IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2017, No. 1, pp. 281–306. DOI:10.13154/tosc.v2017.i1.281-306

Analysis of AES, SKINNY, and Others with Constraint Programming

Siwei Sun[1,4,5], David Gerault[2], Pascal Lafourcade[2], Qianqian Yang[1,4], Yosuke Todo[3], Kexin Qiao[1,4] and Lei Hu[1,4,5]*

[1] State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
sunsiwei@iie.ac.cn,yangqianqian@iie.ac.cn,qiaokexin@iie.ac.cn,hulei@iie.ac.cn
[2] Laboratory of Informatics, Modelling and Optimization of the Systems (LIMOS), University of Clermont Auvergne, Clermont-Ferrand, France
david@gerault.net,pascal.lafourcade@uca.fr
[3] NTT Secure Platform Laboratories, Tokyo, Japan
todo.yosuke@lab.ntt.co.jp
[4] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China
[5] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Search for different types of distinguishers are common tasks in symmetric-key cryptanalysis. In this work, we employ the constraint programming (CP) technique to tackle such problems. First, we show that a simple application of the CP approach proposed by Gerault *et al.* leads to the solution of the open problem of determining the exact lower bound of the number of active S-boxes for 6-round AES-128 in the related-key model. Subsequently, we show that the same approach can be applied in searching for integral distinguishers, impossible differentials, zero-correlation linear approximations, in both the single-key and related-(twea)key model. We implement the method using the open source constraint solver Choco and apply it to the block ciphers PRESENT, SKINNY, and HIGHT (ARX construction). As a result, we find 16 related-tweakey impossible differentials for 12-round SKINNY-64-128 based on which we construct an 18-round attack on SKINNY-64-128 (one target version for the crypto competition https://sites.google.com/site/skinnycipher announced at ASK 2016). Moreover, we show that in some cases, when equipped with proper strategies (ordering heuristic, restart and dynamic branching strategy), the CP approach can be very efficient. Therefore, we suggest that the constraint programming technique should become a convenient tool at hand of the symmetric-key cryptanalysts

Keywords: Differential Cryptanalysis, Integral Cryptanalysis, Constraint Programming, AES, SKINNY

1 Introduction

The design and analysis of symmetric-key cryptographic primitives is considered a tedious, time consuming, and error-prone task which involves tracing the propagation of bit-level patterns against all sorts of different operations according to some intricate rules. These bit patterns of interest for the cryptanalysts represent different meanings in different context. For example, in differential analysis [BS91] the bit patterns represent the differential

*Lei Hu is the corresponding author.

Received: 2016-11-23, Accepted: 2017-01-23, Published: 2017-03-08

characteristics, while in linear cryptanalysis [Mat94], the patterns correspond to the propagation of the linear masks.

In order to avoid extensive manual work and to deeply explore the exponential space of the bit patterns, we, as cryptographic researchers, are in urgent need of automatic tools. In fact, automatic tools for cryptanalysis designed by the community have played a significant role in the design and analysis of symmetric-key primitives.

Roughly speaking, those automatic tools can be divided into four categories, including search algorithms implemented from scratch in general purpose programming languages [Mat95, ANE15, BV14, BN11, FJP13, BDF11, DF16, DEM15, Leu13, YZW15, DDS14, SW16], SAT/SMT (satisfiability modulo theory) based methods [CB07, KY10, RS09, MP13, KLT15, QCW16, AJN14, SHY16], mixed-integer linear programming (MILP) based methods [AC11, MWGP12, WW11, SHW+14, FWG+16, XZBL16] and methods based on classical constraint programming.

To the best of our knowledge, the first application of the classical constraint programming (CP) technique in the field of block cipher cryptanalysis are presented in [GMS16]. In this work, Gerault *et al.* use a CP solver called Choco to search for the related-key differential characteristics of the AES, where some previous results [BN10, FJP13] are rediscovered in a highly automatic way and some better characteristics are found. Recently in [GL16], the authors used CP to perform a related-key cryptanalysis of a symmetric encryption scheme called Midori [BBI+15].

Each method presented above has its own advantages and drawbacks. For example, the method proposed in [BVC16] is able to give provable security bounds of an ARX cipher against simple differential attack, while the MILP/SMT based methods [MP13, FWG+16] can analyze more rounds of a cipher when compared to the method presented in [BVC16]. Moreover, sometimes methods implemented from scratch may be more efficient in some specific cases, and such methods probably are the only choices in some sophisticated situations. When compared with the methods based on SAT, SMT, MILP and CP, they are much more difficult to implement.

Our Contribution. Based on Gerault *et al.*'s work [GMS16], we apply the CP approach to search for differential/linear characteristics, integral distinguishers, impossible differentials and zero-correlation linear approximations automatically. Some experiments are performed on AES, PRESENT, SKINNY and HIGHT. We determine the exact lower bound of the number of active S-boxes of 6-round AES-128 in the related-key model. In addition, we find 16 related-tweakey impossible differentials of 12-round SKINNY, based on which we can attack 18-round SKINNY-64-128 (one target version for the crypto competition announced at ASK 2016).

We argue that the CP approach enjoys certain advantages over other methods in some aspects. Firstly, compared with the methods implemented from scratch, the CP approach is much easier to implement and more efficient in some cases. Second, the solution of the CP model can be delegated to a wide range of open-source or commercially available solvers. These solvers include dedicated CP solvers, but also SAT, MILP or hybrid solvers. In particular, by using the MiniZinc [NSB+07] language, one can express CP model in a language that can be interpreted by a wide range of solvers. Therefore, we directly benefit from the advances in resolution techniques. Thirdly, the modeling process of CP is much more straightforward than that of the MILP based method. Since in the MILP method, we need to encode the allowed bit patterns as a set of linear inequalities, while in the CP approach, we can directly input the allowed bit patterns as tuples into the CP solver. To the best of our knowledge, the MILP approach is unable to search for actual differential characteristics of ciphers with 8×8 S-boxes, while the CP approach does not have this limitation.

Organization. In Section 2, we give a brief introduction to the constraint programming and the Choco CP solver with sample codes. Section 3 explains how to search for differential and linear characteristics with CP, which is applied to determine the exact lower bound of the number of active S-boxes of 6-round AES-128 in the related-key model. In Section 4, we use the example of the search for integral distinguishers of PRESENT and the zero-correlation linear approximations of HIGHT to present some common techniques to improve the efficiency of the search. An impossible related-tweakey differential attack on 18-round SKINNY-64-128 is then given in Section 5 by exploiting some 12-round related-tweakey impossible differentials found by CP. We conclude in Section 6 and give some further discussions.

2 Constraint Programming and the Choco CP Solver

Definition 1. CP is used to solve Constraint Satisfaction Problems (CSPs). A CSP is defined by a triple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ such that

- $\mathcal{X}$ is a finite set of variables;
- $\mathcal{D}$ is a function that maps every variable $x_i \in X$ to its domain $\mathcal{D}(x_i)$, that is, the finite set of values that may be assigned to x_i;
- $\mathcal{C}$ is a set of constraints, that is, relations between some variables which restrict the set of values that may be assigned simultaneously to these variables.

A solution of a CSP is an assignment of values to all the variables in $\mathcal{X} = \{x_0, \cdots, x_{n-1}\}$ such that all constraints $\mathcal{C} = \{c_0, \ldots, c_{m-1}\}$ are satisfied. A CSP is said to be inconsistent if the set of its solutions is empty.

A generic approach for solving a CP model is the depth-first search algorithm with backtracking. At each step of the search, variable assignment is performed followed by a process called *constraint propagation* in which some values which can not occur in any solution are removed. The order in which variables are assigned in the search, as well as the order for exploring the possible values for each variable, has a significant impact on the efficiency. Therefore, choosing a good ordering heuristic is a key issue for solving CP problems.

One generic variable ordering heuristic among many other strategies is the so-called *domain over weighted degree* [BHLS04]. This is a conflict-directed variable ordering heuristic exploiting both the previous and current state of the search, and we refer the reader to [BHLS04] for more technical information.

Here we give a simple example of a constraint programming model with 5 0-1 variables $\{x_0, x_1, x_2, x_3, x_4\}$ and 3 constraints $\{c_0, c_1, c_2\}$

$$\begin{cases} c_0 \ : \ x_0 + x_2 + x_3 + x_4 = 3 \\ c_1 \ : \ x_0 \neq x_1 \\ c_2 \ : \ (x_0, x_1, x_2) \in \{(0,0,0), (0,1,0), (0,1,1)\} \end{cases}$$

According to Definition 1, the above CP model $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ has the following properties

- $\mathcal{X} = \{x_0, x_1, x_2, x_3, x_4\}$ and $\mathcal{D}(x_i) = \{0, 1\}$ for $0 \leq i \leq 4$;
- $\mathcal{C} = \{c_0, c_1, c_2\}$, $\mathsf{vars}(c_0) = \{x_0, x_2, x_3, x_4\}$, $\mathsf{vars}(c_1) = \{x_0, x_1\}$, and $\mathsf{vars}(c_2) = \{x_0, x_1, x_2\}$;
- $(x_0, x_2, x_3, x_4) \in \mathsf{rel}(c_0) = \{(0,1,1,1), (1,0,1,1), (1,1,0,1), (1,1,1,0)\}$, $(x_0, x_1) \in \mathsf{rel}(c_1) = \{(0,1), (1,0)\}$, and $(x_0, x_1, x_2) \in \mathsf{rel}(c_2) = \{(0,0,0), (0,1,0), (0,1,1)\}$.
- A solution of the CP model is $x_0 = 0$, $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$.

Note that a constraint can be declared in *extension*, by defining the valid/invalid tuples, or in *intension*, by defining a relation between the variables. For instance, c_0 and c_1 are declared in *intension*, and c_2 is declared in *extension*. Note that the modelling choices influence the resolution time: in this example, defining c_2 in intention would probably be more efficient. Such models can be solved by CP solvers such as Choco [PFL14]. To illustrate its ease of use, we give a toy example together with its source code in Appendix A. Other solvers exist, and different solvers may have very different performances for a same problem.

3 Search for Differential/Linear Characteristics

In this section, we consider an r-round iterative block cipher $\mathcal{E}_K : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$ with round function $\mathcal{F} : \{0,1\}^n \to \{0,1\}^n$. Typically, the $\mathcal{F}$ function can be decomposed into operations acting on smaller sub-blocks of the input data. When the corresponding parts of the input differences go through these operations, the input differences are transformed to output differences according to the differential properties of the operations.

