A Simpler Model for Recovering Superpoly on Trivium

Stéphanie Delaune, Patrick Derbez, **Arthur Gontier**, Charles Prud'homme

March 14, 2023

Context

Cube attacks: Dinur and Shamir at EUROCRYPT 2009

Variant of higher-order differencial attack

Let $f(x,v)$ a stream cipher: $(x \in \mathbb{F}_2^n, v \in \mathbb{F}_2^m)$

$$\sum_{C_I} f(x,v) = \sum_{C_I} \left(\textbf{p(x,v)} \prod_{i \in I} (v_i) + q(x,v) \right) = \textbf{p(x,v)}$$

main goal: Recover the superpoly **p**.

Context

Division property: Todo at EUROCRYPT 2015

Retrieve partial information of a polynomial

Exact Division property: Hao *et al.* at EUROCRYPT 2020

Retrieve complete superpoly

Monomial prediction: Hu *et al.* ASIACRYPT 2020

Retrieve complete superpoly

Section summary

Definitions

Definition (ANF, Algbraic Normal Form)

is a unique form to describe a boolean function $f \in \mathbb{F}_2$.

$$f(x_1, \dots, x_n) = \sum_{u \in \mathbb{F}_2^n} \prod_{i | u_i = 1} x_i = a_0 + \\ a_1 x_1 + \dots + a_n x_n + \\ a_{1,2} x_1 x_2 + \dots + a_{n-1,n} x_{n-1} x_n + \\ \dots + \\ a_{1,2,\dots,n} x_1 x_2 \dots x_n$$

The constants a fully define the function f and so do the set of chosen u.

Definition

Definition (Cube)

A cube is a set of chosen variables in the Initialisation Vector.

Definition (Superpoly)

A sub-part of the an ANF. Each monomial contain at least the cube.

Example

If the cube is $v_1v_2v_4$ and we find the monomials: $x_1v_1v_2v_4$, $x_1x_5v_1v_2v_4$ and $v_1v_2v_4$ then the superpoly is $x_1 + x_1x_5 + 1$

GOAL: GET THE SUPERPOLY

Retrieve superpoly: Monomial propagation

Example

$$(y_1, y_2) = f^{(1)}(x_1, x_2, x_3) = (x_1 + x_3, x_1x_2 + x_1),$$
$$(z_1, z_2) = f^{(2)}(y_1, y_2) = (y_1y_2, y_1 + y_2)$$

Retrieve superpoly: Monomial propagation

Example

$$(y_1, y_2) = f^{(1)}(x_1, x_2, x_3) = (x_1 + x_3, x_1x_2 + x_1),$$
$$(z_1, z_2) = f^{(2)}(y_1, y_2) = (y_1y_2, y_1 + y_2)$$

$$\begin{array}{lll} y_1 & = \underline{x_1} + x_3 & x_1 \rightarrow y_1 \\ y_2 & = x_1x_2 + \underline{x_1} & x_1 \rightarrow y_2 \\ y_1y_2 & = x_1x_2x_3 + x_1x_2 + x_1x_3 + \underline{x_1} & x_1 \rightarrow y_1y_2 \end{array}$$

Retrieve superpoly: Monomial propagation

Example

$$(y_1, y_2) = f^{(1)}(x_1, x_2, x_3) = (x_1 + x_3, x_1x_2 + x_1),$$
$$(z_1, z_2) = f^{(2)}(y_1, y_2) = (y_1y_2, y_1 + y_2)$$

$$\begin{aligned}
y_1 &= \underline{x_1} + x_3 & x_1 &\to y_1 \\
y_2 &= x_1x_2 + \underline{x_1} & x_1 &\to y_2 \\
y_1y_2 &= x_1x_2x_3 + x_1x_2 + x_1x_3 + \underline{x_1} & x_1 &\to y_1y_2
\end{aligned}$$

$$\begin{aligned}
z_1 &= \underline{y_1y_2} & x_1 &\to y_1y_2 \to z_1 \\
z_2 &= \underline{y_1} + \underline{y_2} & x_1 &\to y_1 \to z_2, x_1 \to y_2 \to z_2
\end{aligned}$$

