Automatic Search of Linear Trails in ARX with Applications to SPECK and Chaskey

Yunwen Liu[1,2], Qingju Wang[1,3(✉)], and Vincent Rijmen[1]

[1] ESAT/COSIC, KU Leuven and iMinds, Leuven, Belgium
qingju.wang@esat.kuleuven.be
[2] Department of Science, National University of Defence Technology, Changsha, China
[3] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

Abstract. In this paper, we study linear cryptanalysis of the ARX structure by means of automatic search. To evaluate the security of ARX designs against linear cryptanalysis, it is crucial to find (round-reduced) linear trails with maximum correlation. We model the problem of finding optimal linear trails by the boolean satisfiability problem (SAT), translate the propagation of masks through ARX operations into bitwise expressions and constraints, and then solve the problem using a SAT solver. We apply the method to find optimal linear trails for round-reduced versions of the block cipher SPECK and the MAC algorithm Chaskey. For SPECK with block size 32/48/64/96/128, we can find optimal linear trails for 22/11/13/9/9 rounds respectively, which largely improves previous results, especially on larger versions. A 3-round optimal linear trail of Chaskey is presented for the first time as far as we know. In addition, our method can be used to enumerate the trails in a linear hull, and we present two linear hulls with the distributions of trails for round-reduced SPECK32. Our work provides designers with more accurate evaluation against linear cryptanalysis on ARX designs, especially for primitives with large block sizes and many rounds.

Keywords: Linear cryptanalysis · ARX structure · Boolean satisfiability problem

1 Introduction

Many symmetric key primitives are proposed with the ARX design strategy which only uses three operations: Additions ($\boxplus$), Rotations ($\lll$) and XORs ($\oplus$). These operations are very simple and efficient in software implementation, but interactively provide non-linearity. The ARX structure can be found in a large number of symmetric key designs, including hash functions BLAKE [2] and Skein [9], which are two of the five SHA-3 finalists, stream ciphers such as Salsa20 [5] and ChaCha [4], block ciphers such as TEA [27], XTEA [18], HIGHT [12] and SPECK [3], and MAC algorithm Chaskey [16]. Even though the ARX

M. Manulis et al. (Eds.): ACNS 2016, LNCS 9696, pp. 485–499, 2016.
DOI: 10.1007/978-3-319-39555-5_26

structure receives a considerable amount of attention due to its elegance and efficiency, it remains a difficult problem to evaluate its security margin against known attacking techniques.

Differential cryptanalysis [6] and linear cryptanalysis [15] are two main techniques used in the analysis of symmetric primitives, including ARX designs. Differential characteristics (resp. linear trails) with optimal probability (resp. correlation) can lead to efficient attacks with complexity better than the brute force searching. Hence the resistance against differential cryptanalysis and linear cryptanalysis is a crucial feature to consider for both designers and attackers. Among the methods and algorithms proposed in finding good differential characteristics and linear trails, automatic searching is a popular and efficient way. Several automatic toolkits dedicated to the searching of differential characteristics in ARX are proposed in the literature [7,14]. Comparing to the significant efforts which have been dedicated to the automatic search of differential characteristics, the searching tool of linear trails in ARX designs fell behind. The first paper on this topic as far as we know is presented by Yao *et al.* [28], where an algorithm based on branch and bound is used to find optimal (round-reduced) linear trails in SPECK32, and short linear trails of larger versions of SPECK.

Our motivation is to model the problem of searching optimal linear trails in an ARX structure as a boolean satisfiability problem [24]. The boolean satisfiability problem is widely used to determine whether the boolean variables in a given set of boolean conditions have valid assignments such that the conditions evaluate to TRUE. Specifically, in order to construct linear trails with nonzero correlation, the idea is to explore the bit-level conditions on the bits of the masks when passing through every operation of an ARX structure, render them into boolean satisfiability language, and call solvers to obtain valid linear trails with certain correlations. Our work can be applied to general ARX designs and has good performance in finding linear trails with best correlation for round-reduced primitives. Therefore it could provide a rigorous security evaluation for some ARX primitives against linear cryptanalysis.

Table 1. The number of covered rounds in finding optimal linear trails for SPECK family and Chaskey

Cipher	#covered rounds [28]	#covered rounds this paper	#total rounds
SPECK32	22	22	22
SPECK48	7	11	22/23
SPECK64	5	13	26/27
SPECK96	4	9	28/29
SPECK128	4	9	32/33/34
Chaskey	-	3	8

In this paper, our method is applied to the linear cryptanalysis of round-reduced SPECK family and Chaskey. Table 1 gives an overview of the number

of rounds for which optimal linear trails are found in SPECK and Chaskey. Note that there is no previous research on finding optimal linear trails in round-reduced Chaskey.