The differential property of an operation $f : \{0,1\}^u \to \{0,1\}^v$ can be completely characterized by its differential distribution table DDT_f, where $\mathrm{DDT}_f[\boldsymbol{\alpha}][\boldsymbol{\beta}]$ specifies the probability $\Pr(\boldsymbol{\alpha} \to \boldsymbol{\beta})$ of the differential $\boldsymbol{\alpha} \to \boldsymbol{\beta}$ for all $\boldsymbol{\alpha} \in \{0,1\}^u$ and $\boldsymbol{\beta} \in \{0,1\}^v$. If we denote the input and output differences of f by $\boldsymbol{\alpha} = (\boldsymbol{\alpha}[0], \cdots, \boldsymbol{\alpha}[u-1]) \in \{0,1\}^u$ and $\boldsymbol{\beta} = (\boldsymbol{\beta}[0], \cdots, \boldsymbol{\beta}[v-1]) \in \{0,1\}^v$ respectively, the constraint imposed by f on the differential $\boldsymbol{\alpha} \to \boldsymbol{\beta}$ can be described in the language of constraint programming by

$$(\boldsymbol{\alpha}[0], \cdots, \boldsymbol{\alpha}[u-1], \boldsymbol{\beta}[0], \cdots, \boldsymbol{\beta}[v-1], p_{\boldsymbol{\alpha}\to\boldsymbol{\beta}}) \in \mathcal{D}_f$$

where $p_{\boldsymbol{\alpha}\to\boldsymbol{\beta}}$ is typically a positive integer called probability variable and

$$\mathcal{D}_f = \{(\mathbf{x}, \mathbf{y}, \log_2(\Pr(\mathbf{x} \to \mathbf{y}))) : \Pr(\mathbf{x} \to \mathbf{y}) > 0, (\mathbf{x}, \mathbf{y}) \in \{0,1\}^u \times \{0,1\}^v\}.$$

By imposing constraints according to the above method for all operations involved in a cipher, we can construct a CP model whose set of solutions is exactly the set of all possible differential characteristics. Further, by setting the objective function to minimize the sum of all probability variables, we can search for the characteristic with the highest probability.

In the following, we use $\mathcal{M}^{\mathsf{DSK}}_{\mathcal{E}_r}(\mathcal{C})$ and $\mathcal{M}^{\mathsf{DRK}}_{\mathcal{E}_r}(\mathcal{C})$ to denote the CP models whose set of solutionss are exactly the set of all differential characteristics satisfying the additional constraints specified in $\mathcal{C}$ in single-key and related-key models, respectively. For example, $\mathcal{M}^{\mathsf{DSK}}_{\mathcal{E}_r}(\Delta_{in} = \alpha, \Delta_{out} = \beta)$ is a CP model whose set of solutions is the set of all single-key differential characteristics of r-round $\mathcal{E}$ with specified input and output differences. When $\mathcal{C} = \emptyset$, it means that there is no additional constraint. Similarly, let $\mathcal{M}^{\mathsf{LIN}}_{\mathcal{E}_r}(\mathcal{C})$ be the CP model whose set of solutions is the set of all valid linear characteristics satisfying the additional constraints specified in $\mathcal{C}$ of an r-round cipher $\mathcal{E}$.

Also note that in the above formulation, we introduce a variable for every bit. When the operations involved in a cipher are all word oriented (aligned), we can introduce a variable x for every c-bit word, such that $\mathsf{dom}(x) = \{0, 1, \cdots, 2^c - 1\}$.

3.1 Exact Lower Bound of the Number of Active S-boxes of 6-round AES-128 in the Related-key Model

In the last few decades, a lot of research has been conducted on analysis and design of block ciphers, and the community has strong confidence in building efficient and secure block ciphers against the classical single-key differential attack.

However, when the adversary is allowed to ask for encryption or decryption with related keys, the situation becomes more complicated. One of the purpose of the key schedule algorithm is to resist against such attacks. In some extreme cases, as in LED [GPPR11], the designers choose to use no key schedule at all, at the expense of a larger number of rounds which may suffer from efficiency issues. In contrast, the key schedule algorithms of some block ciphers, *e.g.* the AES [DR02], are rather *ad-hoc*, in the sense that the designers came up with a key schedule that is quite different from the internal permutation of the cipher, in a hope that no harmful interaction is created by the two components. This approach typically makes the security evaluation in the related-key model very difficult [BKN09]. For example, it is pointed out in [Pey] at ASK 2016 (http://www.nuee.nagoya-u.ac.jp/labs/tiwata/ask2016/) that the exact lower bounds of the number of active S-boxes of r-round AES-128 in the related-key model are still unknown for $r \geq 6$. In the following, we show that a simple reapplication of the method presented in [GMS16] leads to the solution of the 6-round case. Note that, instead of using the solver Choco as for the rest of the paper, we use the setting proposed by Gerault *et al.*, *i.e.* the MiniZinc model they provided [1], as well as the solver Chuffed [2], where MiniZinc [3] is a solver-independent open source language that can be used to express CP models readable by multiple solvers.

First, adapt the parameters of the CP model of Gerault *et al.* to build one whose feasible region is exactly the set of all truncated related-key differential characteristics of 6-round AES-128, and set the objective function to minimize the number of differentially active S-boxes. For each solution, we construct a CP model whose set of solutions is exactly the set of all related-key differential characteristics matching the truncated related-key differential characteristic. The results we give were computed on a regular desktop computer. We find 19 truncated related-key differential characteristics with 20 active S-boxes in 7 hours, but none of them can be instantiated with an actual differential characteristic. We then find 1542 ones with 21 active S-boxes in around 12 hours. Among these, only 20 of them can be instantiated with actual differential characteristics. From that, we can conclude that the minimum number of active S-boxes of 6-round AES-128 in the related-key model is 21. The related-key differential characteristic with maximal probability occurs with probability 2^{-131}, and is given in Table. 1 whose truncated characteristic is depicted in Fig. 1. A comparison between the results obtained by CP and the graph-based search algorithm [FJP13] is given in Table 2.

Table 1: The optimal 6-round related-key differential characteristic for AES-128. It has 21 active S-boxes, and occurs with probability 2^{-131}. The four words represent the four columns and are given in hexadecimal notation. We have the relation: $\delta X \oplus \delta K_0 = \delta X_0$.

Round	$\delta X_i = X_i \oplus X_i'$	$\delta K_i = K_i \oplus K_i'$	Pr(States)	Pr(Key)
init.	366d1b80 dc37dbdb 9bc08d5b 00000000			
$i=0$	00000000 71000000 00004d00 00000000	366d1b80 ad37dbdb 9bc0c05b 00000000	$2^{-6 \cdot 2}$	–
1	b6f60000 009a0000 009a0000 009a0000	366d1b80 9b5ac05b 009a0000 009a0000	$2^{-7 \cdot 2} \cdot 2^{-6 \cdot 3}$	2^{-6}
2	00000000 009a0000 00000000 009a0000	ed6d1b80 7637dbdb 76addbdb 7637dbdb	$2^{-6 \cdot 2}$	$2^{-6} \cdot 2^{-7 \cdot 3}$
3	00000000 009a0000 009a0000 00000000	76addbdb 009a0000 7637dbdb 00000000	$2^{-6 \cdot 2}$	–
4	00000000 009a0000 00000000 00000000	76addbdb 7637dbdb 00000000 00000000	2^{-6}	–
5	00000000 009a0000 009a0000 009a0000	76addbdb 009a0000 009a0000 009a0000	$2^{-6 \cdot 3}$	2^{-6}
End/6	db000000 db9a0000 db000000 ad37dbdb	adaddbdb ad37dbdb adaddbdb ad37dbdb	–	–

Note that a practical reason which often limits the usability of the MILP based method is that it is impractical to compute the convex hull of all valid differential patterns of an 8×8 S-box, while the CP approach does not have this limitation.

[1] http://gerault.net/misc.php
[2] https://github.com/geoffchu/chuffed
[3] http://minizinc.org

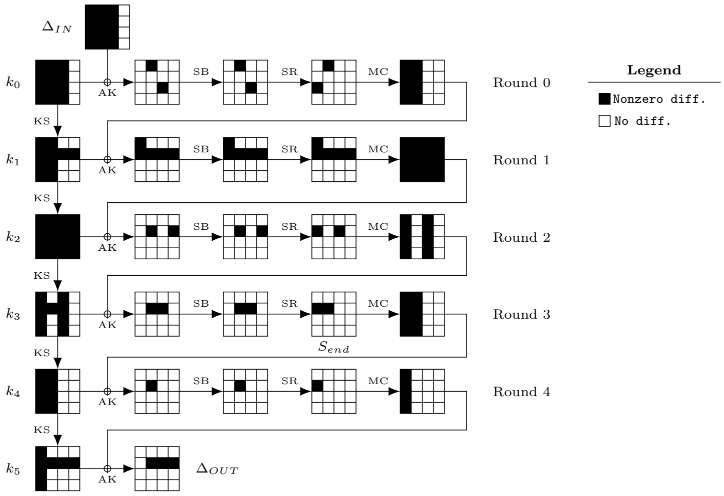

Figure 1: The optimal 6-round related-key differential characteristic for AES-128 in its truncated form.

Table 2: A comparison between the results obtained on AES-128 by constraint programming method and the graph-based search algorithm [FJP13], where #AS denotes the number of active S-boxes while Prob. is the probability of the best characteristic found. When no results are known, we simply write "-".

Rounds	Constraint Programming		Graph Search [FJP13]	
	#AS	Prob.	#AS	Prob.
3	5	2^{-31}	5	2^{-31}
4	12	2^{-79}	13	2^{-81}
5	17	2^{-105}	17	2^{-105}
6	21	2^{-131}	-	-

3.2 Comparing Solvers

In order to evaluate the performances of the Choco solver further, we picked two problems as benchmarks. The first one is an optimization problem, where the solver must find a differential characteristic with optimal probability. The second one is an enumeration problem, where the solver must list all solutions with predefined properties. It appears that the MILP solver Gurobi [Gur13] outperforms Choco on the optimization problem in our benchmark, and that Choco outperforms MILP for enumerating solutions. In order to try other solvers, we also implemented the first problem in MiniZinc. MiniZinc is a CSP modelling language that is accepted by a wide range of solvers. Using it, we could add the solvers Chuffed and PICAT_SAT [ZKF15] to the optimization experiments. It appears that Chuffed is in between Choco and Gurobi, and that PICAT_SAT outperforms Gurobi.