GOAL: FIND ODD TRAILS

Differences with Division property

Division property: Bitwise constraints (copy, and, xor)

Example (Division Property model)

$$
\begin{aligned}
(x_{11}, x_{12}, x_{13}) &= \text{copy}(x_1) \\
y_1 &= \text{xor}(x_{11}, x_3) \\
a &= \text{and}(x_{12}, x_2) \\
y_2 &= \text{xor}(a, x_{13}) \\
(y_{11}, y_{12}) &= \text{copy}(y_1) \\
(y_{21}, y_{22}) &= \text{copy}(y_2) \\
z_1 &= \text{and}(y_{11}, y_{21}) \\
z_2 &= \text{xor}(y_{12}, y_{22})
\end{aligned}
$$

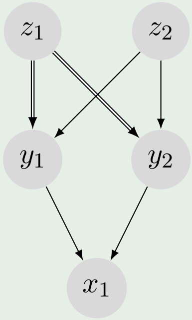
Differences with Division property

Division property: Bitwise constraints (copy, and, xor)

Example (Division Property model)

$$
\begin{aligned}
(x_{11}, x_{12}, x_{13}) &= \mathsf{copy}(x_1) \\
y_1 &= \mathsf{xor}(x_{11}, x_3) \\
a &= \mathsf{and}(x_{12}, x_2) \\
y_2 &= \mathsf{xor}(a, x_{13}) \\
(y_{11}, y_{12}) &= \mathsf{copy}(y_1) \\
(y_{21}, y_{22}) &= \mathsf{copy}(y_2) \\
z_1 &= \mathsf{and}(y_{11}, y_{21}) \\
z_2 &= \mathsf{xor}(y_{12}, y_{22})
\end{aligned}
$$

Example (Graph-based model)

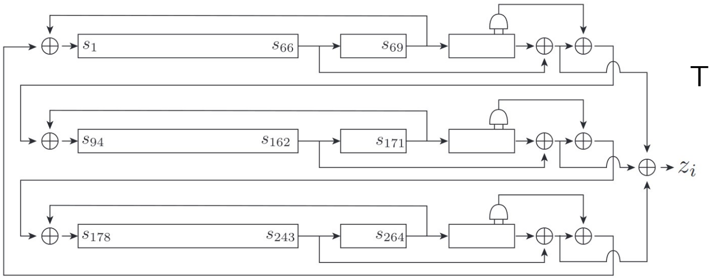


Contribution

- **Graph model**: simple and usable in MILP or CP
- **Doubling patterns**: reduce search space by cutting even trails
- **Arity approximation**: provide good solver strategy.
- **Better, Faster**: Half to 80% even trails cuts, 6 to 60 times faster.

Section summary

Trivium



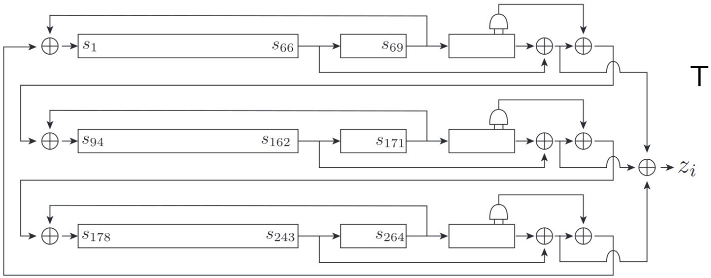
Three registers initialised with :

A = Key

B = Cube

C = Consts

Trivium



Three registers initialised with :

A = Key

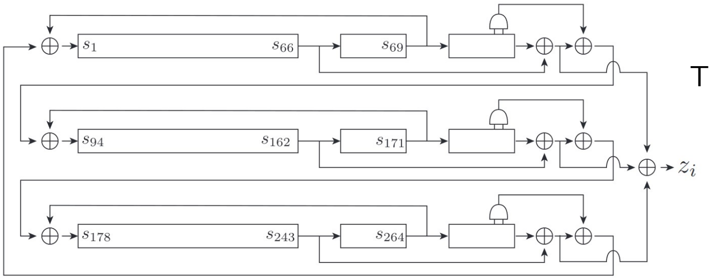
B = Cube

C = Consts

Register update :

$$\begin{array}{lcl} A & \leftarrow & A_{69} + C_{66} + C_{109}C_{110} + C_{111}, A_1, \ldots, A_{92} \\ B & \leftarrow & B_{78} + A_{66} + A_{91}A_{92} + A_{93}, B_1, \ldots, B_{83} \\ C & \leftarrow & C_{87} + B_{69} + B_{82}B_{83} + B_{84}, C_1, \ldots, C_{110} \end{array}$$

