algorithm FEAL, pages 267–278. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.
16. Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending sat solvers to cryptographic problems. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, SAT '09, pages 244–257, Berlin, Heidelberg, 2009. Springer-Verlag.
17. David Wagner. The boomerang attack. In Lars Knudsen, editor, *Fast Software Encryption*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer Berlin Heidelberg, 1999.
18. David J. Wheeler and Roger M. Needham. *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings*, chapter TEA, a tiny encryption algorithm, pages 363–366. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.

A Key Schedules of LEA-192 and LEA-256

Key schedule of LEA-192 Let $K = (k_0, k_1, k_2, k_3, k_4, k_5)$ be a 192-bit key. We set $t_i^0 = k_i$ for $0 \leq i < 6$. For round $i, 0 \leq i < 28$, rk^i is produced through following relations:

$$\begin{aligned}
t_0^{i+1} &\leftarrow (t_0^i \boxplus (\delta^i \lll i)) \lll 1,\\
t_1^{i+1} &\leftarrow (t_1^i \boxplus (\delta^i \lll i+1)) \lll 3,\\
t_2^{i+1} &\leftarrow (t_2^i \boxplus (\delta^i \lll i+2)) \lll 6,\\
t_3^{i+1} &\leftarrow (t_3^i \boxplus (\delta^i \lll i+3)) \lll 11,\\
t_4^{i+1} &\leftarrow (t_4^i \boxplus (\delta^i \lll i+4)) \lll 13,\\
t_5^{i+1} &\leftarrow (t_5^i \boxplus (\delta^i \lll i+5)) \lll 17,\\
rk^i &\leftarrow (t_0^{i+1}, t_1^{i+1}, t_2^{i+1}, t_3^{i+1}, t_4^{i+1}, t_5^{i+1}).
\end{aligned}$$

where δ^i is the constant for round i.

Key schedule of LEA-256 Let $K = (k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7)$ be a 192-bit key. We set $t_i^0 = k_i$ for $0 \leq i < 8$. For round $i, 0 \leq i < 32$, rk^i is produced through following relations:

$$\begin{aligned}
t_{6i \mod 8} &\leftarrow (t_{6i \mod 8} \boxplus (\delta^i \lll i)) \lll 1,\\
t_{6i+1 \mod 8} &\leftarrow (t_{6i+1 \mod 8} \boxplus (\delta^i \lll i+1)) \lll 3,\\
t_{6i+2 \mod 8} &\leftarrow (t_{6i+2 \mod 8} \boxplus (\delta^i \lll i+2)) \lll 6,\\
t_{6i+3 \mod 8} &\leftarrow (t_{6i+3 \mod 8} \boxplus (\delta^i \lll i+3)) \lll 11,\\
t_{6i+4 \mod 8} &\leftarrow (t_{6i+4 \mod 8} \boxplus (\delta^i \lll i+4)) \lll 13,\\
t_{6i+5 \mod 8} &\leftarrow (t_{6i+5 \mod 8} \boxplus (\delta^i \lll i+5)) \lll 17,\\
rk^i &\leftarrow (t_{6i \mod 8}, t_{6i+11 \mod 8}, t_{6i+2 \mod 8}, t_{6i+3 \mod 8}, t_{6i+4 \mod 8}, t_{6i+5 \mod 8}).
\end{aligned}$$

where δ^i is the constant for round i.

B Differential Characteristics

C Itai's Enumeration Technique for Key Recovery Attack against SPECK

In this section, we review Itai's key recovery attack against SPECK and refer the reader to [7] for a complete description.

C.1 The full differential attack

Counting techniques are common in key recovery of differential cryptanalysis, while in the differential cryptanalysis of SPECK, Itai increased the number of

Table 6. Differential characteristics for SPECK32, SPECK48 and SPECK64.

