Analysis of AES, SKINNY, and Others with Constraint Programming

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

[1]Institute of Information Engineering, Chinese Academy of Sciences, China
[2]LIMOS, University Clermont Auvergne, France
[3]NTT Secure Platform Laboratories, Japan
[4]University of Chinese Academy of Sciences, China

FSE 2017 @ Tokyo

Outline

- Constraint programming (CP)
- Automatic cryptanalysis with CP
- Comparing solvers
- Conclusion and Discussion

Constraint Programming

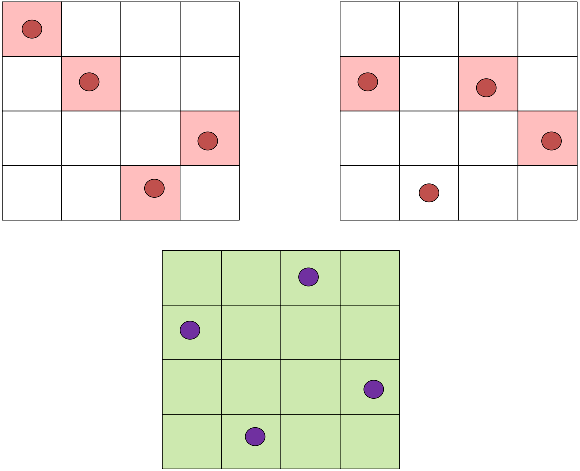
Definition : CP and CSP

CP is used to solve Constraint Satisfaction Problems (CSPs). A CSP is defined by a triple (X, D, C) such that

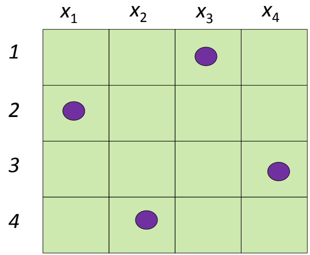
- $X = \{x_1, \cdots, x_n\}$ is a finite set of variables
- $D = \{D_1, \cdots, D_n\}$, where D_i is the domain of x_i, that is, the finite set of values that may be assigned to x_i. Hence $x_i \in D_i$.
- $C = \{C_1, \cdots, C_m\}$ is a set of constraints, where C_i defines a relation over $scope(C_i) \subseteq X$ which restrict the set of values that may be assigned simultaneously to these variables.

Constraint Programming – The *n* Queens Problem

Place n queens on an chessboard such that no queen can attack any other.



Formulating the n-Queens Problem



- Variables : $X = \{x_1, x_2, x_3, x_4\}$, x_i represents the row number of the queen at ith col
- Domains : $D = \{D_1, D_2, D_3, D_4\}$ where $D_i = \{1, 2, 3, 4\}$
- Constraints : $x_i \neq x_j$, $|x_i - x_{i+j}| \neq j$

Declare the constraints in extension

$(x_1, x_2) \in \{(1,3), (1,4), (2,4)(3,1), (4,1), (4,2)\}$
$(x_1, x_3) \in \{\cdots\}$

Constraint Programming : how to solve ?

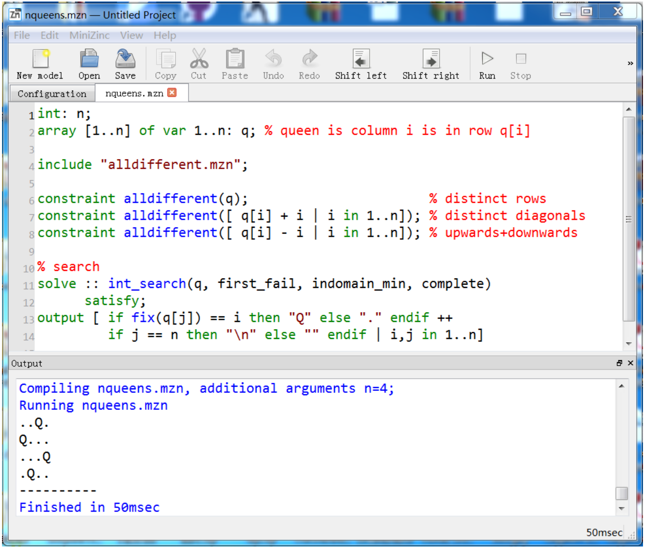
- Step 1. input the variables, domains, and constraints into a CP solver (Declare the problem)
- Step 2 : Wait for the solution

CP Solvers

- The CP solvers implement sophisticated backtracking and inference (constraint propagation) algorithms to find a solution.
- Solvers
  - Dedicated CP solvers : Choco, Chuffed, Gecode ...
  - SAT, MILP or hybrid solvers
  - Standard modelling language : Minizinc.