In the optitmization problem, we search for differential characteristics on PRESENT. It appears that Choco does not scale up well for a straightforward implementation of this search.We build a CP model $\mathcal{M}^{\mathsf{DSK}}_{\mathrm{PRESENT}_r}(\emptyset)$ for some r, set the objective function to minimize the sum of all probability variables, and try to find the optimal solution (corresponding to the best differential characteristic) by Choco. We also try to find the best differential characteristic of r-round PRESENT by the MILP based method using the Gurobi solver. The comparison of the results are listed in Table 3, from which we can see that our Choco implementation is not competitive with Gurobi on this problem, and that both approaches are extremely inefficient compared to Matsui's algorithm, which can find the best characteristic of full PRESENT in several seconds [ANE15]. It is noteworthy that

Table 3: Efficiency comparison of Choco, Gurobi, Chuffed and PICAT_SAT in searching for the best differential characteristic of PRESENT, with a time limit of 2 hours. A "-" means timeout.

Rounds	Prob.	Time by Gurobi (sec.)	Time by Choco (sec.)	Time by Chuffed (sec.)	Time by PICAT_SAT (sec.)
3	2^{-8}	2	4.1	0.2	12.8
4	2^{-12}	25	750.8	11.4	22.5
5	2^{-20}	453	-	3404.5	91.4
6	2^{-24}	2184	-	-	486.2
7	2^{-28}	-	-	-	5883.9

Table 4: Efficiency comparison of SCIP and Choco in enumerating characteristics in the linear hulls of PRESENT.

Rounds	Time by SCIP (sec.)	Number of solutions by SCIP	Time by Choco (sec.)	Number of solutions by Choco
4	0.1	3	0.023	3
5	0.28	17	0.031	17
6	37.7	8064	0.359	8064

Chuffed performs slightly better than the others on small instances, but that PICAT_SAT is the one which scales up the best. Note that out implementation in both frameworks is very straightforward. As opposed to Matsui's algorithm, it does not derive bounds from results on lower number of rounds, which would speed up the search. This does not either set a definitive advantage of one method over the other.

On the other hand, it seems that Choco is very good at enumerating the characteristics in a given differential or linear hull with fixed input and output differences. We construct a CP model $\mathcal{M}^{\mathsf{LIN}}_{\mathrm{PRESENT}_r}(\mathcal{C})$ for some r, where $\mathcal{C}$ dictates the input and output linear masks must be some fixed bit strings. Then we enumerate the set of solutions of the CP model by Choco. Also we try to enumerate the characteristics in the same linear hulls by SCIP [Tob04], which implements an efficient set of solutions enumeration algorithm based on MILP. The comparison of the two methods are given in Table 4, from which we can see that Choco dramatically outperforms SCIP in enumerating characteristics.

These results confirm how solver dependant the resolution process can be. It appears that there is not a definitive advantage of one method or solver for all purposes, and that different solvers perform differently on different problems. Hence, using the MiniZinc language seems to be the best practice, as it allows to try several solvers without having to translate the model to their respective language.

4 Accelerating the Search for Integral Distinguishers and Zero-correlation Linear Approximations

In this section, we apply the CP approach to search for integral distinguishers and zero-correlation linear approximations. Experimental results show that when combined with proper search strategies, the CP approach can be very efficient. Using it, we find again and more efficiently the currently known best integral distinguisher of PRESENT and zero-correlation linear approximations of HIGHT (ARX construction) [HSH+06]. First, we introduce a convenient tool from CP: random restarts.

Please take a look at the so-called *domain over weighted degree* heuristic specified in line 20 and 21 of the Choco code in Appendix A. This heuristic breaks ties at random, using the random seed provided as system time in the example. Ovbviously, the resolution performances from one execution to another, since the random seed changes. Occasionally, the variations of the resolution performances can be extraordinarily large, from one second to more than minutes from one run to another. This behavior was extensively studied, *e.g.* in [GSC97], and is common to many combinatorial problems. This is linked to the determining impact of the order in which the variables are treated on the resolution performance. A bad decision when using the randomized part to break a tie can thus dramatically increase the difficulty of finding a solution. To counter this, the method known as random restarts consists in starting over the search from scratch at a certain point if no solution was found. This results in different random choices, and possibly a way faster resolution. In practice, for one of the experiments described in the next paragraph, the search generally took less than one second per instance, except for some problematic runs where the solving time went over 10 minutes. By setting a random restart if the search took more than 1 second, we observed that no instance had to restart more than 10 times before reaching a solution in less than 1 second. Hence, these instances were solved in less than 10 seconds instead of more than 10 minutes.

4.1 Search for Integral Distinguishers

The division property, a generalized integral property, is proposed by Todo at Eurocrypt 2015 [Tod15b], which leads to the first theoretical attack on the full MISTY1 [Tod15a], and extended to bit-based division properties for analyzing bit-oriented ciphers [TM16]. In [XZBL16, SWW16], Xiang *et al.* and Sun *et al.* model the propagation of bit based division property as mixed-integer programming models, and automatically search for the integral distinguishers of a wide range of block ciphers. In the following, we show how to search for the integral distinguishers by employing the constraint programming technique.

Let $\mathbb{F}_2$ and $\mathbb{Z}$ denote the finite field of two elements and the integer ring, respectively. For vectors $\mathbf{k} = (k_0, k_1, \cdots, k_{n-1})$ and $\mathbf{u} = (u_0, u_1, \cdots, u_{n-1})$ in $\{0,1\}^n \subseteq \mathbb{Z}$, we say $\mathbf{u} \succcurlyeq \mathbf{k}$ if $u_i \geq k_i$ holds for all $i = 0, \cdots, n-1$.

Definition 2 (Conventional Bit-based Division Property [TM16]). Let $\mathbb{X}$ be a multiset whose elements belong to $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, where $\mathbb{K}$ denotes a set of n-dimensional vectors in $\{0,1\}^n \subseteq \mathbb{Z}^n$, it fulfills the following condition

$$\bigoplus_{\mathbf{x}\in\mathbb{X}} \pi_{\mathbf{u}}(\mathbf{x}) = \begin{cases} \text{unknown} & \text{if there are } \mathbf{k} \in \mathbb{K}, \text{s.t.} \mathbf{u} \succcurlyeq \mathbf{k} \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{u} = (u_0, u_1, \cdots, u_{n-1}) \in \{0,1\}^n \subseteq \mathbb{Z}^n$, $\mathbf{x} = (x_0, x_1, \cdots, x_{n-1}) \in \mathbb{F}_2^n$, and $\pi_{\mathbf{u}}(\mathbf{x}) = \prod_{i=0}^{n-1} x_i^{u_i}$.

If a multiset $\mathbb{X}$ has division property $\mathcal{D}_{\mathbb{K}}^{1^m}$, after the application of a vectorial boolean function $\mathbf{f} : \mathbb{F}_2^m \to \mathbb{F}_2^n$, the division property of the output multiset $\mathbb{Y}$ becomes $\mathcal{D}_{\mathbb{K}'}^{1^n}$. We say $\mathcal{D}_{\mathbb{K}}^{1^m}$ propagates to $\mathcal{D}_{\mathbb{K}'}^{1^n}$, which is denoted by $\mathcal{D}_{\mathbb{K}}^{1^m} \xrightarrow{\mathbf{f}} \mathcal{D}_{\mathbb{K}'}^{1^n}$, or $\mathbb{K} \xrightarrow{\mathbf{f}} \mathbb{K}'$.

In the following, we reformulate the propagation of the bit-based division property in the language of boolean functions. Our description is slightly different compared with [BC16, XZBL16, CJF+16], but they are essentially the same thing. Yet we think our description is easier for programming.

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be an n-variable boolean function which can be represented as the *Algebraic Normal Form* (ANF)

$$f(\mathbf{x}) = \bigoplus_{I\in\mathcal{P}_N} a_I \prod_{i\in I} x_i = \bigoplus_{I\in\mathcal{P}_N} a_I x_I$$

where $\mathcal{P}_N$ denotes the power set of $\{0, 1, \cdots, n-1\}$.

The set of all terms $\mathsf{Terms}(f)$ involved in a boolean function $f = \oplus_{I \in \mathcal{P}_N} a_I x_I$ is defined to be the set $\{x^I : a_I = 1\}$. We say a term of product of variables x^I is divisible by a term x^J, denoted by $x^J | x^I$, if $J \subseteq I$. A term x^J is covered by the ANF of a boolean function f if there exits $x^I \in \mathsf{Terms}(f)$, such that $x^J | x^I$, which is denoted by $x^J \lessdot f$. For example, $x_1x_2 \lessdot f = x_1x_2x_3 + x_2x_3 + 1$, while $x_2x_3x_4$ is not covered by $g = x_1x_2x_3 + x_1x_4 + x_3 + 1$.

Let $\mathbf{f} : \mathbb{F}_2^m \to \mathbb{F}_2^n$ be a vectorial boolean function whose coordinate function is denoted by $f_j(\mathbf{x})$, where $\mathbf{x} = (x_0, \cdots, x_{m-1})$. If the input set has division property $\mathcal{D}_{\mathbb{K}}^{1^m}$ where $\mathbb{K} = \{\mathbf{k} = (k_0, \cdots, k_{m-1})\}$ has only one element. The output division property $\mathcal{D}_{\mathbb{K}'}^{1^n}$ can be computed using the following algorithm called $\mathsf{propagate}()$ as $\mathbb{K}' = \mathsf{propagate}(\mathbb{K}, \mathbf{f})$ such that $\mathcal{D}_{\mathbb{K}_0}^{1^m} \xrightarrow{\mathbf{f}} \mathcal{D}_{\mathbb{K}'}^{1^n}$.

Algorithm 1: $\mathsf{propagate}()$ Compute the output division property.