r	SPECK32			SPECK48			SPECK64		
	Δx	Δy	$\log_2 p$	Δx	Δy	$\log_2 p$	Δx	Δy	$\log_2 p$
0	2040	0040		504200	004240		04092400	20040104	
1	8000	8100	-1	001202	020002	-5	20000820	20200001	-6
2	8000	8402	-1	000010	100000	-3	00000009	01000000	-4
3	8D02	9D08	-4	000000	800000	-1	08000000	00000000	-2
4	6002	1420	-9	800000	800004	0	00080000	00080000	-1
5	1060	40E0	-5	808004	808020	-2	00080800	00480800	-2
6	0380	0001	-6	8400A0	8001A4	-4	00480008	02084008	-4
7	0004	0000	-3	608DA4	608080	-9	06080808	164A0848	-7
8	0800	0800	-1	042003	002400	-11	F2400040	40104200	-13
9	0810	2810	-2	012020	000020	-5	00820200	00001202	-8
10	0800	A840	-3	200100	200000	-3	00009000	00000010	-4
11				202001	202000	-3	00000080	00000000	-2
12							80000000	80000000	0
13							80800000	80800004	-1
14							80008004	84008020	-3
15							808080A0	A08481A4	-5
$\Sigma_r \log_2 p_r$			-35			-46			-62
$\log_2 p_{\text{diff}} >$			-31.01			-43.31			-59.56

Table 7. Differential characteristics for SPECK96 and SPECK128.

r	SPECK96			SPECK128		
	Δx	Δy	$\log_2 p$	Δx	Δy	$\log_2 p$
0	240004000009	010420040000		4000400000000012	1042004000000080	
1	082020000000	000120200000	-6	0202000000000080	8012020000000480	-6
2	000900000000	000001000000	-4	0010000000000480	0080100000002084	-5
3	000008000000	000000000000	-2	8080000000006080	84808000000164A0	-6
4	000000080000	000000080000	-1	0400000000032400	2004000000080104	-11
5	000000080800	000000480800	-2	2000000000080020	2020000000480801	-7
6	000000480008	000002084008	-4	0000000000480001	0100000002084008	-6
7	0800FE080808	0800EE4A0848	-12	000000000E080808	080000001E4A0848	-8
8	000772400040	400000104200	-21	00000000F2400040	4000000000104200	-15
9	000000820200	000000001202	-11	0000000000820200	0000000000001202	-8
10	000000009000	000000000010	-4	0000000000009000	0000000000000010	-4
11	000000000080	000000000000	-2	0000000000000080	0000000000000000	-2
12	800000000000	800000000000	0	8000000000000000	8000000000000000	0
13	808000000000	808000000004	-1	8080000000000000	8080000000000004	-1
14	800080000004	840080000020	-3	8000800000000004	8400800000000020	-3
15	808080800020	A08480800124	-5	8080808000000020	A084808000000124	-5
16	800400008124	842004008801	-9	4004000080000124	4420040080000801	-10
17	A0A000008880	81A02004C88C	-9	2020000080800802	0120200480804808	-11
18				0100000480004800	0801002084020840	-11
19				0800002080820808	48080124A0924A08	-10
$\Sigma_r \log_2 p_r$			-96			-129
$\log_2 p_{\text{diff}} >$			-93.94			-123.70

Table 8. Differential characteristics for LEA.

r	12-round		13-round	
	$\Delta x_0 \parallel \Delta x_1 \parallel \Delta x_2 \parallel \Delta x_3$	$\log_2 p$	$\Delta x_0 \parallel \Delta x_1 \parallel \Delta x_2 \parallel \Delta x_3$	$\log_2 p$
0	104010800A0010800204120800049228		0004901840049000000280011022004l	
1	8000001440402014004010041040l080	-20	104010800A0010800204100800049018	-20
2	8040008086000080820000108000001l	-16	800000144040200C0040100410401080	-20
3	8000000C8040000C8040000480400080	-14	8040008086000080820000108000001l	-18
4	800000008000000080000010800000OC	-10	8000000C8040000C8040000480400080	-14
5	000000008000000080000000800000000	-4	800000008000000080000010800000OC	-10
6	000001000000000000000000000000000	0	000000008000000080000000800000000	-4
7	000200000000000000000000000000100	-1	000001000000000000000000000000000	-0
8	040000000000000000000002000020000	-2	000200000000000000000000000000100	-1
9	000000080000000700004004040000000	-6	040000000000000000000002000020000	-2
10	000002000800020080800800000000008	-11	000000080000000700004004040000000	-6
11	000000100444005010100101000002000	-9	000002000800020080800800000000008	-11
12	8800800888A2A00A2202006000000010	-19	000000100444005010100101000002000	-9
13			8800800888A2A00A2202006000000010	19
$\Sigma_r \log_2 p_r$		-112		-134
$\log_2 p_{\text{diff}} >$		-101.71		-123.02

rounds attacked by the application of enumeration techniques in the key recovery [7]. Instead of extracting partial key material from outer rounds of the cipher using statistical analysis, the enumeration technique tries all suggestions for the full key proposed by a sub-cipher attack.