Eugene C. Freuder, April 1997

Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming : the user states the problem, the computer solves it.


nqueens.mzn — Untitled Project
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop
Configuration nqueens.mzn
int: n;
array [1..n] of var 1..n: q; % queen is column i is in row q[i]
include "alldifferent.mzn";
constraint alldifferent(q); % distinct rows
constraint alldifferent([q[i] + i | i in 1..n]); % distinct diagonals
constraint alldifferent([q[i] - i | i in 1..n]); % upwards+downwards
% search
solve :: int_search(q, first_fail, indomain_min, complete)
satisfy;
output [if fix(q[j]) == i then "Q" else "." endif ++
if j == n then "\n" else "" endif | i,j in 1..n]
Output
Compiling nqueens.mzn, additional arguments n=4;
Running nqueens.mzn
..Q.
Q...
...Q
.Q..

Finished in 50msec
50msec

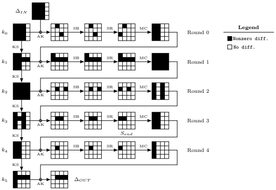
Automatic Cryptanalysis of Symmetric-key Algorithms

- Search algorithms implemented from scratch in general-purpose programming languages
- SAT/SMT based methods
- Mixed-integer programming (MILP) based methods
- Constraint programming (CP) based methods

Advantages of the CP approach

- Easy to implement
- Modelling process of CP is much more straightforward : input allowed tuples directly
- directly benefit from the advances in the resolution technique

Search for related-key differential characteristics of AES-128

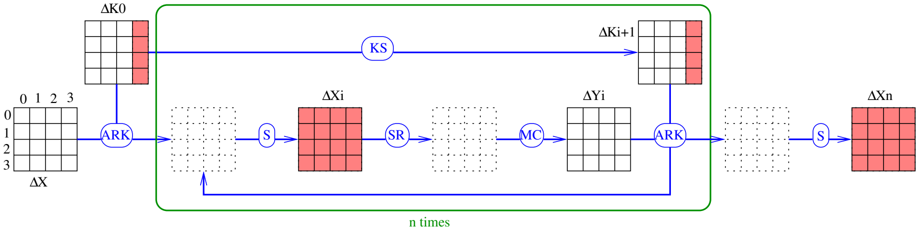


Related work

- [Alex Biryukov and Ivica Nikolić, EUROCRYPT 2010]
- [Pierre-Alain Fouque, Jérémy Jean and Thomas Peyrin, CRYPTO 2013]
- [David Gerault, Marine Minier and Christine Solnon, CP 2016]

- Step 1 : Find truncated differential characteristics with the minimum number of active S-boxes
- Step 2 : Instantiate the truncated differential characteristics with actual differences

CP Model for Step 1 : Variables and Constraints



- 0-1 variables
  - $\Delta X[j][k]$
  - $\Delta X_i[j][k]$
  - $\Delta Y_i[j][k]$
  - $\Delta K_i[j][k]$
- Constraints
  - ARK
  - SR-MC
  - KS
  - XOR

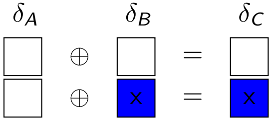
Semantics of the variables

These variables are used to trace the propagation of the truncated differences.

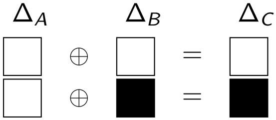
XOR Constraint

(white $= 0$, colored $\neq 0$)

Byte values



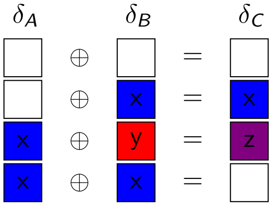
Boolean abstraction



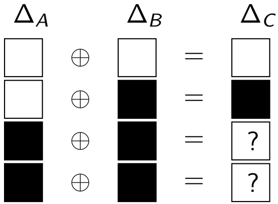
XOR Constraint

(white $=0$, colored $\neq 0$)

Byte values



Boolean abstraction

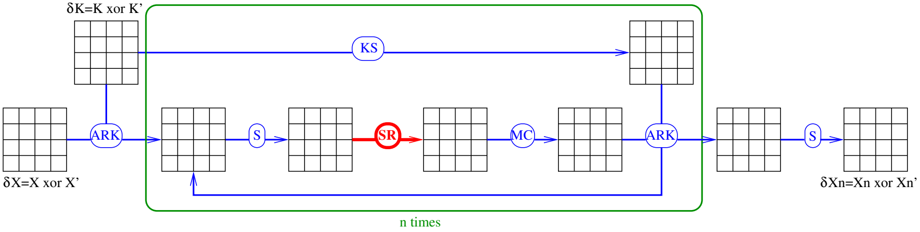


Δ_A	Δ_B	Δ_C
0	0	0
0	1	1
1	0	1
1	1	?