Input: A vectorial boolean function $\mathbf{f} : \mathbb{F}_2^m \to \mathbb{F}_2^n$, and an input pattern $\mathbf{u} = (u_0, \cdots, u_{m-1}) \in \mathbb{F}_2^m$, where $f(\mathbf{x}) = (f_0(\mathbf{x}), \cdots, f_{n-1}(\mathbf{x}))$ and $\mathbf{x} = (x_0, \cdots, x_{m-1})$;

Output: $\mathcal{O}$: a set of patterns $\mathbf{v} \in \mathbb{F}_2^n$ describing the division property of the output set;

$\mathcal{O} = \emptyset$;
if $\mathbf{u} = (0, \cdots, 0)$ **then**
  return $\mathcal{O} = \{(0, \cdots, 0)\}$
else
  for $\mathbf{v} \in \mathbb{F}_2^n / (0, \cdots, 0)$ **do**
    Let $F = \prod_{j=0}^{n-1} f_j^{v_j}(x_0, \cdots, x_{n-1})$;
    if $\prod_{j=0}^{m-1} x_j^{u_j} \lessdot F$ **then**
      $\mathcal{O} = \mathcal{O} \cup \{v\}$;
    end
  end
end
return $\mathsf{reduced}(\mathcal{O})$;

The $\mathsf{reduced}()$ subroutine is used to remove all redundant vectors in a set such that there are no vectors $\mathbf{k}$ and $\mathbf{k}^*$ in $\mathbb{K}$ satisfying $\mathbf{k} \succcurlyeq \mathbf{k}^*$. If the input set has division property $\mathcal{D}_{\mathbb{K}}^{1^n}$ with $\mathbb{K} = \{\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_q\}$, after the application of a vectorial function $\mathbf{f}$, the division property of the output set $\mathcal{D}_{\mathbb{K}'}^{1^m}$ can be computed as follows

$$\mathbb{K}' = \mathsf{reduced}\Big(\bigcup_{i=1}^{q} \mathsf{propagate}(\{\mathbf{k}_i\}, \mathbf{f})\Big)$$

Example: Core operation of the SIMON family. The core operation of SIMON is a vectorial boolean function $f : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ with algebraic normal form

$$\begin{cases} y_0 = f_0(x_0, x_1, x_2, x_3) = x_0 \\ y_1 = f_1(x_0, x_1, x_2, x_3) = x_1 \\ y_2 = f_2(x_0, x_1, x_2, x_3) = x_2 \\ y_3 = f_3(x_0, x_1, x_2, x_3) = x_0x_1 + x_2 + x_3 \end{cases}$$

We show how to deduce the valid output patterns for the input division property $(1, 0, 1, 0)$.

Taking the output pattern $(0, 0, 0, 1)$ for example, since $F = f_0^0 f_1^0 f_2^0 f_3^1 = x_0x_1 + x_2 + x_3$, and $\mathsf{Terms}(F) = \{x_0x_1, x_2, x_3\}$. Therefore x_0x_2 is not covered by F and $(0, 0, 0, 1)$ is an invalid output pattern.

For output pattern $(0,0,1,1)$, since $F = f_0^0 f_1^0 f_2^1 f_3^1 = x_2(x_0x_1 + x_2 + x_3) = x_0x_1x_2 + x_2x_3 + x_2$, and $\mathsf{Terms}(F) = \{x_0x_1x_2, x_2x_3, x_2\}$. Therefore $x_0x_2 \lessdot F$ and $(0,0,1,1)$ is a valid pattern. Similarly, we can deduce that $(1,0,1,0)$ and $(1,0,0,1)$ are also valid output patterns. Note that $(0,1,1,1)$ is also a valid pattern according to Algorithm 1. But this pattern can be removed since $(0,1,1,1) \succcurlyeq (0,0,1,1)$.

Definition 3 (Division Trail [XZBL16])**.** Let $\mathcal{F}$ be the round function of an iterated block cipher. Assume that the input multi-set to the block cipher has initial division property $\mathcal{D}_{\mathbb{K}_0}^{1^n}$ with $\mathbb{K}_0 = \{\mathbf{k}\}$. This initial division property propagates through the round function which forms a chain

$$\mathcal{D}_{\mathbb{K}_0}^{1^n} \xrightarrow{\mathcal{F}} \mathcal{D}_{\mathbb{K}_1}^{1^n} \xrightarrow{\mathcal{F}} \mathcal{D}_{\mathbb{K}_2}^{1^n} \xrightarrow{\mathcal{F}} \cdots$$

For any vector $\mathbf{k}_i^* \in \mathbb{K}_i (i \geq 1)$, there must exist a vector $\mathbf{k}_{i-1}^*$ in $\mathbb{K}_{i-1}$ such that $\mathbf{k}_{i-1}^*$ can propagate to $\mathbf{k}_i^*$ according to the rules of division property propagation. Furthermore, for $(\mathbf{k}_0, \mathbf{k}_1, \cdots, \mathbf{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$, if $\mathbf{k}_{i-1}$ can propagate to $\mathbf{k}_i$ for all $i \in \{1, 2, \cdots, r\}$, we call $(\mathbf{k}_0, \mathbf{k}_1, \cdots, \mathbf{k}_r)$ an r-round division trail.

Similarly to the case of differential analysis, the propagation of the division property against a specific operation can also be described by allowing bit-level patterns. Taking the XOR operation for example, let $(x[0], x[1]) \in \{0,1\}^2$ and $x[2] \in \{0,1\}$ be the vectors describing the input and output division properties of the XOR operation respectively. Then we have the following constraint $(x[0], x[1], x[2]) \in \{(0,0,0), (0,1,1), (1,0,1)\} \subseteq \{0,1\}^3$. Therefore, by considering the constraints imposed on the propagation of division properties for all operations involved in a cipher, we can construct a CP model whose set of solutions is the set of all division trails for an r-round cipher $\mathcal{E}$.

Theorem 1 (Set without Integral Property [XZBL16])**.** *Let $\mathbb{X}$ be a multiset with division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, then $\mathbb{X}$ does not have integral property if and only if $\mathbb{K}$ contains all the n unit vectors.*

According to Theorem 1, whether there exists an integral distinguisher for an r-round iterative block cipher $\mathcal{E}$ with n-bit block size can be determined by Algorithm 2, where $\mathcal{M}_{\mathcal{E}_r}^{\mathsf{INT}}(\mathcal{C}_j)$ denotes the CP model whose set of solutions is the set of all division trails satisfying $\mathcal{C}_j$, which dictates that the output division property is the unit vector $\mathbf{e}_j$.

Algorithm 2: Search for integral distinguishers of r-round $\mathcal{E}$.

for $j \in \{0, 1, \cdots, n-1\}$ **do**
  $\mathcal{M} = \mathcal{M}_{\mathcal{E}_r}^{\mathsf{INT}}(\mathcal{C}_j)$
  if $\mathcal{M}$ is infeasible **then**
    An integral distinguisher is found.
  end
end

We implement the Algorithm 2 in the Choco solver combined with random restarts to search for the 9-round integral distinguisher of PRESENT, and the source code can be found in Appendix B. The 9-round distinguisher presented in [XZBL16] is rediscovered on an ordinary PC in no more than 36 seconds (the time of the resolution of 64 CP models). By contrast, the same search without using restarts was more than 10 times longer, and the MILP approach needs 3.4 minutes (roughly 204 seconds) [XZBL16] to solve the same problem. Note that only one thread is used in our experiment, but since each of the 64 models is independant, they could be solved in parallel.

4.2 Search for Impossible Differentials and Zero-Correlation Linear Approximations

Impossible differential cryptanalysis (IDC) [BBS99] is different from standard differential analysis in that IDC tries to recover the secret key by exploiting some differentials of the target cipher which never occur, instead of differentials with high probability. Similarly, zero-correlation linear cryptanalysis [BR14] uses linear approximations with zero correlation. The links between impossible differential, integral and zero-correlation linear approximation are explored in [C14]. Existing tools used to search for impossible differentials and zero-correlation linear approximations include $\mathcal{U}$-method [KHL10], UID-method [LLWG14], and the MILP based methods [WW12, CJF$^+$16, ST17].

Given an r-round cipher $\mathcal{E}_r$, it is trivial to see that a specific differential (linear approximation) $\alpha \to \beta$ is an impossible differential (zero-correlation linear approximation) if and only if the CP model $\mathcal{M}_{\mathcal{E}_r}^{\mathsf{DSK/LIN}}(\Delta_{in} = \alpha, \Delta_{out} = \beta)$ is infeasible, where $(\Delta_{in} = \alpha, \Delta_{out} = \beta)$ represents input-output difference patterns or input-output linear masks accordingly. Therefore, the problem of searching for impossible differential or zero-correlation linear approximation is equivalent to looking for the infeasible CP models in

$$\{\mathcal{M}_{\mathcal{E}_r}^{\mathsf{Property}}(\Delta_{in} = \alpha, \Delta_{out} = \beta) : \alpha, \beta \in \mathbb{F}_2^n - \{0\}\},\ \ \mathsf{Property} \in \{\mathsf{DSK}, \mathsf{DRK}, \mathsf{LIN}\}$$

However, the search space is too large to be enumerated by considering all possible α and β. Hence, typically the cryptanalysts only test those models whose input and output patterns with low Hamming weights. For example, a lot of work only search for those distinguishers whose Hamming weights of both the input and output bit patterns are 1, which can be accomplished by Algorithm 3.

Algorithm 3: Search for impossible differential or zero-correlation linear approximations.

for $i \in \{0, 1, \cdots, n-1\}$ **do**
  for $j \in \{0, 1, \cdots, n-1\}$ **do**
    $\mathcal{M} = \mathcal{M}_{\mathcal{E}_r}^{\mathsf{Property}}(\Delta_{in} = \mathbf{e}_i, \Delta_{out} = \mathbf{e}_j)$
    if $\mathcal{M}$ is infeasible **then**
      $\mathbf{e}_i \to \mathbf{e}_j$ is an impossible differential or zero-correlation linear approximation
    end
  end
end

We implement Algorithm 3 in Choco and applied it to HIGHT, which is an ISO standard lightweight block cipher introduced by Hong *et al.* at CHES 2006 [HSH$^+$06]. In [CJF$^+$16], Cui *et al.* tried to search for all 17-round zero-correlation linear approximations of HIGHT using the MILP method such that the Hamming weights of both the input and output linear masks are 1, and 4 zero-correlation linear approximations were found, which costs 4786 seconds on a server (Intel(R) Xeon(R) CPU E5-2620, 2.00GHz, 47GB RAM) using 12 threads. By using the CP approach with restarts, we rediscover this result on a PC using only one thread in 1709 seconds.