To describe the details of Itai's attack on SPECK with enumeration technique, we first consider the case that $m = 2$, i.e. the master key contains 2 words. The attack of the case that $m = 2$ can be easily extended to other cases, in which $m = 3$ or $m = 4$.

Given an r-round differential of the cipher $(\Delta x_0, \Delta y_0) \rightarrow (\Delta x_r, \Delta y_r)$ with high probability p, the $(r + 2)$-round attack is proceeded as follow.

1. Request the encryption of p^{-1} plaintext pairs P and $P' = P \oplus (\Delta x_0, \Delta y_0)$ and denote the corresponding ciphertexts by C and C', respectively.
2. For each plaintext pair P and P':
   (a) Execute the 2-round attack (Section 7 of [7]) using $(\Delta x_r, \Delta y_r)$, C and C' and get suggestions for k_{r+1} and k_r.
   (b) For each returned value of k_{r+1} and k_r, reverse the key schedule to obtain the master key. Test the master key using additional encryptions. Return the master key if it passes the test.

The above attack requires $2 \cdot p^{-1}$ chosen plaintexts. Since in the key recovery, the 2-round attack has an average time complexity less than 2 encryptions, the total time complexity of the attack is $2 \cdot p^{-1}$. As Itai pointed in [7], the memory complexity is 2^{22} bytes, i.e. only a few megabytes. In next subsection, we describe the 2-round attack.

For $m > 2$ ($m = 3$ or $m = 4$), by guessing the last $m - 2$ round key(s), the attack can cover $r + m$ rounds with data complexity of $2 \cdot p^{-1}$ plaintexts and time complexity of $2 \cdot p^{-1} \cdot 2^{(m-2)n}$ encryptions.

C.2 The 2-round attack

As described in Section C.1, for a 2-round attack we have a known input difference $(\Delta x_r, \Delta y_y)$ and two ciphertexts (x_{r+2}, y_{r+2}) and $(x_{r+2} \oplus \Delta x_{r+2}, y_{r+2} \oplus \Delta y_{r+2})$. The 2-round attack is to enumerate all the possible round keys k_r and k_{r+1} under which after the 2-round decryption the difference of the two ciphertexts is equal to $(\Delta x_r, \Delta y_y)$.

Since $\Delta y_{r+1} = (\Delta x_{r+2} \oplus \Delta y_{r+2}) \ggg \beta$ and $\Delta x_{r+1} = \Delta y_{r+1} \oplus (\Delta y_r \lll \beta)$ can be calculated immediately according known variables, all the differences in the 2-round scheme are determined. Similarly, the value y_{r+1} can be calculated from the known ciphertexts, whereas (x_r, y_r) and x_{r+1} remain unknown. Further, deriving k_r and k_{r+1} is equivalent to deriving x_r and x_{r+1}, because $k_{r+1} = (y_{r+1} \boxplus (x_{r+1} \ggg \alpha)) \oplus x_{r+1}$ and as $y_r = (x_{r+1} \oplus y_{r+1}) \ggg \beta$, then $k_r = (y_r \boxplus (x_r \ggg \alpha)) \oplus x_{r+1}$ can be computed as well. Thus, the key point is to deriving the values x_r and x_{r+1}.

For convenience, we omit the right circular shift $\ggg \alpha$, and then we have two differential equations of addition

$$(x_r \oplus \Delta x_r) \boxplus (y_r \oplus \Delta y_r) = (x_r \boxplus y_r) \oplus \Delta x_{r+1},$$
$$(x_{r+1} \oplus \Delta x_{r+1}) \boxplus (y_{r+1} \oplus \Delta y_{r+1}) = (x_{r+1} \boxplus y_{r+1}) \oplus \Delta x_{r+2}.$$

where all differences are known, and in the second equation y_{r+1} and $y_{r+1} \oplus \Delta y_{r+1}$ are also known.