This paper is organised as follows. In Sect. 2, we recall linear cryptanalysis and the boolean satisfiability problem. We study the propagation of bits in masks through operations of the ARX structure and transform them using boolean satisfiability language such that they can be solved automatically in Sect. 3. In Sect. 4, we apply the method to block cipher SPECK and MAC algorithm Chaskey, and find linear hulls for round-reduced SPECK32. Finally, we conclude in Sect. 5.

2 Preliminaries

We denote an n-bit boolean vector by $x = (x_{n-1}, \cdots, x_1, x_0)$, where x_0 is the least significant bit. For two n-bit boolean vectors x and y, the inner product is $x \cdot y = \bigoplus_{i=0}^{n-1} x_i y_i$. The partial order $\preccurlyeq$ is defined by $x \preccurlyeq y \Leftrightarrow x_i \leq y_i, \forall i \in \{0, \cdots, n-1\}$. The characteristic function $1_{x \preccurlyeq y}$ is defined as

$$1_{x \preccurlyeq y} = \begin{cases} 1, & \text{if } x \preccurlyeq y, \\ 0, & \text{otherwise.} \end{cases}$$

Logical operations OR, AND, NOT, XOR are referred to as $\vee, \wedge, \neg, \oplus$, respectively. All linear masks are hexadecimal, and we omit the 0x symbol.

2.1 Linear Cryptanalysis

Linear cryptanalysis investigates linear relations among the parities of plaintext, ciphertext and the secret key. Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}$ be a vectorial boolean function. Assume that masks for input x and output $f(x)$ are Γ_{in} and Γ_{out}. The *correlation* of the linear approximation is defined as

$$C(\Gamma_{in}, \Gamma_{out}) = 2 \cdot \Pr(\Gamma_{in} \cdot x \oplus \Gamma_{out} \cdot f(x) = 0) - 1.$$

Equivalently, the correlation can also be written as a Walsh transformation,

$$C(\Gamma_{in}, \Gamma_{out}) = 2^{-n} \sum_{x \in \mathrm{GF}(2^n)} (-1)^{\Gamma_{in} \cdot x \oplus \Gamma_{out} \cdot f(x)}.$$

Let $g = f_{r-1} \circ \cdots \circ f_1 \circ f_0$ be an iterated permutation which is the composition of r round functions f_i. Linear approximations (γ_i, γ_{i+1}) of a single round f_i can be concatenated into a *linear trail* $(\gamma_0, \gamma_1, \cdots, \gamma_r)$ of g.

Lemma 1 ([8]). *Let $(\gamma_0, \gamma_1, \cdots, \gamma_r)$ be a linear trail of an iterated permutation. Then the correlation of the linear trail can be calculated as*

$$C(\gamma_0, \gamma_r) = \prod_{i=0}^{r-1} C(\gamma_i, \gamma_{i+1})$$

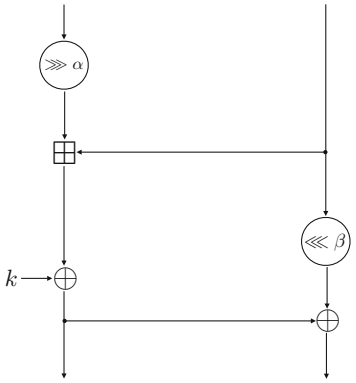

Fig. 1. Round function of SPECK

We call a linear trail over a (round-reduced) cipher with maximum correlation amplitude an *optimal linear trail*.

线性壳，仅仅给定输入掩码和输出掩码（和差分类似）

A linear approximation $(\Gamma_{in}, \Gamma_{out})$ of a block cipher is called a linear hull [19], which contains all linear trails with input mask Γ_{in} and Γ_{out}. The *potential* (averaged linear probability over the key space $\mathcal{K}$) of a linear hull is defined as

$$ALP(\Gamma_{in}, \Gamma_{out}) = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} C(\Gamma_{in}, \Gamma_{out})^2,$$

and gives the expected value of the data complexity of a linear attack.

