Massive Superpoly Recovery with a Meet-in-the-middle Framework
Improved Cube Attacks on Trivium and Kreyvium

Jiahui He[1], Kai Hu[1,2], Hao Lei[1], Meiqin Wang[1]

[1]Shandong University
[2]Nanyang Technological University

Eurocrypt 2024 @ Zurich
May 27, 2024

Outline

Cube Attack

- Proposed by Dinur and Shamir at EUROCRYPT 2009 [1]
- Any output bit of a symmetric cipher can be expressed as an Boolean function f of $\boldsymbol{k}$ and $\boldsymbol{v}$. The cube attack focuses on the coefficient of a monomial $t_I = \prod_{i \in I} v[i]$.
  - I is called cube indices and $v[i]$'s ($i \in I$) are called cube variables.
  - $f(\boldsymbol{k}, \boldsymbol{v}) = p(\boldsymbol{k}, \boldsymbol{v}) \cdot t_I + q(\boldsymbol{k}, \boldsymbol{v})$,
  - $p(\boldsymbol{k}, \boldsymbol{v})$ is called the superpoly of t_I and can be computed as $p(\boldsymbol{k}, \boldsymbol{v}) = \sum_{\boldsymbol{v} \in C_I} f(\boldsymbol{k}, \boldsymbol{v})$
- Cube attack
  - Offline phase: Fix non-cube variables to constants (usually $\mathbf{0}$) and recover the expression of the superpoly
  - Online phase: Calculate the value of the superpoly and establish an equation with respect to $\boldsymbol{k}$. The attacker can extract some information about $\boldsymbol{k}$ by solving the equation.

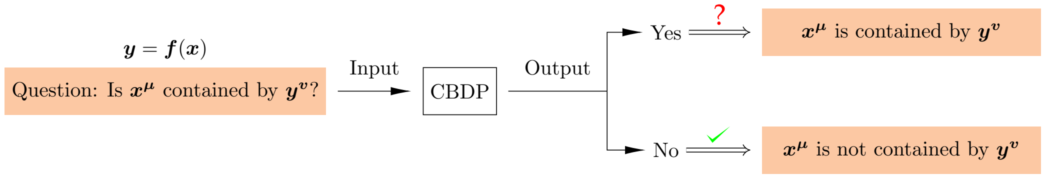
[1] I. Dinur, A. Shamir: Cube Attacks on Tweakable Black Box Polynomials. EUROCRYPT 2009

Previous Techniques for Superpoly Recovery

- Experimental methods
  - Only linear or quadratic superpolies are targeted
- Division Property
  - Todo et al. proposed the division property at EUROCRYPT 2015
  - Todo et al. introduced the division property into the cube attack at CRYPTO 2017
  - Wang et al. recovered the exact superpoly using three-subset division property at ASIACRYPT 2019
  - Hao et al. proposed the three-subset division property without the unknown subset (3SDPwoU) and utilized the Gurobi solver to recover the exact superpolies at EUROCRYPT 2020
- Methods based on monomial prediction
  - Hu et al. reinterpreted the 3SDPwoU from the perspective of monomial propagation at ASIACRYPT 2020
  - Hu et al. proposed a nested framework of monomial prediction to accelerate the recovery of superpoly at ASIACRYPT 2021
  - He et al. proposed the core monomial prediction technique at ASIACRYPT 2022 as a scaled-down version of monomial prediction

Conventional Bit-based Division Property (CBDP)

- CBDP was originally proposed as by Todo et al. at EUROCRYPT 2015 as a bit-level generalization of the integral property.
- At ASIACRYPT 2016, Xiang et al. introduced the concept of the division trail.
- At CRYPTO 2017, Todo et al. showed that CBDP can achieve perfect accuracy in claiming that a specific set of monomials of the input is not contained by the ANF of a output bit. However, CBDP can produce false positives.