5 Related-tweakey Impossible Differential Attack on 18-round SKINNY-64-128

SKINNY is a new family of tweakable block ciphers presented at CRYPTO 2016 [BJK$^+$16] designed under the TWEAKEY framework [JNP14], whose goal is to compete with the

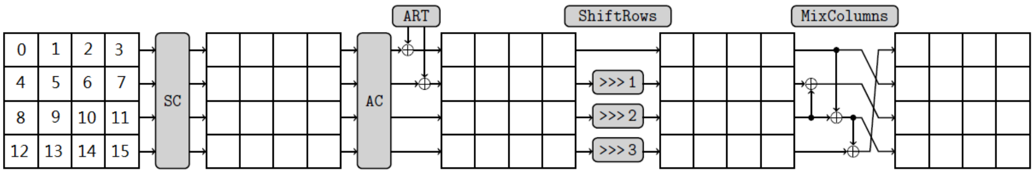

Figure 2: The SKINNY Round function: SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR) and MixColumns (MC).

NSA recent design SIMON in terms of hardware/software performances. Unlike SIMON, the designers of SKINNY provide strong bounds for all versions of the cipher, and not only in the single-key model, but also in the related-key or related-tweak model. At ASK 2016, the designers initiated a cryptanalysis competition to encourage third party analysis, and the 18-round SKINNY-64-128 is one target version (https://sites.google.com/site/skinnycipher/). In this section, we target this version with the aid of the CP. Some existing cryptanalysis of SKINNY which are better than the results presented in this paper can be found in [ABC+16, SMB16, TAY16, LGS16].

For the convenience of the discussion, we describe an attack on SKINNY-64-64, that is, the TK1 version with 64-bit block size and 64-bit secret key. We will see in the following that this attack can be directly converted to an attack on 18-round SKINNY-64-128 with 64-bit block size, 96-bit secret key, and 32-bit tweak. We refer the reader to [BJK+16] for the detailed description of the SKINNY cipher.

5.1 Notations

- $E_K^T(\cdot)$: The encryption oracle with key K and tweak T.
- K: The 64-bit master key of SKINNY-64-64.
- K_i: The ith round subkey ($1 \leq i \leq 18$). Hence, K_1 is the master key.
- $K_i[j]$: The jth nibble of K_i ($0 \leq j \leq 15$).
- $K_i[j_0, j_1, \cdots]$: $K_i[j_0]||K_i[j_1]|| \cdots$.
- $\Delta K_i, \Delta K_i[j], \Delta K_i[j_0, j_1, \cdots]$: The differences at the corresponding positions.
- I_i: The input internal state of round i ($1 \leq i \leq 18$).
- I_i^{SC}, I_i^{ART}, I_i^{SR}, I_i^{MC}: The internal state of round i after the SC, ART, SR, and MC operations, respectively.
- $\Longrightarrow$: logical implication. For example:

$$\{\Delta I_5 = 0, \Delta K_5 = \mathsf{0000000080000000}\} \Longrightarrow \Delta I_2[0, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 15] = 0$$

Note that in the above "0" represents the bit string of 64 0's and 0000000080000000 is in hexadecimal notation, which should be clear from the context. Under this notation, the input internal state of round i is I_i, which is transformed to I_i^{SC} after the application of the SC operation. I_i^{SC} is XORed with the subkey $K_i[0, \cdots, 7]$ to produce I_i^{ART}. The rows of I_i^{ART} are rotated (the SR operation) to get I_i^{SR} which subsequently becomes $I_i^{\mathrm{MC}} = I_{i+1}$ after the application of the MC operation. We refer the reader to Fig. 2 for more information.

5.2 Cryptanalysis

We implement Algorithm 3 in Choco with $\mathsf{Property} = \mathsf{DRK}$ and $\mathcal{E}_r = \text{SKINNY}_{12}$, and we search for related-tweakey impossible differentials of SKINNY-64-64 with the following input, output, and key differences. The Hamming weights of both $\Delta I_1||\Delta K_1$ and ΔI_{13} are all 1.

Since no differences are injected into the remaining 16 nibbles of the tweakey if we consider SKINNY-64-128, we are essentially analyzing the SKINNY-64-64, that is, the TK1 version. Therefore, in the figures demonstrating the analysis (see Fig. 3), we only draw 64-bit of the 128-bit tweakey state, and according to the tweakey schedule algorithm of SKINNY, this will not affect the differences of the subkeys.

Finally, we find 16 related-tweakey impossible differentials for 12-round SKINNY-64-64 (the results are summarized in Table 5), which is one more round than the impossible differentials presented in the SKINNY paper. With these related-tweakey impossible differentials, we can construct an attack on 18-round SKINNY-64-64 which directly leads to an attack on 18-round SKINNY-64-128 with 96-bit secret key and 32-bit tweak. The attack is depicted in Fig. 3.

Table 5: 16 related-tweakey impossible differentials for 12-round SKINNY-64-64 (In hexadecimal representation).

$\Delta I_1 \|\| \Delta K[0,\cdots,15]$	ΔI_{13}
0000000000000000||0000000080000000	0000000080000000
0000000000000000||0000000040000000	0000000040000000
0000000000000000||0000000020000000	0000000020000000
0000000000000000||0000000010000000	0000000010000000
0000000000000000||0000000000080000	0000000000080000
0000000000000000||0000000000040000	0000000000040000
0000000000000000||0000000000020000	0000000000020000
0000000000000000||0000000000010000	0000000000010000
0000000000000000||0000000000000800	0000000008000000
0000000000000000||0000000000000400	0000000004000000
0000000000000000||0000000000000200	0000000002000000
0000000000000000||0000000000000100	0000000001000000
0000000000000000||0000000000000008	0000000000800000
0000000000000000||0000000000000004	0000000000400000
0000000000000000||0000000000000002	0000000000200000
0000000000000000||0000000000000001	0000000000100000

Assuming $\Delta I_5 = 0$, $\Delta I_{17} = \mathsf{0000000080000000}$, and $\Delta K_5 = \mathsf{0000000080000000}$, we extend the 12-round related-tweakey impossible differential 4 rounds on the top and 2 rounds at the bottom, which is illustrated in Fig. 3. Note that

$$\Delta K_5 = \mathsf{0000000080000000} \implies \Delta K = \Delta K_1 = \mathsf{0000000000000008}.$$

Data Collection. Prepare 2^x structures $S_t = [P_0^{S_t}, P_1^{S_t}, \cdots, P_{2^{32}-1}^{S_t}]$ $(0 \le t \le 2^x - 1)$ each of which has 2^{32} plaintexts, and all plaintexts in the same structure share the same values in $I_1[1,2,3,4,9,11,12,13]$. For each plaintext $P_j^{S_t}$ we ask the encryption oracle to get $(C_j^{S_t}, \hat{C}_j^{S_t})$ where $C_j^{S_t} = E_K^{T_q}(P_j^{S_t})$ and $\hat{C}_j^{S_t} = E_{K\oplus\Delta}^{T_q}(P_j^{S_t})$, where T_q is an arbitrary tweak and

$$\Delta = \Delta K_1 = \mathsf{0000000000000008}$$

which requires totally $2 \times 2^x \times 2^{32}$ 18-round SKINNY encryptions. Then, for each structure, we can create approximately $2^{32} \times 2^{32} = 2^{64}$ pairs $[(P_i^{S_t}, P_j^{S_t}), (C_i^{S_t}, \hat{C}_j^{S_t})]$ such that

$P_i^{S_t} \oplus P_j^{S_t} = \Delta I_1$ with $\Delta I_1 = [0, 5, 6, 7, 8, 10, 14, 15] = 0$. If we choose 2^y different tweaks, we can get approximately $2^y \times 2^x \times 2^{64} = 2^{x+y+64}$ pairs (without increasing the number of chosen plaintexts) satisfying the desired condition with $2 \times 2^x \times 2^y \times 2^{32} = 2^{x+y+33}$ 18-round SKINNY encryptions. Note that this is equivalent to that we have 2^{x+y} structures denoted by $S_t = [P_0^{S_t}, P_1^{S_t}, \cdots, P_{2^{32}-1}^{S_t}]$ $(0 \le t \le 2^{x+y} - 1)$, and will use $[(P_i^{S_t}, P_j^{S_t}), (C_i^{S_t}, \hat{C}_j^{S_t})]$ to represent the pairs in the the t-th structure for the convenience of discussion.

Filtering. In this step, we will discard those pairs such that $\Delta I_5 \neq 0$ or $\Delta I_{17} \neq \mathsf{0000000080000000}$ under any key guess. Such pairs are helpless in discovering wrong key guesses.

Since $\Delta I_5 = 0$ and $\Delta K_5 = \mathsf{0000000080000000}$ implies $\Delta I_1^{\mathrm{SC}}[0] = \Delta I_1^{\mathrm{SC}}[7] = \Delta I_1^{\mathrm{SC}}[10]$ and $\Delta I_1^{\mathrm{SC}}[5] = \Delta I_1^{\mathrm{SC}}[8] = \Delta I_1^{\mathrm{SC}}[15]$, we discard those pairs that do not have this property, and there are $2^{x+y+64} \times 2^{-16} = 2^{x+y+48}$ pairs left. Similarly, $\Delta I_{17} = \mathsf{0000000080000000}$ and $\Delta K_5 = \mathsf{0000000080000000}$ implies $\Delta I_{18}^{\mathrm{ART}}[0, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15] = 0$ and $\Delta I_{18}^{\mathrm{ART}}[1] = \mathsf{8}$, it remains approximately $2^{x+y+48} \times 2^{-13\times 4} = 2^{x+y-4}$ pairs satisfying this property.

Key Recovery. We try to reduce the key space by spotting wrong key guesses. To deduce the values of ΔI_5 and ΔI_{17}, we need to know the values of the following 12 nibbles

$$K_1[0, 1, 2, 5, 6, 7], K_2[0, 1, 5], K_3[0], K_{18}[2, 6].$$

Step 1. For each guess of $K_1[0, 1, 2, 5, 6, 7] = k_{K_1} \in \{0, 1\}^{4\times 6}$, encrypt all the 2^{x+y-4} pairs, and create a set $\mathcal{X}_1$ (for each guess) contains all the pairs satisfying $\Delta I_2^{\mathrm{SC}}[1] = \mathsf{8}, \Delta I_2^{\mathrm{SC}}[4] = \Delta I_2^{SC}[11] = \Delta I_2^{SC}[14]$. The average size of one $\mathcal{X}_1$ set is approximately $2^{x+y-4} \times 2^{-12} = 2^{x+y-16}$. The time complexity of this step is $2^{x+y-4} \times 2^{4\times 6} = 2^{x+y+20}$ 1-round SKINNY encryptions.