Trivium



Three registers initialised with :

A = Key

B = Cube

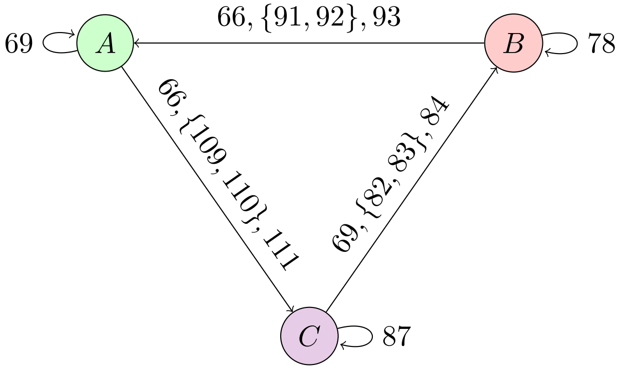
C = Consts

Register update :

$$\begin{array}{lcl} A & \leftarrow & A_{69} + C_{66} + C_{109}C_{110} + C_{111}, A_1, \ldots, A_{92} \\ B & \leftarrow & B_{78} + A_{66} + A_{91}A_{92} + A_{93}, B_1, \ldots, B_{83} \\ C & \leftarrow & C_{87} + B_{69} + B_{82}B_{83} + B_{84}, C_1, \ldots, C_{110} \end{array}$$

Output bit $z \leftarrow A_{66} + A_{93} + B_{69} + B_{84} + C_{66} + C_{111}$.

Trivium DFA



Idea: Develop the DFA from z

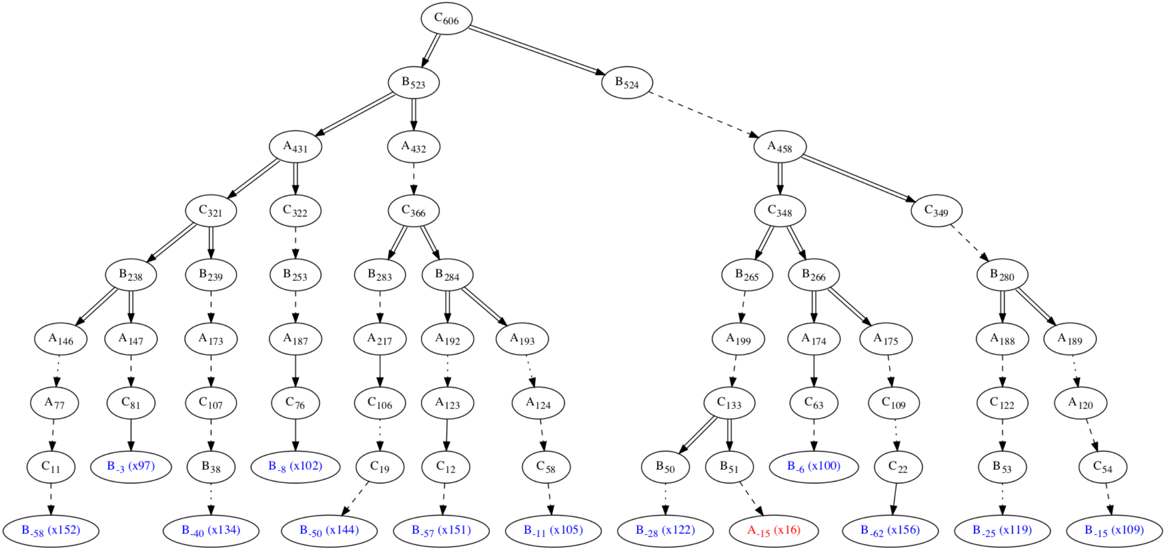
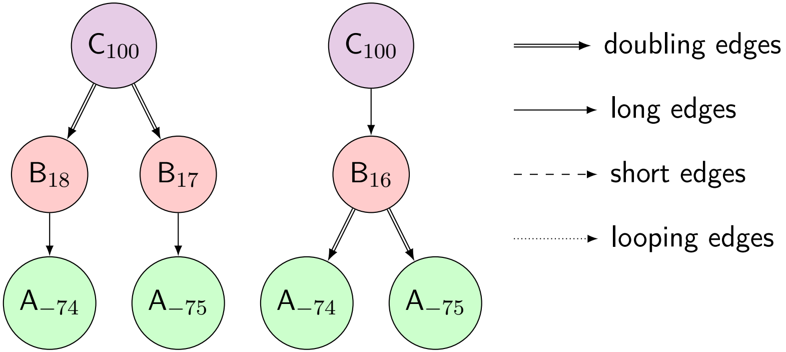