This type of differential equations of addition has an average of 1 solution. However, for almost any value of $(\Delta x_r, \Delta y_r)$, a large part of ciphertext pairs lead to no solutions. Therefore, to save the time, a filtering process is needed before solving these two equations.

Again, the property of addition explained by Equation (1) and Equation (2) are used as filters, which means checking whether the equation holds given all differences of the 2-round scheme. Using the filter, the complexity of the 2-round attacked can be optimized to less than 2 encryptions, which was verified by a lots of experiments on SPECK in [7].

D Source Code

Copy the source code below, paste it in a file named filename.stp, and solve it with stp.

```
% inputs
x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18,
    x19 : BITVECTOR( 16 );

% intermediate variables
v0, v1, v2, v3, v4, v5, v6, v7, v8 : BITVECTOR( 16 );
```

```
% penalty variables
p0, p1, p2, p3, p4, p5, p6, p7, p8 : BITVECTOR( 16 );

% weight of the characteristic
weight : BITVECTOR( 11 );

ASSERT( (v0 = (((x1) << 9)[15:0] | ((x1) >>7))) );
ASSERT( ( (BVXOR(~( (((v0)<<1)[15:0] ) ), ((((x0)<<1)[15:0] ) )) & BVXOR(~( (((v0)
     <<1)[15:0] ) ), ((((x3)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v0), (x0) ), (
     x3) ) , (((x0)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x2 = BVXOR(x3, (((x0) << 2)[15:0] | ((x0) >>14)) ) ) );
ASSERT( ((p0 ) = (~(BVXOR(~(v0), (x0)) & BVXOR(~(v0), (x3))))) );

ASSERT( (v1 = (((x3) << 9)[15:0] | ((x3) >>7))) );
ASSERT( ( (BVXOR(~( (((v1)<<1)[15:0] ) ), ((((x2)<<1)[15:0] ) )) & BVXOR(~( (((v1)
     <<1)[15:0] ) ), ((((x5)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v1), (x2) ), (
     x5) ) , (((x2)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x4 = BVXOR(x5, (((x2) << 2)[15:0] | ((x2) >>14)) ) ) );
ASSERT( ((p1 ) = (~(BVXOR(~(v1), (x2)) & BVXOR(~(v1), (x5))))) );

ASSERT( (v2 = (((x5) << 9)[15:0] | ((x5) >>7))) );
ASSERT( ( (BVXOR(~( (((v2)<<1)[15:0] ) ), ((((x4)<<1)[15:0] ) )) & BVXOR(~( (((v2)
     <<1)[15:0] ) ), ((((x7)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v2), (x4) ), (
     x7) ) , (((x4)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x6 = BVXOR(x7, (((x4) << 2)[15:0] | ((x4) >>14)) ) ) );
ASSERT( ((p2 ) = (~(BVXOR(~(v2), (x4)) & BVXOR(~(v2), (x7))))) );

ASSERT( (v3 = (((x7) << 9)[15:0] | ((x7) >>7))) );
ASSERT( ( (BVXOR(~( (((v3)<<1)[15:0] ) ), ((((x6)<<1)[15:0] ) )) & BVXOR(~( (((v3)
     <<1)[15:0] ) ), ((((x9)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v3), (x6) ), (
     x9) ) , (((x6)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x8 = BVXOR(x9, (((x6) << 2)[15:0] | ((x6) >>14)) ) ) );
ASSERT( ((p3 ) = (~(BVXOR(~(v3), (x6)) & BVXOR(~(v3), (x9))))) );