2.2 Description of SPECK and Chaskey

The lightweight block cipher SPECK family was designed by the NSA in 2013. The block sizes are defined as $2n$ with $n \in \{16, 24, 32, 48, 64\}$, and key size as mn with $m \in \{2, 3, 4\}$ depending on n. The instances corresponding to a block size $2n$ and key size mn are denoted by SPECK$2n/mn$. Since we do not explore the key schedule in this paper, the instances of SPECK will simply be referred to as SPECK$2n$. The round function of SPECK with inputs x and y, a round key k is defined as:

$$F_k(x, y) = (f_k(x, y), f_k(x, y) \oplus (y \lll \beta))$$

where $f_k(\cdot, \cdot)$ is defined as $f_k(x, y) = ((x \ggg \alpha) \boxplus y) \oplus k$, the rotation offset (α, β) is $(7, 2)$ for SPECK32, and $(8, 3)$ for the larger instances. One round of SPECK is depicted in Fig. 1. For more details, we refer to the design [3].

Chaskey is a permutation-based MAC algorithm presented by Mouha *et al.* in 2014. The underlying permutation is an Even-Mansour block cipher with the ARX structure. The block size is 128-bits, which is separated into four 32-bit words. The design of Chaskey is inspired by Siphash [1], and has a structure similar to the block cipher Threefish [9]. The total number of rounds is 8, and

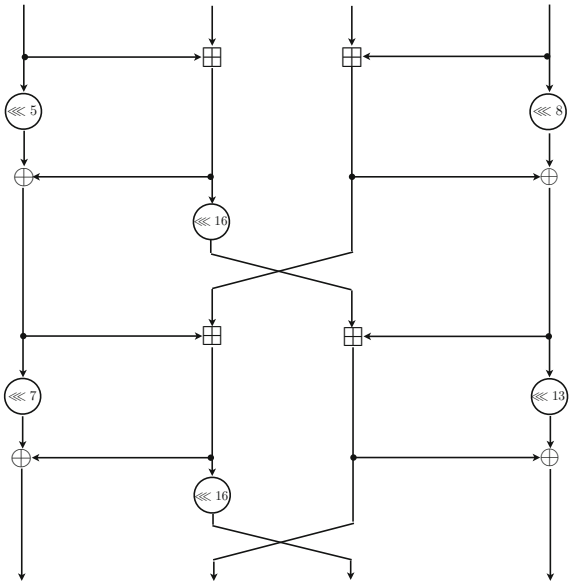

Fig. 2. Round function of Chaskey

there are four modular addition operations and some rotation operations in each round. The round function of the Chaskey permutation is showed in Fig. 2.

2.3 Boolean Satisfiability Problem

The boolean satisfiability problem is often called SAT. It considers whether there is a valid assignment to boolean variables satisfying a given set of boolean clauses. A Boolean clause consists of boolean variables (called literals), operators AND, OR, NOT, and parentheses. For example, the clause x AND (NOT y) is satisfiable since x = TRUE, y = FALSE is a valid assignment.

The SAT problem is NP-complete. However for most practical situations, the solutions can be found in reasonable time. There are a large number of heuristic SAT solvers, and all of them accept DIMACS CNF (Conjunctive Normal Form) files as the standard input format. In CNF format, all clauses are literals with logical operation OR and NOT, while the clauses are concatenated by AND. The output is either *satisfiable* or *unsatisfiable*, when satisfiable, the solver can also return a valid assignment to all literals. More specifically, SAT solvers will start searching with an initial assignment, then calculate the number of conflicting clauses, based on which the search tree of the SAT solver decides the next step of searching to eliminate possible conflicts until a valid or no solution is found. It is believed that, for cryptographic problems, the time for *unsatisfiable* decision is much longer than that of *satisfiable*, because the search is roughly brute-force before returning the decision of *unsatisfiable* [23].

In some applications, we also consider arithmetic operations, for instance, the arithmetic sum of boolean variables, which leads to the satisfiability modulo theory (SMT) problem. SMT has certain similarity with the 0-1 integer programming problem or mixed integer linear programming (MILP), while the underlying ideas to solve them differ significantly. For the MILP problem, linear programming solvers first regard the problem as a general linear programming problem in real numbers, then by Branch and Cut, they carefully rule out illegal branches and then limit the solution to 0-1 integers. SMT solvers try to translate the problem to SAT, then solve it within a binary field. Due to the different methodologies of solvers, the performances depend heavily on the background and structure of the underlying problem.

3 Translating Clauses for Modular Addition

The behaviours of masks through linear operations are easy to describe, since the correlation is either zero or ± 1. For example, with input masks Γ_a, Γ_b and output mask Γ_c, the condition for being a linear approximation of XOR with nonzero correlation is $\Gamma_a = \Gamma_b = \Gamma_c$. The condition for being a nontrivial linear approximation of three-fork branching is $\Gamma_a \oplus \Gamma_b \oplus \Gamma_c = 0$, and the conditions for rotational circular shift is the equality on each corresponding bit of masks.