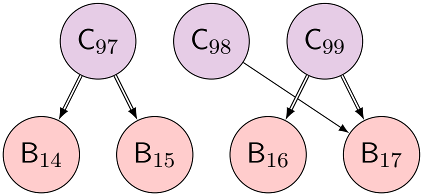
Figure: A solution for Trivium 672 considering s_{243} (66th bit of register C) as starter node. Blue nodes are the cube bits; the red one is the key bit.

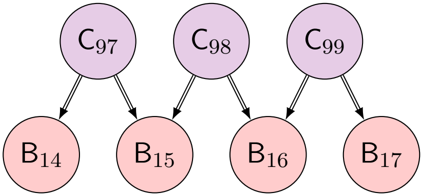
Section summary

Doubling patterns: Long-Double



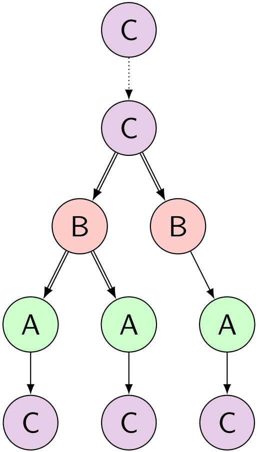
Doubling patterns: 3-Consecutive

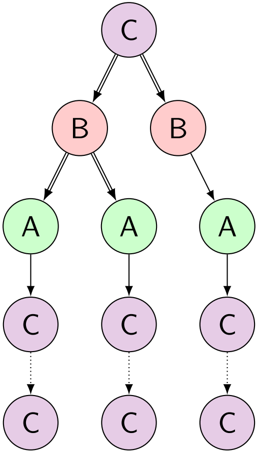




Hard patterns: Looping edge

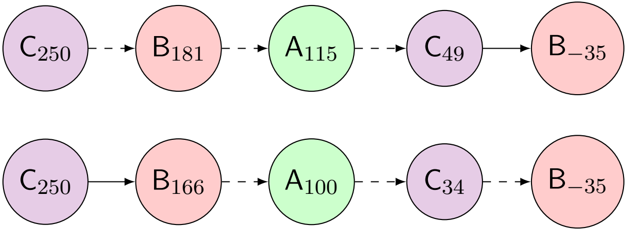
Looping edge is useful but too much constraints