ASSERT( (v4 = (((x9) << 9)[15:0] | ((x9) >>7))) );
ASSERT( ( (BVXOR(~( (((v4)<<1)[15:0] ) ), ((((x8)<<1)[15:0] ) )) & BVXOR(~( (((v4)
     <<1)[15:0] ) ), ((((x11)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v4), (x8) ),
     (x11) ) , (((x8)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x10 = BVXOR(x11, (((x8) << 2)[15:0] | ((x8) >>14)) ) ) );
ASSERT( ((p4 ) = (~(BVXOR(~(v4), (x8)) & BVXOR(~(v4), (x11))))) );

ASSERT( (v5 = (((x11) << 9)[15:0] | ((x11) >>7))) );
ASSERT( ( (BVXOR(~( (((v5)<<1)[15:0] ) ), ((((x10)<<1)[15:0] ) )) & BVXOR(~( (((v5)
     <<1)[15:0] ) ), ((((x13)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v5), (x10) ),
      (x13) ) , (((x10)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x12 = BVXOR(x13, (((x10) << 2)[15:0] | ((x10) >>14)) ) ) );
ASSERT( ((p5 ) = (~(BVXOR(~(v5), (x10)) & BVXOR(~(v5), (x13))))) );

ASSERT( (v6 = (((x13) << 9)[15:0] | ((x13) >>7))) );
ASSERT( ( (BVXOR(~( (((v6)<<1)[15:0] ) ), ((((x12)<<1)[15:0] ) )) & BVXOR(~( (((v6)
     <<1)[15:0] ) ), ((((x15)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v6), (x12) ),
      (x15) ) , (((x12)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x14 = BVXOR(x15, (((x12) << 2)[15:0] | ((x12) >>14)) ) ) );
ASSERT( ((p6 ) = (~(BVXOR(~(v6), (x12)) & BVXOR(~(v6), (x15))))) );

ASSERT( (v7 = (((x15) << 9)[15:0] | ((x15) >>7))) );
ASSERT( ( (BVXOR(~( (((v7)<<1)[15:0] ) ), ((((x14)<<1)[15:0] ) )) & BVXOR(~( (((v7)
     <<1)[15:0] ) ), ((((x17)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v7), (x14) ),
      (x17) ) , (((x14)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x16 = BVXOR(x17, (((x14) << 2)[15:0] | ((x14) >>14)) ) ) );
ASSERT( ((p7 ) = (~(BVXOR(~(v7), (x14)) & BVXOR(~(v7), (x17))))) );

ASSERT( (v8 = (((x17) << 9)[15:0] | ((x17) >>7))) );
ASSERT( ( (BVXOR(~( (((v8)<<1)[15:0] ) ), ((((x16)<<1)[15:0] ) )) & BVXOR(~( (((v8)
     <<1)[15:0] ) ), ((((x19)<<1)[15:0] ))) ) & ( BVXOR( BVXOR( BVXOR( (v8), (x16) ),
      (x19) ) , (((x16)<<1)[15:0] ) ) ) = 0hex0000));
ASSERT( (x18 = BVXOR(x19, (((x16) << 2)[15:0] | ((x16) >>14)) ) ) );
ASSERT( ((p8 ) = (~(BVXOR(~(v8), (x16)) & BVXOR(~(v8), (x19))))) );

% No all-zero characteristic :
ASSERT( NOT( (x0|x1) = 0hex0000 ) );

ASSERT( (weight = ( BVPLUS(11, 0b0000000@(BVPLUS( 4, 0bin000@( p0[0:0]) , 0bin000@(p0
     [ 1:1] ), 0bin000@( p0[2:2]), 0bin000@(p0[3:3]), 0bin000@( p0[4:4]),0bin000@( p0
     [5:5]), 0bin000@( p0[6:6]), 0bin000@(p0[7:7]), 0bin000@( p0[8:8]),0bin000@( p0
     [9:9]), 0bin000@( p0[10:10]),0bin000@(p0[11:11]), 0bin000@(p0[12:12]),0bin000@(
     p0[13:13] ), 0bin000@(p0[14:14]))), 0b0000000@(BVPLUS( 4, 0bin000@( p1[0:0]) , 0
     bin000@(p1[ 1:1] ), 0bin000@( p1[2:2]), 0bin000@(p1[3:3]), 0bin000@( p1[4:4]),0