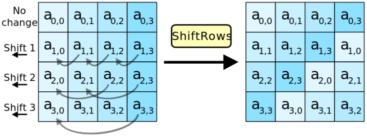
Definition of the XOR constraint

$\Delta_A + \Delta_B + \Delta_C \neq 1$

SR-MC Constraint



At byte level

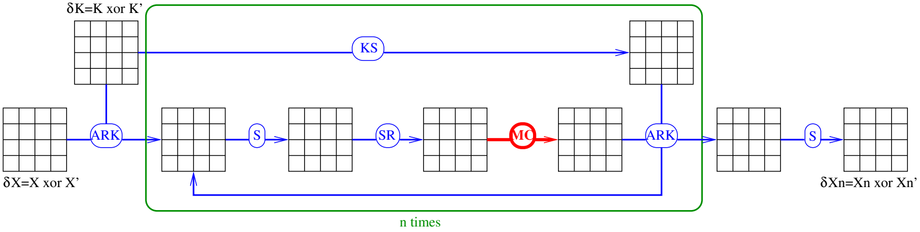


Definition of the SR-MC constraint

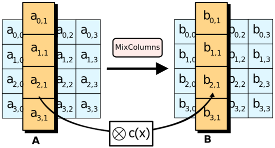
$\forall j \in [0;3]$:

$$\sum_{k=0}^{3} \Delta X_i[(k+j)\%4][k] + \Delta Y_i[j][k] \in \{0,5,6,7,8\}$$

SR-MC Constraint



At byte level



MDS property :
$|A| + |MC(A)| \in \{0, 5, 6, 7, 8\}$
(for diffusion of active cells)

Definition of the SR-MC constraint

$\forall j \in [0; 3]$:
$\sum_{k=0}^{3} \Delta X_i[(k + j)\%4][k] + \Delta Y_i[j][k] \in \{0, 5, 6, 7, 8\}$

CP Model for Step 1

- Impose constraints for all operations having an effect on the the truncated differences
- Impose additional constraints (at least one active byte)
- Set the objective function to minimize the number of active S-boxes

Problem

Too many inconsistent solutions!

CP Model for Step 1

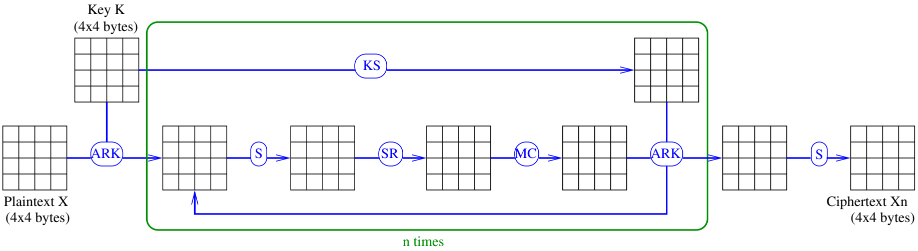
Reduce the number of inconsistent solutions

- Take the equality relationship into consideration : when $A == B$, $A \oplus B == 0$
- Consider the MDS property of two different columns

The Minizinc Code

`http://www.gerault.net/resources/CP_AES.tar.gz`

CP Model for Step 2



- Introduce a variable for every byte, whose domain is $\{0, 255\}$
- Impose the constraints of the differential distribution table, XOR etc. as table constraints
- Impose constraints according to the truncated differential characteristic

The Choco Code

`http://www.gerault.net/resources/Step2_AES.tar.gz`

Results for AES-128

- 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.
- The probability of the optimal characteristic is 2^{-131}.

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	$2^{-6.2}$	–
1	b6f60000 009a0000 009a0000 009a0000	366d1b80 9b5ac05b 009a0000 009a0000	$2^{-7.2} \cdot 2^{-6.3}$	2^{-6}
2	00000000 009a0000 00000000 009a0000	ed6d1b80 7637dbdb 76addbdb 7637dbdb	$2^{-6.2}$	$2^{-6} \cdot 2^{-7.3}$
3	00000000 009a0000 009a0000 00000000	76addbdb 009a0000 7637dbdb 00000000	$2^{-6.2}$	–
4	00000000 009a0000 00000000 00000000	76addbdb 7637dbdb 00000000 00000000	2^{-6}	–
5	00000000 009a0000 009a0000 009a0000	76addbdb 009a0000 009a0000 009a0000	$2^{-6.3}$	2^{-6}
End/6	db000000 db9a0000 db000000 ad37dbdb	adaddbdb ad37dbdb adaddbdb ad37dbdb	–	–

TABLE – The optimal characteristic

TABLE – A comparison between the results obtained by CP and the graph-based search algorithm [Pierre-Alain Fouque, Jérémy Jean and Thomas Peyrin, CRYPTO 2013].