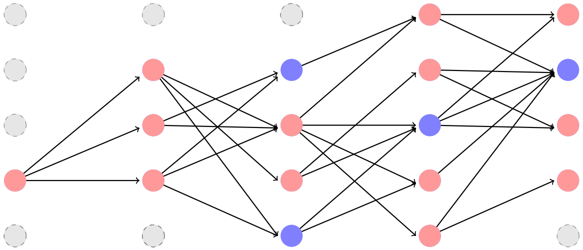


$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$
Question: Is $\boldsymbol{x}^{\boldsymbol{\mu}}$ contained by $\boldsymbol{y}^{\boldsymbol{v}}$?
Input
CBDP
Output
Yes
?
$\boldsymbol{x}^{\boldsymbol{\mu}}$ is contained by $\boldsymbol{y}^{\boldsymbol{v}}$
No
✓
$\boldsymbol{x}^{\boldsymbol{\mu}}$ is not contained by $\boldsymbol{y}^{\boldsymbol{v}}$

[1] $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{u[i]=1} x[i]$

Monomial Prediction (MP)

- For a composite Boolean function $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}^r \circ \cdots \circ \boldsymbol{f}^0(\boldsymbol{x})$, the monomial prediction/three-subset division property without unknown subset (3SDPwoU) technique can predict if $\boldsymbol{x}^{\boldsymbol{u}} \rightarrow \boldsymbol{y}^{\boldsymbol{v}}$, by counting the number of monomial/division trails.



- Let $f(\boldsymbol{k}, \boldsymbol{v}) = p(\boldsymbol{k}, \boldsymbol{v}) \cdot t_I + q(\boldsymbol{k}, \boldsymbol{v})$. To recover p after fixing non-cube variables to $\mathbf{0}$, use the monomial prediction technique to find all possible $g(\boldsymbol{k})$ that satisfy $g(\boldsymbol{k}) \cdot t_I \rightarrow f$, then $p = \sum g(\boldsymbol{k})$.

Initial Core Monomial Prediction (CMP)

Target cipher

1. Given an r-round cipher as
$$\boldsymbol{f}(\boldsymbol{k},\boldsymbol{v}) = \boldsymbol{f}^{r-1} \circ \boldsymbol{f}^{r-2} \circ \cdots \circ \boldsymbol{f}^{0},$$
where $\boldsymbol{f}^i$ is the round function at round i with $\boldsymbol{x}^i$ and $\boldsymbol{x}^{i+1}$ being the input and output, respectively.
2. The initial state $\boldsymbol{x}^0$ is loaded with $\boldsymbol{k}, \boldsymbol{v}$, constant 0 bits and constant 1 bits. The output bit z is the sum of monomials of $\boldsymbol{x}^r$.
3. Choose cube indices I and set the non-cube variables to constants, we want to compute the superpoly of t_I in z.

[1] Sometimes we use $\pi(\boldsymbol{x}^r, \boldsymbol{t}^r)$ to represent $\boldsymbol{x}^{r\boldsymbol{t}^r}$ for the sake of clarity

Initial Core Monomial Prediction (CMP)

Flag technique

For each bit b of a round state $\boldsymbol{x}^i$, we express it a polynomial g of cube variables and $\boldsymbol{k}$, and assign it an additional flag $b.F \in \{1_c, 0_c, \delta\}$.

- 0_c means g is constant 0.
- δ means g involves at least one cube variable.
- 1_c means g is non-zero and does not involve cube variables.

Example

Let $(k[0], k[1], v[0], v[1], v[2])$ be the input of $\boldsymbol{f}$ and cube indices $I = \{0, 1\}$. We set $v[2] = 0$.

- $g = k[0]k[1] + 1 \Rightarrow b.F = 1_c$
- $g = k[0]v[0] + k[1] \Rightarrow b.F = \delta$
- $g = 0 \Rightarrow b.F = 0_c$
- $g = k[1]v[2] + k[0] \Rightarrow b.F = 1_c$

Initial Core Monomial Prediction (CMP)

Operations of flags

We define $=$, $\oplus$ and $\times$ operations for the elements of $\{1_c, 0_c, \delta\}$, corresponding to the basic operations COPY, XOR and AND, respectively:

- The $=$ operations sets one element equal to another element.
- The $\oplus$ operations follows the rules:
  - $1_c \oplus 1_c = 1_c$
  - $0_c \oplus x = x \oplus 0_c = x$ for arbitrary $x \in \{1_c, 0_c, \delta\}$
  - $\delta \oplus x = x \oplus \delta = \delta$ for arbitrary $x \in \{1_c, 0_c, \delta\}$
- The $\times$ operation follows the rules:
  - $1_c \times x = x \times 1_c = x$ for arbitrary $x \in \{1_c, 0_c, \delta\}$
  - $0_c \times x = x \times 0_c = 0_c$ for arbitrary $x \in \{1_c, 0_c, \delta\}$
  - $\delta \times \delta = \delta$

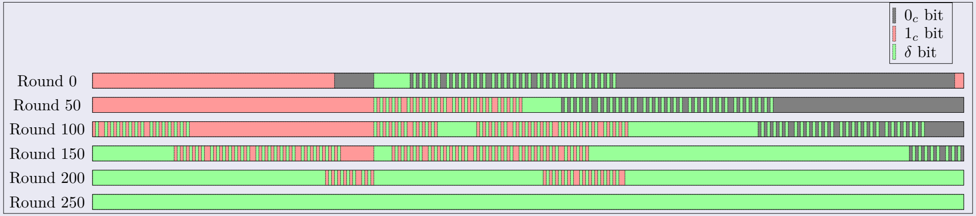
Remark: The flags can be computed round by round quickly without the help of the MILP solver.

Initial Core Monomial Prediction (CMP)

Example

This is the distribution of flags at different rounds of Trivium for the cube indices

$$I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19, 21, 24, 26, 28, 30, 32, 34, 36, 39, 41, 43, 45, 47, 49, 51, 54, 56, 58, 60, 62, 64, 66, 69, 71, 73, 75, 77, 79\}.$$



Initial Core Monomial Prediction (CMP)

Flag masks

For i-th round state $\boldsymbol{x}^i$, we use three binary vectors $\boldsymbol{M}^{i,\delta}, \boldsymbol{M}^{i,1_c}, \boldsymbol{M}^{i,0_c}$ to indicate the δ, 1_c, 0_c bits of $\boldsymbol{x}^i$, respectively.

$$\boldsymbol{M}^{i,\delta}[j] = 1 \text{ if and if only } x^i[j] \text{ is a } \delta \text{ bit.}$$

$$\boldsymbol{M}^{i,1_c}[j] = 1 \text{ if and if only } x^i[j] \text{ is a } 1_c \text{ bit.}$$

$$\boldsymbol{M}^{i,0_c}[j] = 1 \text{ if and if only } x^i[j] \text{ is a } 0_c \text{ bit.}$$

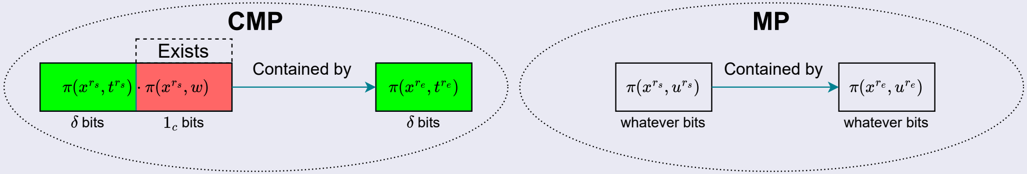
Example

Let $\boldsymbol{x}^i = (x[0], x[1], x[2], x[3])$ with $x[0].F = 0_c, x[1].F = x[2].F = \delta, x[3].F = 1_c$. Then, $\boldsymbol{M}^{i,\delta} = (0, 1, 1, 0)$, $\boldsymbol{M}^{i,1_c} = (0, 0, 0, 1)$, $\boldsymbol{M}^{i,0_c} = (1, 0, 0, 0)$. Obviously, $\boldsymbol{M}^{i,0_c} = \neg(\boldsymbol{M}^{i,\delta} \vee \boldsymbol{M}^{i,1_c})$.

Initial Core Monomial Prediction (CMP)

Definition

If $\pi(\boldsymbol{x}^{r_s}, \boldsymbol{t}^{r_s})$ can propagate to $\pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e})$ by CMP, we denote it by $\pi(\boldsymbol{x}^{r_s}, \boldsymbol{t}^{r_s}) \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e})$.