     bin000@( p1[5:5]), 0bin000@( p1[6:6]), 0bin000@(p1[7:7]), 0bin000@( p1[8:8]),0
     bin000@( p1[9:9]), 0bin000@( p1[10:10]),0bin000@(p1[11:11]), 0bin000@(p1[12:12])
     ,0bin000@(p1[13:13] ), 0bin000@(p1[14:14]))), 0b0000000@(BVPLUS( 4, 0bin000@( p2
     [0:0]) , 0bin000@(p2[ 1:1] ), 0bin000@( p2[2:2]), 0bin000@(p2[3:3]), 0bin000@(
     p2[4:4]),0bin000@( p2[5:5]), 0bin000@( p2[6:6]), 0bin000@(p2[7:7]), 0bin000@( p2
     [8:8]),0bin000@( p2[9:9]), 0bin000@( p2[10:10]),0bin000@(p2[11:11]), 0bin000@(p2
     [12:12]),0bin000@(p2[13:13] ), 0bin000@(p2[14:14]))), 0b0000000@(BVPLUS( 4, 0
     bin000@( p3[0:0]) , 0bin000@(p3[ 1:1] ), 0bin000@( p3[2:2]), 0bin000@(p3[3:3]),
     0bin000@( p3[4:4]),0bin000@( p3[5:5]), 0bin000@( p3[6:6]), 0bin000@(p3[7:7]), 0
     bin000@( p3[8:8]),0bin000@( p3[9:9]), 0bin000@( p3[10:10]),0bin000@(p3[11:11]),
     0bin000@(p3[12:12]),0bin000@(p3[13:13] ), 0bin000@(p3[14:14]))), 0b0000000@(
     BVPLUS( 4, 0bin000@( p4[0:0]) , 0bin000@(p4[ 1:1] ), 0bin000@( p4[2:2]), 0
```

```
    bin000@(p4[3:3]), 0bin000@( p4[4:4]),0bin000@( p4[5:5]), 0bin000@( p4[6:6]), 0
    bin000@(p4[7:7]), 0bin000@( p4[8:8]),0bin000@( p4[9:9]), 0bin000@( p4[10:10]),0
    bin000@(p4[11:11]), 0bin000@(p4[12:12]),0bin000@(p4[13:13] ), 0bin000@(p4
    [14:14]))), 0b0000000@(BVPLUS( 4, 0bin000@( p5[0:0]) , 0bin000@(p5[ 1:1] ), 0
    bin000@( p5[2:2]), 0bin000@(p5[3:3]), 0bin000@( p5[4:4]),0bin000@( p5[5:5]), 0
    bin000@( p5[6:6]), 0bin000@(p5[7:7]), 0bin000@( p5[8:8]),0bin000@( p5[9:9]), 0
    bin000@( p5[10:10]),0bin000@(p5[11:11]), 0bin000@(p5[12:12]),0bin000@(p5[13:13]
    ), 0bin000@(p5[14:14]))), 0b0000000@(BVPLUS( 4, 0bin000@( p6[0:0]) , 0bin000@(p6
    [ 1:1] ), 0bin000@( p6[2:2]), 0bin000@(p6[3:3]), 0bin000@( p6[4:4]),0bin000@( p6
    [5:5]), 0bin000@( p6[6:6]), 0bin000@(p6[7:7]), 0bin000@( p6[8:8]),0bin000@( p6
    [9:9]), 0bin000@( p6[10:10]),0bin000@(p6[11:11]), 0bin000@(p6[12:12]),0bin000@(
    p6[13:13] ), 0bin000@(p6[14:14]))), 0b0000000@(BVPLUS( 4, 0bin000@( p7[0:0]) , 0
    bin000@(p7[ 1:1] ), 0bin000@( p7[2:2]), 0bin000@(p7[3:3]), 0bin000@( p7[4:4]),0
    bin000@( p7[5:5]), 0bin000@( p7[6:6]), 0bin000@(p7[7:7]), 0bin000@( p7[8:8]),0
    bin000@( p7[9:9]), 0bin000@( p7[10:10]),0bin000@(p7[11:11]), 0bin000@(p7[12:12])
    ,0bin000@(p7[13:13] ), 0bin000@(p7[14:14]))), 0b0000000@(BVPLUS( 4, 0bin000@( p8
    [0:0]) , 0bin000@(p8[ 1:1] ), 0bin000@( p8[2:2]), 0bin000@(p8[3:3]), 0bin000@(
    p8[4:4]),0bin000@( p8[5:5]), 0bin000@( p8[6:6]), 0bin000@(p8[7:7]), 0bin000@( p8
    [8:8]),0bin000@( p8[9:9]), 0bin000@( p8[10:10]),0bin000@(p8[11:11]), 0bin000@(p8
    [12:12]),0bin000@(p8[13:13] ), 0bin000@(p8[14:14])))) ) ) );

ASSERT( BVLE( weight, 0b00000011110) );

QUERY(FALSE);

COUNTEREXAMPLE;
```