However for the nonlinear operation modular addition, it is necessary to have a better understanding on the nature of addition modulo 2^n.

3.1 Propagation of Masks Through Modular Addition

The milestone works on linear correlation of modular addition are by Wallén *et al.* [20,26]. They propose a recursive method to calculate the correlation of a linear approximation in addition modulo 2^n efficiently by an automaton. The only drawback of the recursive automaton is that it is very difficult to translate the expression into bit-level linear relations in masks, i.e. every bit is dependent on all previous bits, which leads to a huge number of complex constraints. Therefore, even though there are several papers discussing the heuristic search methods of differentials, no previous result is on finding linear trails in ARX ciphers with SAT theory.

In order to avoid the recursive expression, an explicit result on calculating the correlation of linear approximations in modular addition is proven by Schulte-Geers [21]. Despite the recursive property of the carry, modular addition is CCZ-equivalent to a vectorial quadratic boolean function. A more natural formula to calculate the correlation in addition modulo 2^n is given in Proposition 1.

Proposition 1 ([21]). *Let z be an n-bit vector satisfying $z \oplus (z \gg 1) \oplus ((u \oplus v \oplus w) \gg 1) = 0$, $z_{n-1} = 0$, where u is the output mask, v, w are the input masks in a linear approximation of addition modulo 2^n. Then the correlation of the linear approximation is given by*

$$cor(u, v, w) = 1_{u \oplus v \preccurlyeq z} 1_{u \oplus w \preccurlyeq z} (-1)^{(u \oplus w) \cdot (u \oplus v)} 2^{-|z|}.$$

Comparing to a recursive algorithm, the Hamming weight of z determines the amplitude of the correlation directly, while each bit of z can be explicitly calculated from input and output masks. Next, we will mainly focus on the absolute value of the correlation.

From Proposition 1, to obtain a valid linear approximation, the input masks v, w and output mask u through addition modulo 2^n need to follow the constraints below.

$$\begin{aligned} z_{n-1} &= 0, \\ z_{n-2} &= u_{n-1} \oplus v_{n-1} \oplus w_{n-1}, \\ z_j &= z_{j+1} \oplus u_{j+1} \oplus v_{j+1} \oplus w_{j+1}, \\ z_i &\geq u_i \oplus v_i, \\ z_i &\geq u_i \oplus w_i, \end{aligned} \tag{1}$$

where $0 \leq i \leq n-1$, $0 \leq j \leq n-3$.

3.2 From Linear Relations Towards SATisfiability

When considering problems in cryptanalysis, XOR is one of the most common operations. If we translate XOR clauses into CNF, a sentence $a \oplus b$ becomes two clauses $(\neg a \vee \neg b) \wedge (a \vee b)$. In general, the XOR of n boolean variables will give 2^{n-1} clauses in CNF format. Even if the expressions are logically equivalent, the underlying structure of the XOR equation system is missing in terms of the CNF format. A system of XOR equations is in fact a linear equation system on GF(2), therefore, it can be solved by Gaussian elimination in time $O(n^3)$, where n is the number of variables. In many circumstances, Gaussian elimination is much more efficient than translating XOR into operations $\vee$ and $\wedge$. One SAT solver called Cryptominisat4 [23] is specially designed to be compatible with XOR operations and solve the XOR equation system by Gaussian elimination.

The remaining constraints in Eq. (1) are inequalities. Consider the inequality in boolean variables, $z \geq a \oplus b$. It is equivalent to *if* $a \oplus b$*, then* z, which is logically consistent with $(\neg a \vee b \vee z) \wedge (a \vee \neg b \vee z)$.

Recall that in order to find good linear trails with large correlation values, we need to minimize the Hamming weight of z. By the piling-up lemma, the sum of z in every round $\sum_{i,r} z_i^r$ is the objective function to be minimized. Addition over integers is an unnatural operation in SAT language, which is not easy to describe with only OR and AND. In SAT/SMT theory, Constraints like objective function $\sum_i x_i \leq k$, where $k \geq 1$, are called cardinality constraints, which belongs to an even larger class called Pseudo Boolean constraints (PB-constraints). There are two directions to handle the cardinality constraints: one is to develop new PB-solvers dedicated to cardinality constraints, the other one is to convert cardinality constraints into CNF format, which is what we adopt in this paper.