Rounds	Constraint Programming		Graph Search	
	#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}	-	-

Search for Impossible differential and Zero-correlation Linear Approximation

Related work

- [Yu Sasaki and Yosuke Todo, EUROCRYPT 2017]
- [Cui, Jia, Fu, Chen and Wang, IACR ePrint 2016/689]

- Choose an input-output difference pattern (α, β).
- Construct a CP model $\mathcal{M}_{(\alpha,\beta)}$ whose solution set includes all valid differential characteristics.
- Solve $\mathcal{M}_{(\alpha,\beta)}$. If $\mathcal{M}_{(\alpha,\beta)}$ is infeasible, (α, β) is an impossible differential.
- Choose another (α, β) and repeat.

Search for Integral Distinguishers based on Bit-based Dvision Property

- Division property was proposed by Todo [Todo, EUROCRYPT 2015] which was extended to Bit-based division property [Todo and Morii, FSE 2016].

Bit-based division property

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

$$\bigoplus_{\mathbf{x}\in\mathbb{X}} x_0^{u_0} x_1^{u_1} \cdots x_{n-1}^{u_{n-1}} = \begin{cases} \text{unknown} & \text{if there are } \mathbf{k} \in \mathbb{K}, \text{s.t.} \mathbf{u} \succcurlyeq \mathbf{k} \\ 0 & \text{otherwise} \end{cases}$$

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

Using Division Property

- Construct an input set with division property $\mathcal{D}^{1^n}_{\mathbb{K}}$.
- Propagate it against the target cipher to get the output set with division property $\mathcal{D}^{1^n}_{\mathbb{K}'}$
- Extract some useful integral property from $\mathcal{D}^{1^n}_{\mathbb{K}'}$

The rule of propagation

The propagation of the division property can be described as a set of bit vectors, which in turn can be modeled by the language of CP.

Propagation of Division Property against Vectorial Boolean Functions

Algorithm 1: propagate() Compute the output division property.

Input: A vectorial boolean function $\mathbf{f} : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$, 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 reduced($\mathcal{O}$);

- [Xiang, Zhang, Bao and Lin, ASIACRYPT 2016]
- [Christina Boura and Anne Canteaut, CRYPTO 2016]
- [Ling Sun and Meiqin Wang, IACR ePrint 2016/392]

Example : the PRESENT S-box

Table: Division Trails of PRESENT Sbox

Input $\mathcal{D}_k^{1,4}$	Output $\mathcal{D}_{\mathbb{K}}^{1,4}$
(0,0,0,0)	(0,0,0,0)
(0,0,0,1)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,0,1,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,0,1,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,1,0,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,1,0,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,1,1,0)	(0,0,0,1) (0,0,1,0) (1,0,0,0)
(0,1,1,1)	(0,0,1,0) (1,0,0,0)
(1,0,0,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,0,0,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,0,1,0)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,0,1,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,1,0,0)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,1,0,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,1,1,0)	(0,1,0,1) (1,0,1,1) (1,1,1,0)
(1,1,1,1)	(1,1,1,1)

```
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);
```

Propagation of Division Property : Division Trail

- The bit-based division property can be described by the propagation of bit patterns with some special meaning, which leads to the concept of *division trail*.

Division Trail [Xiang, Zhang, Bao and Lin, ASIACRYPT 2016]

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 $\mathcal{D}^{1^n}_{\mathbb{K}_0}$ with $\mathbb{K}_0 = \{\mathbf{k}\}$. This initial division property propagates through the round function which forms a chain

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

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.

The rule for detecting integral distinguisher based on division property

Set without Integral Property

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.

- Construct a CP model $\mathcal{M}_{\mathbf{e}_j}$ whose solution set contains all the division trails whose output division property is set to $\mathbf{e}_j$.
- If we can find at least one $\mathcal{M}_{\mathbf{e}_j}$ for $j \in \{0, \cdots, n-1\}$ which is infeasible, then we find an integral distinguisher.