Step 2. For each guess of $K_1[0, 1, 2, 5, 6, 7] = k_{K_1}\{0, 1\}^{4\times 6}$ and $K_{18}[2, 6] = k_{K_{18}} \in \{0, 1\}^{4\times 2}$, decrypt all the pairs in the corresponding $\mathcal{X}_1$ associated with k_{K_1}, and create a set $\mathcal{X}_2 \subseteq \mathcal{X}_1$ contains all the pairs satisfying $\Delta I_{18}[2] = \Delta I_{18}[10] = \Delta I_{18}[14]$ and $\Delta I_{17}[8] = \mathsf{8}$. The average size of one $\mathcal{X}_2$ set is approximately $2^{x+y-16} \times 2^{-12} = 2^{x+y-28}$. The time complexity of this step is $2^{x+y-16} \times 2^{4\times 6} \times 2^{4\times 2} = 2^{x+y+16}$ 2-round SKINNY encryptions.

Step 3. For each guess of $K_1[0, 1, 2, 5, 6, 7] = k_{K_1}\{0, 1\}^{4\times 6}$, $K_{18}[2, 6] = k_{K_{18}} \in \{0, 1\}^{4\times 2}$, and $K_2[0, 1, 5]||K_3[0] = k_{K_{2,3}} \in \{0, 1\}^{4\times 4}$, encrypt all the pairs in the corresponding $\mathcal{X}_2$ associated with $k_{K_1}||k_{K_{18}}$, and create a set $\mathcal{X}_3 \subseteq \mathcal{X}_2$ contains all the pairs satisfying $\Delta I_4^{SC}[0] = \mathsf{8}$. The average size of one $\mathcal{X}_3$ set is approximately $2^{x+y-28} \times 2^{-4} = 2^{x+y-32}$. The time complexity of this step is $2^{x+y-28} \times 2^{4\times 6} \times 2^{4\times 2} \times 2^{4\times 4} = 2^{x+y+20}$ 2-round SKINNY encryptions.

We can confirm a guess is a wrong key guess if and only if one of the 2^{x+y-4} pairs has the following property under the guess

$$\Delta I_5 = \mathsf{0000000000000000},\ \Delta I_{17} = \mathsf{0000000080000000}.$$

A key value is still in the key space if and only if no one of the 2^{x+y-4} pairs has the above property under the guessed key value. Hence, after the above steps, each of the $\mathcal{X}_4$s which is nonempty suggests a wrong key. The probability of a given pair surviving the filtering step satisfying $\Delta I_5 = \mathsf{0000000000000000}$ and $\Delta I_{17} = \mathsf{0000000080000000}$ under a random key guess is about

$$(1 - 2^{-28})^{2^{x+y-4}} = (1 - \frac{1}{2^{28}})^{2^{28+x+y-32}} \approx e^{-2^{x+y-32}}.$$

By adopting the strategy presented in [BNS14], we consider the number of pairs such that $(1 - 2^{-28})^{2^{x+y-4}}$ is slightly smaller than 2^{-1} so to reduce the exhaustive search by at

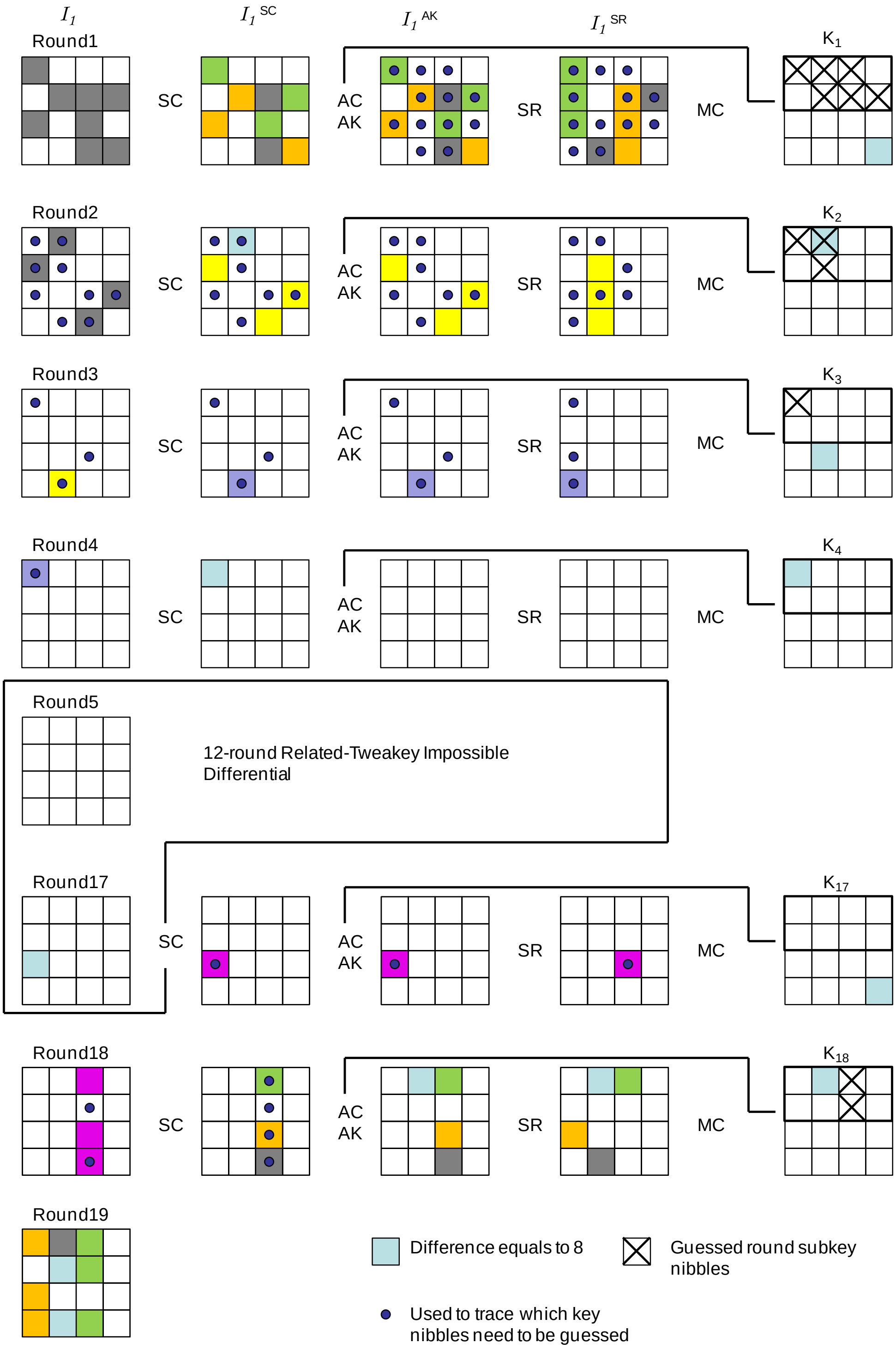

Figure 3: A relate-tweakey impossible differential attack on 18-round SKINNY-64-128.

least one bit. If we choose $x = 31$, $y = 2$, the remaining 12-nibble subkey space is reduced to $2^{4\times12} \times e^{-2} < 2^{4\times12-2} = 2^{46}$.

Complexity Analysis. Since $x = 31$ and $y = 2$, the number of chosen plaintexts is $2^{31} \times 2^{32} = 2^{63}$. The time complexity of the data collection step is $2^{31+2+33} = 2^{66}$. In step 1 through step 3, the time complexity is

$$\frac{1}{18} \times (2^{x+y+20} + 2^{x+y+16} \times 2 + 2^{x+y+20} \times 2) \approx 2^{51}.$$

If we chose to attack SKINNY-64-128 with 96-bit key and 32-bit tweak, from an information theoretical point of view, we reduce $12 \times 4 = 48$-bit key information to $48 - 2 = 46$ bits. Therefore, we still need to do an exhaustive search with complexity $2^{96-48} \times 2^{46} = 2^{94}$. In this attack, only one related-tweakey impossible differential is used. It is interesting to investigate how to improve the attack by exploiting multiple impossible differentials [BNS14].

6 Conclusion and Discussion

In this work, we apply the constraint programming method to search for integral distinguishers, impossible differentials, zero-correlation linear approximations and differential, linear characteristics in both single-key and related-key models. By using some searching strategies properly, we show the CP approach is faster than other method in some cases. Moreover, the CP approach has some appealing advantages. Firstly, it is highly automatic. Secondly, modeling under the CP framework is more straightforward than other methods. We can directly input the allowed tuples for some variables without converting them to linear inequalities or boolean formulas. Hence, there is no difficulty to model the cryptographic properties of an 8×8 S-box by using the CP approach. Therefore, we think the CP approach, together with the MILP, SMT, and SAT based techniques should become standard tools for symmetric-key crypanalysts. Also, we would like to propose some problems deserving further investigation:

- How to combine the technique of constraint programming and Matsui's algorithm to produce better method for finding the best differential/linear characteristics?

- Investigate how the ordering heuristic affects the resolution performance of the CP models derived form the problems of symmetric-key cryptanalysis.

- Solve the CP models derived from the problems of symmetric-key cryptanalysis by using other CP solvers rather than Choco to compare the performance.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. The figure of AES in the paper is produced by TkiZ [Jea16]. The work of this paper was supported by the National Key Basic Research Program of China (2013CB834203), the National Natural Science Foundation of China (61402469), and the State Key Laboratory of Information Security. The work of Siwei Sun is supported by the Youth Innovation Promotion Association of Chinese Academy of Sciences, and the Institute of Information Engineering (Qing-Nian-Zhi-Xing project). This research was conducted with the support of the FEDER program of 2014-2020, the region council of Auvergne, and the Digital Trust Chair of the University of Auvergne.

References

[ABC+16] Ralph Ankele, Subhadeep Banik, Avik Chakraborti, Eik List, Florian Mendel, Siang Meng Sim, and Gaoli Wang. Related-key impossible-differential attack on reduced-round skinny. Cryptology ePrint Archive, Report 2016/1127, 2016. http://eprint.iacr.org/2016/1127.

[AC11] Martin Albrecht and Carlos Cid. Cold boot key recovery by solving polynomial systems with noise. In *Applied Cryptography and Network Security – ACNS 2011*, pages 57–72. Springer, 2011.