One plain method is enumerating all the possible combinations of no more than k out of n variables being true, i.e. the conjunction of $\binom{n}{k+1}$ clauses $\bigwedge_{i_1,\ldots,i_{k+1}} (\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}})$. However it is not applicable when n, k are large. Throughout the literature, a large number of methods to encode the cardinality

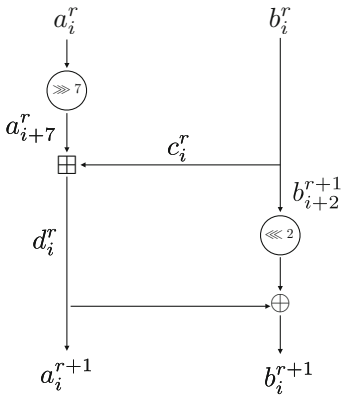

Fig. 3. Notation of masks in round function of SPECK32

constraints are presented. The basic idea is to add new variables to reduce the number of constraints. Since it is a tradeoff between the number of new variables needed and the number of clauses, while the sizes of variables and clauses both have a significant influence on the efficiency of solving, it is critical to find a good encoding method. In this paper, we use sequential encoding method [22], as shown in Eq. (2). For $\sum_i x_i \leq k$, new dummy variables $\{u_{i,j}\}_{1\leq i\leq n-1, 1\leq j\leq k}$ are introduced to return contradiction when the cardinality is larger than k.

$$\begin{cases} (\neg x_1 \vee u_{1,1}) \wedge (\neg u_{1,j}), \\ (\neg x_i \vee u_{i,1}) \wedge (\neg u_{i-1,1} \vee u_{i,1}) \wedge (\neg x_i \vee \neg u_{i-1,j-1} \vee u_{i,j}) \\ \qquad \wedge (\neg u_{i-1,j} \vee u_{i,j}) \wedge (\neg x_i \vee \neg u_{i-1,k}), \\ \neg x_n \vee \neg u_{n-1,k}, \end{cases} \tag{2}$$

where $1 < j \leq k, 1 < i < n$. The sequential encoding of cardinality constraints is one of the best methods, with relatively small amount of additional variables and a great reduction of clauses.

When $k = 0$, all variables are zero, which can be translated to n clauses as $\neg x_i$, $1 \leq i \leq n$.

4 Applications

4.1 Application to the SPECK Family

For simplicity, we take SPECK32 as an illustration. Figure 3 shows the notation of the masks in round r. From Eq. (1), we can derive the constraints on linear approximation of SPECK32 in round r as

$$
\begin{aligned}
z_{15}^r &= 0,\\
z_{14}^r &= a_6^r \oplus c_{15}^r \oplus d_{15}^r,\\
z_j^r &= z_{j+1}^r \oplus a_{j+8}^r \oplus c_{j+1}^r \oplus d_{j+1}^r,\\
z_i^r &\geq a_{i+7}^r \oplus d_i^r,\\
z_i^r &\geq c_i^r \oplus d_i^r,\\
d_i^r &= a_i^{r+1} \oplus b_i^{r+1},\\
c_i^r &= b_i^r \oplus b_{i+2}^{r+1},
\end{aligned}
\tag{3}
$$

where $0 \leq i \leq 15$, $0 \leq j \leq 13$, and $\sum_{r,i} z_i^r$ is to be minimized.

Since usually the time for *unsatisfiable* decision is much longer than that for *satisfiable*, we follow Algorithm 1 below to find linear trails with optimal correlation, which ensures that the most time-consuming part *unsatisfiable* only appears once during the search.

Algorithm 1. Find optimal linear trail

Input: An optimal linear trail L with correlation $2^{-\ell}$ of an r round-reduced cipher
Output: The correlation of the optimal linear trail in $r+1$ round-reduced cipher

1: Append a 1-round trail at the end of L to extend it into a $r+1$ round valid linear trail L' with correlation $2^{-\ell'}$
2: **while** the problem is *satisfiable* with $\sum_{r,i} z_i^r \leq \ell'$ **do**
3: $\ell' \leftarrow \ell' - 1$
return $2^{-(\ell'+1)}$

Table 2 gives an overview of the correlation of optimal linear trails in round-reduced SPECK ciphers.[1] We confirm all the correlations of optimal linear trails in [28]. Moreover, our method covers significantly more rounds in larger versions of SPECK: 11/13/9/9 rounds comparing to 7/5/4/4 rounds in previous paper [28] for SPECK48/64/96/128.