Accelerating the Search

- Ordering heuristic
  - The order in which the variables are assigned has significant impact on the efficiency of the resolution.
  - We choose the generic ordering heuristic called domain over weighted degree [Frédéric Boussemart et al., ECAI 2004]
- Random restart

Results on PRESENT, HIGHT, and SKINNY

- Retrieve the 9-round distinguisher of PRESENT found by MILP method(cost 3.4 minutes) in 36 seconds.
- Rediscover all zero-correlation linear approximations of the 17-round in 1709 seconds (MILP cost 4786).
- SKINNY : We found 16 impossible differentials leading to 18-round attack. Better results obtained by other researchers are now available for SKINNY [IACR ePrint 2016/1127, 1120, 1115, and 1108]

Note

During the process of designing new ciphers, the evaluation sometimes needs to be repeated several times. Hence, even though not crucial, a good CPU time is a desirable feature.

Comparing Solvers

- Pick two problems as benchmark
  - Optimization : find the best trail of PRESENT
  - Enumeration : list all solutions in a given linear hull of PRESENT
- Solvers
  - MILP solvers : Gurobi, SCIP
  - CP solvers : Choco, Chuffed, PICAT_SAT

Comparing Solvers

TABLE – Optimization problem, with a time limit of 2 hours.

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

Comparing Solvers

TABLE – Enumerating the linear hull 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

Conclusion and Discussion

- CP is indeed a convenient tool for symmetric-key cryptanalysis
  - Easy to implement
  - Sometimes faster
- Further directions
  - Most automatic tools focus on the search for distinguishers
  - Can we automate the key-recovery part ?
    [Patrick Derbez and Pierre-Alain Fouque, CRYPTO 2016]
    [Li Lin, Wenling Wu, Yafei Zheng, FSE 2016]

References

Mitsuru Matsui (1994)
On correlation between the Order of S-boxes and the Strength of DES
Advances in Cryptology–EUROCRYPT 1994

Alex Biryukov and Ivica Nikolić (2010)
Automatic search for related-key differential characteristics in byte-oriented block ciphers : Application to AES, Camellia, Khazad and others
Advances in Cryptology–EUROCRYPT 2010

Christoph Dobraunig and Maria Eichlseder and Florian Mendel (2015)
Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates
Advances in Cryptology–ASIACRYPT 2015

Patrick Derbez and Pierre-Alain Fouque
Automatic Search of Meet-in-the-Middle and Impossible Differential Attacks
Advances in Cryptology – CRYPTO 2016

Pierre-Alain Fouque and Jérémy Jean and Thomas Peyrin (2013)
Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128
Advances in Cryptology–CRYPTO 2013

Stefan Kölbl and Gregor Leander and Tyge Tiessen (2015)
Observations on the SIMON Block Cipher Family
Advances in Cryptology–CRYPTO 2015

David Gerault and Marine Minier and Christine Solnon (2016)
Constraint Programming Models for Chosen Key Differential Cryptanalysis
Principles and Practice of Constraint Programming–CP 2016

References

Yu Sasaki and Yosuke Todo (2017)
New Impossible Differential Search Tool from Design and Cryptanalysis Aspects
Advances in Cryptology–EUROCRYPT 2017

Tingting Cui and Keting Jia and Kai Fu and Shiyao Chen and Meiqin Wang (2016)
New Automatic Search Tool for Impossible Differentials and Zero-Correlation Linear Approximations
http://eprint.iacr.org/2016/689

Todo Yosuke (2015)
Structural Evaluation by Generalized Integral Property
Advances in Cryptology–EUROCRYPT 2015

Todo Yosuke (2015)
Integral Cryptanalysis on Full MISTY1
Annual Cryptology Conference–CRYPTO 2015

Yosuke Todo and Masakatu Morii (2016)
Bit-Based Division Property and Application to SIMON Family
Fast Software Encryption–FSE 2016

Christina Boura and Anne Canteaut (2016)
Another View of Division Property
Advances in Cryptology–CRYPTO 2016

Zejun Xiang and Wentao Zhang and Zhenzhen Bao and Dongdai Lin (2016)
Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers
Advances in Cryptology – ASIACRYPT 2016

References

Ling Sun and Meiqin Wang (2016)
Towards a Further Understanding of Bit-Based Division Propert
http://eprint.iacr.org/2016/392

Frédéric Boussemart and Fred Hemery and Christophe Lecoutre and Lakhdar Sais (ECAI 2004)
Boosting Systematic Search by Weighting Constraints
ECAI 2004

Li Lin and Wenling Wu and Yafei Zheng (2016)
Automatic Search for Key-Bridging Technique : Applications to LBlock and TWINE
FSE 2016

Thanks for your attention !