[AJN14] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. Analysis of NORX: investigating differential and rotational properties. In *Progress in Cryptology - LATINCRYPT 2014 - Third International Conference on Cryptology and Information Security in Latin America, Florianópolis, Brazil, September 17-19, 2014, Revised Selected Papers*, pages 306–324, 2014.

[ANE15] Bannier Arnaud, Bodin Nicolas, and Filiol Eric. Automatic Search for a Maximum Probability Differential Characteristic in a Substitution-Permutation Network. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, pages 5165–5174. IEEE, 2015.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 411–436, 2015.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 12–23, 1999.

[BC16] Christina Boura and Anne Canteaut. Another view of the division property. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 654–682, 2016.

[BDF11] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In *CRYPTO 2011*, pages 169–187. Springer, 2011.

[BHLS04] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 146–150, 2004.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 123–153, 2016.

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 231–249, 2009.

[BN10] Alex Biryukov and Ivica Nikolić. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, Camellia, Khazad and others. In *Advances in Cryptology–EUROCRYPT 2010*, pages 322–344. Springer, 2010.

[BN11] Alex Biryukov and Ivica Nikolić. Search for related-key differential characteristics in DES-like ciphers. In *Fast Software Encryption – FSE 2011*, pages 18–34. Springer, 2011.

[BNS14] Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and Improving Impossible Differential Attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 179–199, 2014.

[BR14] Andrey Bogdanov and Vincent Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Des. Codes Cryptography*, 70(3):369–383, 2014.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[BV14] Alex Biryukov and Vesselin Velichkov. Automatic search for differential trails in ARX ciphers. In *Topics in Cryptology–CT-RSA 2014*, pages 227–250. Springer, 2014.

[BVC16] Alex Biryukov, Vesselin Velichkov, and Yann Le Corre. Automatic Search for the Best Trails in ARX: Application to Block Cipher SPECK. In *Fast Software Encryption – FSE 2016*. Springer, 2016.

[C14] Céline Blondeau and Kaisa Nyberg. Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In *Advances in Cryptology–EUROCRYPT 2014*, pages 165–182. Springer, 2014.

[CB07] Nicolas Courtois and Gregory V. Bard. Algebraic cryptanalysis of the data encryption standard. In Steven D. Galbraith, editor, *Cryptography and Coding, 11th IMA International Conference, Cirencester, UK, December 18-20, 2007, Proceedings*, volume 4887 of *Lecture Notes in Computer Science*, pages 152–169. Springer, 2007.

[CJF+16] Tingting Cui, Keting Jia, Kai Fu, Shiyao Chen, and Meiqin Wang. New automatic search tool for impossible differentials and zero-correlation linear approximations. Cryptology ePrint Archive, Report 2016/811, 2016. http://eprint.iacr.org/2016/689.

[DDS14] Itai Dinur, Orr Dunkelman, and Adi Shamir. Improved practical attacks on round-reduced keccak. *J. Cryptology*, 27(2):183–209, 2014.

[DEM15] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 490–509, 2015.

[DF16] Patrick Derbez and Pierre-Alain Fouque. Automatic Search of Meet-in-the-Middle and Impossible Differential Attacks. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 157–184, 2016.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[FJP13] Pierre-Alain Fouque, Jérémy Jean, and Thomas Peyrin. Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128. In *Advances in Cryptology–CRYPTO 2013*, pages 183–203. Springer, 2013.

[FWG+16] Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and Lei Hu. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 268–288, 2016.

[GL16] David Gérault and Pascal Lafourcade. Related-key cryptanalysis of midori. In *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, pages 287–304, 2016.

[GMS16] David Gerault, Marine Minier, and Christine Solnon. Constraint programming models for chosen key differential cryptanalysis. In *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, pages 584–601, 2016.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 326–341, 2011.

[GSC97] Carla P. Gomes, Bart Selman, and Nuno Crato. Heavy-tailed distributions in combinatorial search. In *Principles and Practice of Constraint Programming - CP97, Third International Conference, Linz, Austria, October 29 - November 1, 1997, Proceedings*, pages 121–135, 1997.

[Gur13] Gurobi Optimization. Gurobi optimizer reference manual. 2013. http://www.gurobi.com.

[HSH+06] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A new block cipher suitable for low-resource device. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pages 46–59, 2006.

[Jea16] Jérémy Jean. TikZ for Cryptographers. http://www.iacr.org/authors/tikz/, 2016.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 274–288, 2014.

[KHL10] Jongsung Kim, Seokhie Hong, and Jongin Lim. Impossible differential cryptanalysis using matrix method. *Discrete Mathematics*, 310(5):988–1002, 2010.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 161–185, 2015.

[KY10] Abdel Alim Kamal and Amr M. Youssef. Applications of SAT Solvers to AES Key Recovery from Decayed Key Schedule Images. In *Fourth International Conference on Emerging Security Information Systems and Technologies, SECURWARE 2010, Venice, Italy, July 18-25, 2010*, pages 216–220, 2010.

[Leu13] Gaëtan Leurent. Construction of differential characteristics in ARX designs application to SKEIN. In *Advances in Cryptology–CRYPTO 2013*, pages 241–258. Springer, 2013.

[LGS16] Guozhen Liu, Mohona Ghosh, and Ling Song. Security analysis of skinny under related-tweakey settings. Cryptology ePrint Archive, Report 2016/1108, 2016. http://eprint.iacr.org/2016/1108.

[LLWG14] Yiyuan Luo, Xuejia Lai, Zhongming Wu, and Guang Gong. A unified method for finding impossible differentials of block cipher structures. *Information Sciences*, 263:211–220, 2014.

[LSTV09] Christophe Lecoutre, Lakhdar Sais, Sébastien Tabary, and Vincent Vidal. Reasoning from last conflict(s) in constraint programming. *Artif. Intell.*, 173(18):1592–1614, 2009.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology–EUROCRYPT 1993*, pages 386–397. Springer, 1994.

[Mat95] Mitsuru Matsui. On correlation between the order of S-boxes and the strength of DES. In *Advances in Cryptology–EUROCRYPT 1994*, pages 366–375. Springer, 1995.

[MP13] Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for ARX: Application to Salsa20. IACR Cryptology ePrint Archive, Report 2013/328, 2013. http://eprint.iacr.org/2013/328.

[MWGP12] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology –ISC 2012*, pages 57–76. Springer, 2012.

[NSB+07] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard CP modelling language. In *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, pages 529–543, 2007.

[Pey] Thomas Peyrin. Presentation at ASK 2016: The Skinny Family of Tweakable Block Ciphers. http://www.nuee.nagoya-u.ac.jp/labs/tiwata/ask2016/slides/ask2016_10_Thomas.pdf.

[PFL14] Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco Solver Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2014. http://www.choco-solver.org.

[QCW16] Lingyue Qin, Huaifeng Chen, and Xiaoyun Wang. Linear hull attack on round-reduced simeck with dynamic key-guessing techniques. In *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*, pages 409–424, 2016.

[RS09] Mathieu Renauld and François-Xavier Standaert. Algebraic side-channel attacks. In *Information Security and Cryptology - 5th International Conference, Inscrypt 2009, Beijing, China, December 12-15, 2009. Revised Selected Papers*, pages 393–410, 2009.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 158–178, 2014.

[SHY16] Ling Song, Zhangjie Huang, and Qianqian Yang. Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*, pages 379–394, 2016.

[SMB16] Sadegh Sadeghi, Tahere Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round skinny block cipher. Cryptology ePrint Archive, Report 2016/1120, 2016. http://eprint.iacr.org/2016/1120.

[ST17] Yu Sasaki and Yosuke Todo. New Impossible Differential Search Tool from Design and Cryptanalysis Aspects. In *Advances in Cryptology - EUROCRYPT 2017*, 2017.

[SW16] Ling Sun and Meiqin Wang. Towards a Further Understanding of Bit-Based Division Propert. Cryptology ePrint Archive, Report 2016/392, 2016. http://eprint.iacr.org/2016/392.

[SWW16] Ling Sun, Wei Wang, and Meiqin Wang. MILP-Aided Bit-Based Division Property for Primitives with Non-Bit-Permutation Linear Layers. Cryptology ePrint Archive, Report 2016/811, 2016. http://eprint.iacr.org/2016/811.

[TAY16] Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. Impossible differential cryptanalysis of reduced-round skinny. Cryptology ePrint Archive, Report 2016/1115, 2016. http://eprint.iacr.org/2016/1115.

[TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 357–377, 2016.

[Tob04] Tobias Achterberg. *SCIP-a framework to integrate constraint and mixed integer programming*. Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2004.

[Tod15a] Yosuke Todo. Integral cryptanalysis on full MISTY1. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 413–432, 2015.

[Tod15b] Yosuke Todo. Structural evaluation by generalized integral property. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 287–314, 2015.

[WW11] Shengbao Wu and Mingsheng Wang. Security evaluation against differential cryptanalysis for block cipher structures. IACR Cryptology ePrint Archive, Report 2011/551, 2011. https://eprint.iacr.org/2011/551.

[WW12] Shengbao Wu and Mingsheng Wang. Automatic search of truncated impossible differentials for word-oriented block ciphers. In *Progress in Cryptology-INDOCRYPT 2012*, pages 283–302. Springer, 2012.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 648–678, 2016.

[YZW15] Yuan Yao, Bin Zhang, and Wenling Wu. Automatic search for linear trails of the SPECK family. In *Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings*, pages 158–176, 2015.

[ZKF15] Neng-Fa Zhou, Håkan Kjellerstrand, and Jonathan Fruhman. *Constraint Solving and Planning with Picat*. Springer Briefs in Intelligent Systems. Springer, 2015.

A The Choco CP Solver

Choco is an open source Java framework dedicated to constraint programming, which is among the fastest CP solvers on the market, and has been awarded two silver medals and three bronze medals at the MiniZinc challenge in 2013 and 2014 [PFL14]. In this section, we give a brief introduction of the relevant parts of Choco [4] by a simple example, and we refer the reader to the Choco documentation [PFL14] for more technical information.