We also show examples of linear trails with best correlation for round-reduced SPECKs in Table 3. Sometimes without further constraints, input and output masks may have very high Hamming weight. By setting cardinality constraints on the Hamming weights of the masks, we can obtain trails with input and output masks of the lowest Hamming weight under a given correlation and number of rounds, an example is the linear trail of 11-round SPECK32 in Table 3.

4.2 Application to Chaskey

The designers of Chaskey did not give a security evaluation against linear cryptanalysis in their paper. Using our method, we are able to find the correlation

[1] Our experiments for searching optimal linear trails are performed on a PC with 8 Intel® Core™ i7 clocked at 3.40 GHz. In order to speed up the searching for linear hulls by utilising the parallel mode in Cryptominisat4, we run the program on a cruncher with 40 Intel® Xeon™ E5-2687W v3 clocked at 3.1 GHz.

Table 2. Correlation of best linear trail in SPECK family.

R	SPECK32	R	SPECK32	R	SPECK48	SPECK64	SPECK96	SPECK128
1	1	12	2^{-20}	1	1	1	1	1
2	1	13	2^{-22}	2	1	1	1	1
3	2^{-1}	14	2^{-24}	3	2^{-1}	2^{-1}	2^{-1}	2^{-1}
4	2^{-3}	15	2^{-26}	4	2^{-3}	2^{-3}	2^{-3}	2^{-3}
5	2^{-5}	16	2^{-28}	5	2^{-6}	2^{-6}	2^{-6}	2^{-6}
6	2^{-7}	17	2^{-30}	6	2^{-8}	2^{-9}	2^{-9}	2^{-9}
7	2^{-9}	18	2^{-34}	7	2^{-12}	2^{-13}	2^{-13}	2^{-13}
8	2^{-12}	19	2^{-36}	8	2^{-15}	2^{-17}	2^{-18}	2^{-18}
9	2^{-14}	20	2^{-38}	9	2^{-19}	2^{-19}	2^{-22}	2^{-22}
10	2^{-17}	21	2^{-40}	10	2^{-22}	2^{-21}		
11	2^{-19}	22	2^{-42}	11	2^{-25}	2^{-24}		
				12		2^{-27}		
				13		2^{-30}		

of the best linear trail for the round-reduced Chaskey permutation, as shown in Table 4. Table 5 is an example trail for 3-round Chaskey. Notations a, b, c, d are the masks on each 32-bit branch.

4.3 Enumerating Linear Trails in a Linear Hull

For most SAT solvers, if the problem is *satisfiable*, they can print all the solutions. However, due to the additional variables introduced by encoding methods in generating the CNF files, the solvers may output duplicated solutions which represent the same trail, as also observed by Kölbl *et al.* in [13]. To avoid inaccuracy, we generate the solutions one by one:

Step 1: Generate the CNF file for the problem, ask the solver to give one solution $\bar{s}$ if it exists.
Step 2: Append a new clause to the current CNF file in order to rule out $\bar{s}$.
Step 3: Ask solver to give a solution, repeat step 2 until the solver returns *unsatisfiable*.

In Table 6, we give the best linear hulls found and their corresponding distribution of trails for 9-round, 10-round SPECK32, where ALP is the estimated averaged linear probability. The experimental average ALP with 128 random keys for the above linear hulls are $2^{-28.9}$ and $2^{-31.1}$ respectively.

4.4 Comparison of Solvers

In some previous papers on automatic searching of differential and linear trails, e.g. [17,25], the searching idea is modelled as a MILP problem and solved

Table 3. Linear trail with best correlation in reduced-round SPECK.

R	SPECK32		SPECK48		SPECK64	
1	4000	00b0	800121	158021	00101800	00001812
2	0000	00c0	018100	200101	00001000	00000010
3	0300	0300	000100	000001	00000018	00000000
4	0c1e	0818	000001	000000	d8000000	c0000000
5	f000	d010	098000	080000	04100006	04800006
6	4683	4743	406100	406800	0026d030	0420c030
7	00a0	0629	00024b	00420a	01070101	21073781
8	78a0	18a1	001040	5e1042	01b00100	00318601
9	0090	6021	9082c0	f082d0	01800001	0181b000
10	6080	4081	000018	80d09b	01000000	00018000
11	0080	0001	de84dc	c684dc	00010000	00000000
12	0001	0000			00000d00	00000c00
13					00006065	00006068