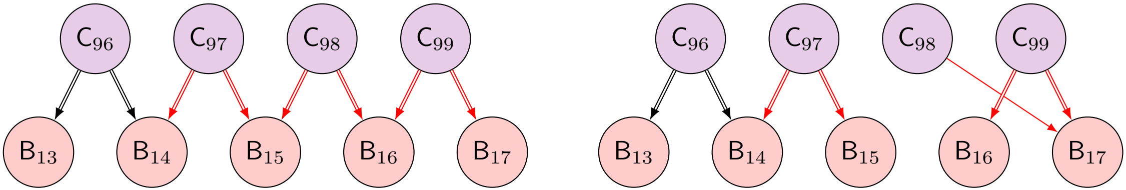


Useless patterns: Cycle

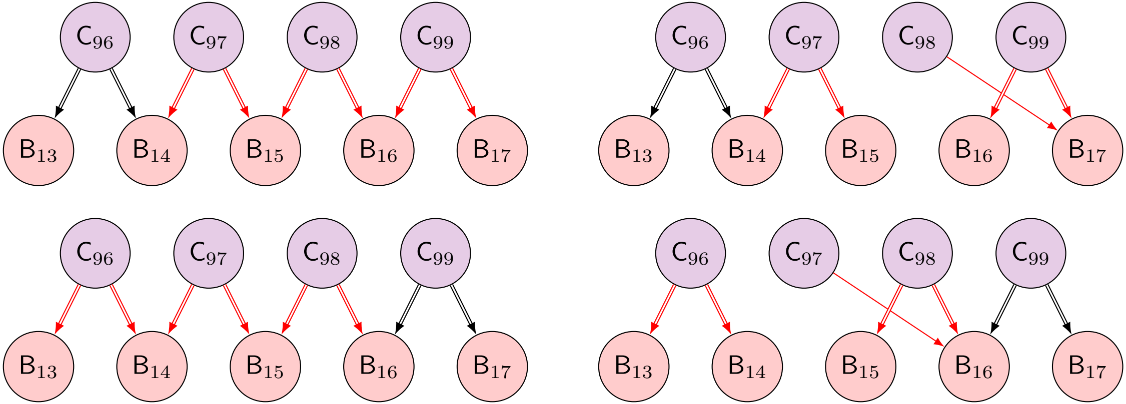
Cycle is easier to constrain but not oftenly used in our case



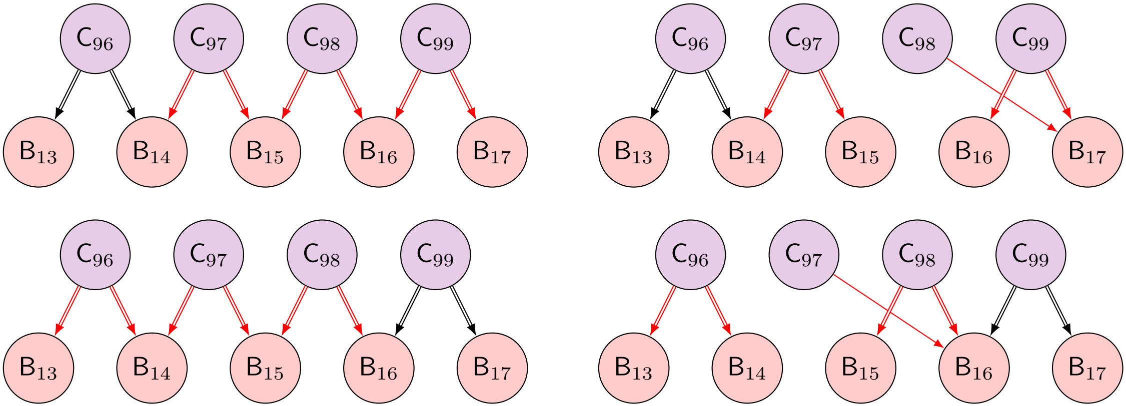
Overlapping patterns: example on 3-Consecutive



Overlapping patterns: example on 3-Consecutive



Overlapping patterns: example on 3-Consecutive



$$2X_{(\mathsf{C}_{96},\mathsf{B}_{13})} + 2X_{(\mathsf{C}_{97},\mathsf{B}_{14})} + X_{(\mathsf{C}_{97},\mathsf{B}_{16})} + 2X_{(\mathsf{C}_{98},\mathsf{B}_{15})} - X_{(\mathsf{C}_{99},\mathsf{B}_{16})} \leq 4$$

Results on Trivium

Graph solver	$R = 840/1$	$R = 841$	$R = 842$
without doubling constraints	12 909	30 177	3 188 835
with Pattern 3-consecutive	5 953	18 929	720 779