Let $(\mathcal{X}, \mathcal{C})$ be a CP model with $\mathcal{X} = \{x_0, x_1, x_2\}$ and $\mathcal{C} = \{c_0\}$, where $\mathsf{dom}(x_0) = \mathsf{dom}(x_1) = \mathsf{dom}(x_2) = \{0, 1\}$, $\mathsf{vars}(c_0) = \{x_0, x_2\}$ and c_0 dictates that

$$(x_0, x_2) \in \{(1,1), (0,0), (0,1)\} \subseteq \mathsf{dom}(x_0) \times \mathsf{dom}(x_2) = \{0,1\}^2$$

The following code snippet gives all solutions of the CP model. The `solver` object returned by calling `new Solver()` in line 4 is a central object of the Choco framework and must be created first. In line 7 we create an array of three 0-1 variables `x[0]`, `x[1]` and `x[2]`. In line 10 to 17, we impose the constraint such that `(x[0], x[2])` can only take values from `(1, 1), (0, 0), (0, 1)`.

```
1  public class ToyExample {
2    public static void main(String[] args) {
3      // Create a Solver
4      Solver solver = new Solver();
5  
6      // Create variables through the variable factory
7      IntVar[] x = VF.enumeratedArray("x", 3, new int[]{0, 1}, solver);
8  
9      // Prepare the tuples representing the constraint
10     Tuples tuples = new Tuples(true);
11     tuples.add(1, 1);
12     tuples.add(0, 0);
13     tuples.add(0, 1);
14 
15     // Select variables and impose the constraint
16     IntVar[] vs = new IntVar[]{x[0], x[2]};
17     solver.post(ICF.table(vs, tuples, "AC2001"));
18 
19     // Specify a search heuristic
20     solver.set(ISF.domOverWDeg(vars, System.currentTimeMillis()));
21     solver.set(ISF.lastConflict(solver, solver.getStrategy()));
22 
23     // solve the model
24     if (solver.findSolution()){
25       do{
26         System.out.println(solver.toString());
27       }while(solver.nextSolution());
28     }
29   }
30 }
```

The third parameter (`"AC2001"`) of `ICF.table()` is used to specify an extensional constraint enforcing, most of the time, arc-consistency, and there are many other choices of this parameter [PFL14].

The so-called *domain over weighted degree* heuristic is specified in line 20. The parameter `System.currentTimeMillis()` of `ISF.domOverWDeg()` is used to seed the heuristic. `ISF.lastConflict(solver, solver.getStrategy())` is a composite dynamic branching heuristic which override the defined strategy by forcing some decisions to branch on variables involved in recent conflicts. After each conflict, the last assigned variable is

[4]In this paper, we work with Choco 3. Note that since the beginning of this work, version 4 was released, and is not backwards compatible

selected in priority, and we refer the reader to [LSTV09] for more technical information. Finally, we output all solutions of the CP model in line 24 to 27.

B Source code for finding 9-round Integral Distinguisher of PRESENT

```
import java.io.FileNotFoundException;
import org.chocosolver.solver.Solver;
import org.chocosolver.solver.constraints.ICF;
import org.chocosolver.solver.constraints.extension.Tuples;
import org.chocosolver.solver.search.limits.TimeCounter;
import org.chocosolver.solver.search.loop.monitors.SearchMonitorFactory;
import org.chocosolver.solver.search.strategy.IntStrategyFactory;
import org.chocosolver.solver.variables.IntVar;
import org.chocosolver.solver.variables.VariableFactory;

//find 9-round integral distinguisher of PRESENT
public class v2 {
   public static int R=9;
   public static int bl=64;


   public static void main(String[] args) throws FileNotFoundException {
      int[] values;
      long startTime = System.currentTimeMillis();
      for (int i=0;i<bl;i++) {
         values=new int[bl];
         for (int j=0;j<bl;j++) {
            if (j==i)
               values[j]=1;
            else
               values[j]=0;
         }
         System.out.println("i = "+ i);


         if (!testSolver(values, true)) {
            //No solution, checking without restarts
            if (!testSolver(values, false)) {
            System.out.println("No solution when the 1 is at position "+ i);
         }
         }
      }
      long endTime = System.currentTimeMillis();
      System.out.println("Running time: "+(endTime-startTime)+"ms");
   }
   public static boolean testSolver(int[] values, boolean restart) {
      IntVar[] vars= new IntVar[(R+1)*bl+R*16*8];
      int cpt=0;
      Tuples integral_path = new Tuples(true);
      integral_path.add(0, 0, 0, 0, 0, 0, 0, 0);
      integral_path.add(0, 0, 0, 1, 0, 0, 0, 1);
      integral_path.add(0, 0, 0, 1, 0, 0, 1, 0);
      integral_path.add(0, 0, 0, 1, 0, 1, 0, 0);
      integral_path.add(0, 0, 0, 1, 1, 0, 0, 0);
      integral_path.add(0, 0, 1, 0, 0, 0, 0, 1);
      integral_path.add(0, 0, 1, 0, 0, 0, 1, 0);
      integral_path.add(0, 0, 1, 0, 0, 1, 0, 0);
      integral_path.add(0, 0, 1, 0, 1, 0, 0, 0);
      integral_path.add(0, 0, 1, 1, 0, 0, 1, 0);
      integral_path.add(0, 0, 1, 1, 0, 1, 0, 0);
      integral_path.add(0, 0, 1, 1, 1, 0, 0, 0);
      integral_path.add(0, 1, 0, 0, 0, 0, 0, 1);
```

```
        integral_path.add(0, 1, 0, 0, 0, 0, 1, 0);
        integral_path.add(0, 1, 0, 0, 0, 1, 0, 0);
        integral_path.add(0, 1, 0, 0, 1, 0, 0, 0);
        integral_path.add(0, 1, 0, 1, 0, 0, 1, 0);
        integral_path.add(0, 1, 0, 1, 0, 1, 0, 0);
        integral_path.add(0, 1, 0, 1, 1, 0, 0, 0);
        integral_path.add(0, 1, 1, 0, 0, 0, 0, 1);
        integral_path.add(0, 1, 1, 0, 0, 0, 1, 0);
        integral_path.add(0, 1, 1, 0, 1, 0, 0, 0);
        integral_path.add(0, 1, 1, 1, 0, 0, 1, 0);
        integral_path.add(0, 1, 1, 1, 1, 0, 0, 0);
        integral_path.add(1, 0, 0, 0, 0, 0, 0, 1);
        integral_path.add(1, 0, 0, 0, 0, 0, 1, 0);
        integral_path.add(1, 0, 0, 0, 0, 1, 0, 0);
        integral_path.add(1, 0, 0, 0, 1, 0, 0, 0);
        integral_path.add(1, 0, 0, 1, 0, 0, 1, 0);
        integral_path.add(1, 0, 0, 1, 0, 1, 0, 0);
        integral_path.add(1, 0, 0, 1, 1, 0, 0, 0);
        integral_path.add(1, 0, 1, 0, 0, 0, 1, 0);
        integral_path.add(1, 0, 1, 0, 0, 1, 0, 0);
        integral_path.add(1, 0, 1, 0, 1, 0, 0, 0);
        integral_path.add(1, 0, 1, 1, 0, 0, 1, 0);
        integral_path.add(1, 0, 1, 1, 0, 1, 0, 0);
        integral_path.add(1, 0, 1, 1, 1, 0, 0, 0);
        integral_path.add(1, 1, 0, 0, 0, 0, 1, 0);
        integral_path.add(1, 1, 0, 0, 0, 1, 0, 0);
        integral_path.add(1, 1, 0, 0, 1, 0, 0, 0);
        integral_path.add(1, 1, 0, 1, 0, 0, 1, 0);
        integral_path.add(1, 1, 0, 1, 0, 1, 0, 0);
        integral_path.add(1, 1, 0, 1, 1, 0, 0, 0);
        integral_path.add(1, 1, 1, 0, 0, 1, 0, 1);
        integral_path.add(1, 1, 1, 0, 1, 0, 1, 1);
        integral_path.add(1, 1, 1, 0, 1, 1, 1, 0);
        integral_path.add(1, 1, 1, 1, 1, 1, 1, 1);

        int[] P = {0, 16, 32, 48, 1, 17, 33, 49, 2, 18, 34, 50, 3, 19, 35, 51,
               4, 20, 36, 52, 5, 21, 37, 53, 6, 22, 38, 54, 7, 23, 39, 55,
               8, 24, 40, 56, 9, 25, 41, 57, 10, 26, 42, 58, 11, 27, 43, 59,
               12, 28, 44, 60, 13, 29, 45, 61, 14, 30, 46, 62, 15, 31, 47, 63};

        Solver present = new Solver("present Integral2");

        IntVar[][] x = new IntVar[R+1][bl];

        for (int i = 0; i< R+1; i++)
        {
           for (int j=0;j<bl;j++) {
              x[i][j] = VariableFactory.bounded("x"+i+j,0 , 1, present);
              vars[cpt++]=x[i][j];
           }
        }

        for (int r = 0; r < R;r++)
        {
           for (int j = 0; j < 16;j++)
           {
              IntVar[] Svar = new IntVar[8];
              for (int i = 0; i < 4;i++)
              {
                 Svar[i] = x[r][j*4+i];
                 Svar[i+4] = x[r+1][P[j*4+i]];
                 vars[cpt++]=Svar[i];
                 vars[cpt++]=Svar[i+4];

              }

```

```java

          present.post(ICF.table(Svar, integral_path, "STR2+" /*"STR2+"*/));

        }

      }

      for (int i=0;i<bl;i++) {
        present.post(ICF.arithm(x[R][i],"=",values[i]));
      }
      IntVar varsSetTo1[]=new IntVar[60];
      IntVar varsSetTo0[]=new IntVar[4];

      for (int i=0;i<60;i++)
        varsSetTo1[i]=x[0][i];

      for (int i=0;i<4;i++)
        varsSetTo0[i]=x[0][i+60];

      present.post(ICF.sum(varsSetTo1,"=",VariableFactory.fixed(60,present)));
      present.post(ICF.sum(varsSetTo0,"=",VariableFactory.fixed(0,present)));

      present.set(IntStrategyFactory.domOverWDeg(vars, System.currentTimeMillis()));
      present.set(IntStrategyFactory.lastConflict(present,present.getStrategy()));
      // If restart is set, use geometrical restart 10 times:
      // first after 1 second(1000000000 ns), and then multiplying the time limit by 1.2
          at every iteration.
      if (restart)
        SearchMonitorFactory.geometrical(present, 1000000000, 1.2, new
             TimeCounter(present, 1), 10);
      boolean ret=present.findSolution();
      return ret;
    }
}
```