R	SPECK96		SPECK128	
1	000001800120	140000018021	0000000001800120	1400000000018021
2	000000018100	200000000101	0000000000018100	2000000000000101
3	000000000100	000000000001	0000000000000100	0000000000000001
4	000000000001	000000000000	0000000000000001	0000000000000000
5	098000000000	080000000000	0d00000000000000	0c00000000000000
6	404000000000	404800000000	6040000000000000	604c000000000000
7	000000000002	004000000002	0000000000000003	0060000000000003
8	180000000010	1a0000000010	1800000000000018	1b00000000000018
9	009000000080	108000000080	00900000000000c0	18800000000000c0
10	440458000404	840480000404	0000000004045e06	c404800000000606
11				

Table 4. Correlation of optimal linear trails in round-reduced Chaskey.

R	1	2	3	4
Best cor.	2^{-1}	2^{-2}	2^{-9}	-

Table 5. A linear trail with optimal correlation in 3-round Chaskey.

R	a	b	c	d
1	00000020	00000000	0001800d	08018189
2	00000000	00000000	00010000	00010000
3	00800000	00000000	00000081	00000000
4	0260c080	18208006	01010260	18208000

Table 6. The Distribution of linear trails in best found 9-/10-round SPECK32 linear hull.

Cor.	9-round[a] #trails	Cor.	10-round[b] #trails
2^{-14}	0	2^{-17}	1
2^{-15}	1	2^{-18}	1
2^{-16}	0	2^{-19}	6
2^{-17}	3	2^{-20}	16
2^{-18}	2	2^{-21}	81
2^{-19}	21	2^{-22}	344
2^{-20}	69	2^{-23}	1298
2^{-21}	346	2^{-24}	4873
2^{-22}	1196	2^{-25}	17781
2^{-23}	4461	2^{-26}	$\geq$60480
2^{-24}	15241	2^{-27}	$\geq$23951
2^{-25}	48397	2^{-28}	$\geq$11272
2^{-26}		2^{-29}	$\geq$3789
2^{-27}		2^{-30}	$\geq$5883
2^{-28}		2^{-31}	$\geq$48951
ALP	$2^{-29.1}$	ALP	$\geq 2^{-32.1}$

[a] input masks: 0010, 1400, output masks: 0b00, 0800
[b] input masks: 0000, 0306, output masks: 0b00, 0800

by CPLEX. To compare the performance of CPLEX and Cryptominisat4, we encode the same constraints with MILP language and CNF without optimisation. Despite the connection between the MILP and the SAT problem with an objective function, our method has an advantage over CPLEX. For instance, to find an optimal linear trail in 6-round SPECK32, it takes over 4000 s on CPLEX, comparing to about 2 s on Cryptominisat4.[2]

Another commonly used solver is STP [11], which is a SMT solver and also a CNF generator. It can encode constraints into CNF file inside the solver based on SMTLIB2 language, and then call a SAT solver to solve the problem. Unlike Cryptominisat4, STP does not support XOR clauses and Gaussian elimination, therefore all clauses involving XOR are translated into standard CNF format. Thus, with exactly the same constraints derived in Sect. 3.1, we generate different

[2] Recently, the MILP-based method was applied to the search of differential characteristics and linear trails of SPECK [10]. The formulae describing the linear approximations differ from those of this paper, and dedicated technics are used to improve their search. In addition, the authors concatenate two or three shorter linear trails to attack more rounds, while this paper focuses on finding optimal trails in reduced-round primitives.

Table 7. Comparison between the runtime of CNF files generated by Sect. 3 and STP on the searching problems of SPECK128.

	Section 3		STP	
Round	time1	time2	time1	time2
4	0.05s	0.09s	2s	2s
5	0.8s	1s	4s	7s
6	8s	10s	18s	19s
7	4m44s	1m56s	6m2s	4m20s
8	2s	643m55s	55m4s	114m26s
9	53m51s	16523m	10m27s	12184m

CNF files encoded by STP and our method, and compare their performances on the searching problem of SPECK by considering the number of variables and clauses in corresponding CNF file, as well as the run time for getting optimal linear trails and *unsatisfiable* decision. Both CNF files run on Cryptominisat4.

In most cases, the CNF file encoded by our method has a smaller number of variables and clauses than the STP-generated ones, and the difference can be two times for problems in SPECK with larger block sizes. Although the size of the problem and the speed of solving are not strictly proportional, in general, less variables and clauses are preferable. Table 7 shows the comparison between the runtime of CNF files generated by the method in Sect. 3 and STP solver, where time1 is the time to find an optimal linear trail, and time2 is the time to return *unsatisfiable*. In general, the performance of both methods is comparable. However it is interesting to notice that, it takes 2 s to find one optimal trail for 8-round SPECK128 by our method while STP uses around one hour. It shows that the performance of CNF files depends heavily on the encoding method and the underlying problem, therefore our method may provide an alternative way to solve problems which are not solvable using other solvers.