Table: Number of solutions

Section summary

Strategy

Division property: Hot restart

- Find one solution
- Restart from the beginning of the solution

Monomial Prediction: Divide-and-conquer

- Cut 200-400 rounds
- Solve all sub-solution

$\hookrightarrow$ **Basic MILP strategy is not sufficient**

$\hookrightarrow$ **How to use trivium structure?**

Arity approximation

Definition (Arity approximation: Liu at CRYPTO'17)

For each node approximate the number of reachable cube bits

Idea: develop the formula on two rounds to prevent most redundancies

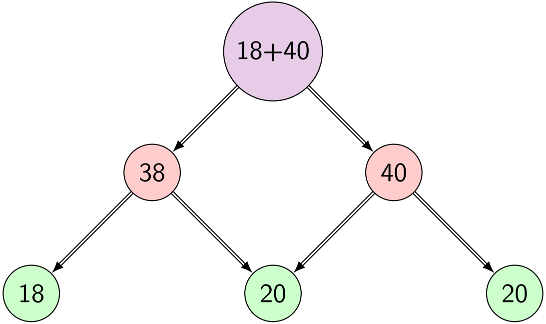


Figure: Arity redundancies on two doubling transitions

Strategy

1. **Compute arity approximation:** Dynamic programming

2. **Use solver strategy:** (ex: `Gurobi`)
   - `VarHintVal`: Focus on doubling edges
   - `BranchPriority`: Focus on big arity

     $\hookrightarrow$ no cut

Results on Trivium

Model	Monomial Prediction	Division Property	MILP Graph	CP Graph
$R = 675$	3m	1m	**3s**	15s
$R = 735$	4m	2m	**10s**	31m
$R = 840/1$	472m	269m	**10m**	$>$ 24h
$R = 840/2$	316m	91m	**10m**	
$R = 840/3$	351m	108m	**6m**	
$R = 841$	956m	282m	**19m**	
$R = 842$	$>$ 24h	990m	**182m**	

Table: Times on Trivium with 32 threads

Conclusion

A new **Graph model** that enables

- **Doubling patterns**
- **Arity approximation**
- faster MILP model: less even trails

Conclusion

A new **Graph model** that enables

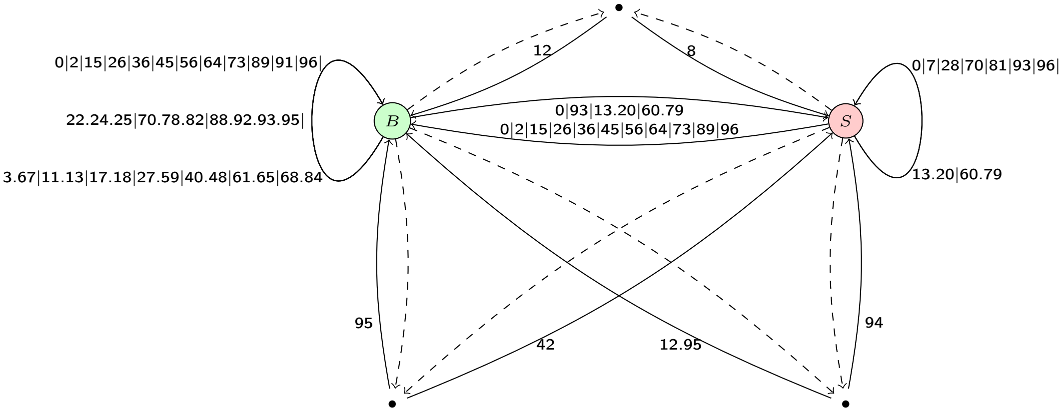
- **Doubling patterns**
- **Arity approximation**
- faster MILP model: less even trails

Open question

- Multiple pattern overlapping ?
- Strengthen the solver strategy (other trivium features ?)
- Apply the method on other ciphers

Thank you for listening

Grain DFA



0|2|15|26|36|45|56|64|73|89|91|96|
22.24.25|70.78.82|88.92.93.95|
3.67|11.13|17.18|27.59|40.48|61.65|68.84
12
8
B
S
0|93|13.20|60.79
0|2|15|26|36|45|56|64|73|89|96
0|7|28|70|81|93|96|
13.20|60.79
95
94
42
12.95