5 Conclusion

In this paper, we focus on how to find linear trails with optimal correlations in the ARX structures. We model the question as a boolean satisfiability problem, translate the propagation of masks through ARX operations into bitwise expressions and CNF constraints, and then solve the problem by SAT solvers. We apply the automatic search method to the block cipher SPECK and MAC proposal Chaskey, and obtain the correlation of optimal linear trails for 22/11/13/9/9-round reduced SPECK32/48/64/96/128 and 3-round Chaskey, where the analysis of optimal linear trails on Chaskey is presented for the first time so far. In addition, our method is applied to enumerate linear trails in two linear hulls of 9-round and 10-round SPECK32.

Our work provides a searching tool with improved performance towards analysing the security of ARX designs against linear cryptanalysis, which is meaningful to both designers and attackers.

Acknowledgements. We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007) and the Research Fund KU Leuven OT/13/071. Yunwen Liu is partially supported by the China Scholarship Council. Qingju Wang is in part sponsored by National Natural Science Foundation of China (61472250, U1536103) and Major State Basic Research Development Program (973 Plan) of China (2013CB338004).

References

1. Aumasson, J.-P., Bernstein, D.J.: SipHash: a fast short-input PRF. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 489–508. Springer, Heidelberg (2012)
2. Aumasson, J.-P., Henzen, L., Meier, W., Phan, R.C.W.: SHA-3 proposal BLAKE. Submission to NIST (2008)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, DAC 2015, pp. 175:1–175:6. ACM (2015)
4. Bernstein, D.J.: ChaCha, a variant of Salsa20. http://cr.yp.to/chacha.html
5. Bernstein, D.J.: The Salsa20 family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 84–97. Springer, Heidelberg (2008)
6. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991)
7. Biryukov, A., Velichkov, V.: Automatic search for differential trails in ARX ciphers. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 227–250. Springer, Heidelberg (2014)
8. Daemen, J., Govaerts, R., Vandewalle, J.: Correlation matrices. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 275–285. Springer, Heidelberg (1995)
9. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family. Submission to NIST (round 3) (2010)
10. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-based automatic search algorithms for differential and linear trails for SPECK. In: Fast Software Encryption, FSE 2016. Springer (2016, to appear)
11. Ganesh, V.: STP constraint solver: Simple theorem prover SMT solver. http://stp.github.io
12. Hong, D., Sung, J., Hong, S.H., Lim, J.-I., Lee, S.-J., Koo, B.-S., Lee, C.-H., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J.-S., Chee, S.: HIGHT: a new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
13. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 161–185. Springer, Heidelberg (2015)

14. Leurent, G.: Construction of differential characteristics in ARX designs application to Skein. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 241–258. Springer, Heidelberg (2013)
15. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
16. Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: an efficient MAC algorithm for 32-bit microcontrollers. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 306–323. Springer, Heidelberg (2014)
17. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012)
18. Needham, R.M., Wheeler, D.J.: TEA extensions. Technical report (1997)
19. Nyberg, K.: Linear approximation of block ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
20. Nyberg, K., Wallén, J.: Improved linear distinguishers for SNOW 2.0. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 144–162. Springer, Heidelberg (2006)
21. Schulte-Geers, E.: On CCZ-equivalence of addition mod 2^n. Des. Codes Crypt. **66**(1–3), 111–127 (2013)
22. Sinz, C.: Towards an optimal CNF encoding of Boolean cardinality constraints. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 827–831. Springer, Heidelberg (2005)
23. Soos, M.: A blog about SAT solving and cryptography. http://www.msoos.org
24. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 244–257. Springer, Heidelberg (2009)
25. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014)
26. Wallén, J.: Linear approximations of addition modulo 2^n. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 261–273. Springer, Heidelberg (2003)
27. Wheeler, D.J., Needham, R.M.: TEA, a tiny encryption algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Heidelberg (1995)
28. Yao, Y., Zhang, B., Wu, W.: Automatic search for linear trails of the SPECK family. In: López, J., Mitchell, C.J. (eds.) ISC 2015. LNCS, vol. 9290, pp. 158–176. Springer, Heidelberg (2015)