Core monomial trail

The connection of transitions $\pi(\boldsymbol{x}^{r_s}, \boldsymbol{t}^{r_s}) \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{r_s+1}, \boldsymbol{t}^{r_s+1}) \xrightarrow{\mathcal{C}} \cdots \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e})$ is called an r'-round core monomial trail, where $r' = r_e - r_s$.

Initial Core Monomial Prediction (CMP)

General propagation rules and models

For a specific round function $\boldsymbol{x}^i$, we list all pairs $(\boldsymbol{t}^i, \boldsymbol{t}^{i+1})$ that satisfy $\pi(\boldsymbol{x}^i, \boldsymbol{t}^i) \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{i+1}, \boldsymbol{t}^{i+1})$, then we establish certain rules such that for any pair $(\boldsymbol{t}^i, \boldsymbol{t}^{i+1})$, the transition from $\boldsymbol{t}^i$ to $\boldsymbol{t}^{i+1}$ fulfills the rules. These rules can be encoded as linear inequalities suitable for MILP modeling.

Rules for basic operations and S-box

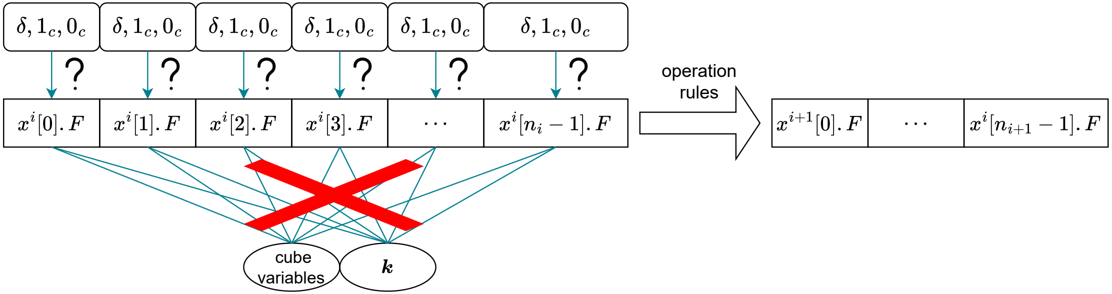
Using the general propagation rules and models, we can derive the propagation rules and models for basic operations COPY, AND, XOR and single S-box. This allowing us to construct an MILP model to capture the propagation of CMP through any cipher.

$$\boldsymbol{f} = \overset{\boldsymbol{f}^3}{\text{XOR}} \circ \overset{\boldsymbol{f}^2}{\text{AND}} \circ \overset{\boldsymbol{f}^1}{\text{COPY}} \circ \overset{\boldsymbol{f}^0}{\text{S-box}}$$

Indeterminate Flags

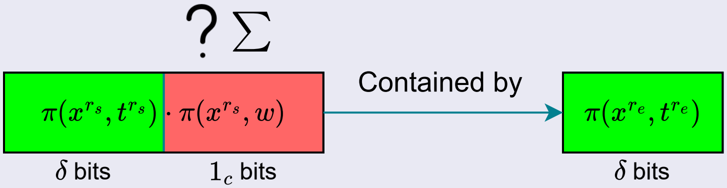
Indeterminate flag masks

The flags are no longer determined based on cube variables and $\boldsymbol{k}$. They become variables that can take δ, 1_c or 0_c now, but operation rules still hold. If the flags of $\boldsymbol{x}^i$ are determined, then the flags of any round state after round i are determined. Flag masks become variables as well.

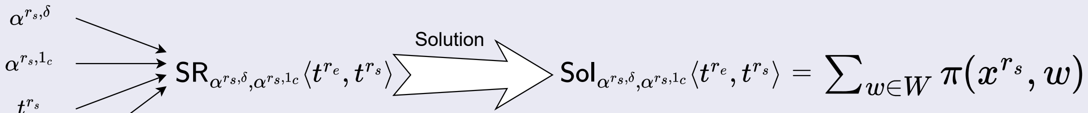


SR problem

SR problem



Instance of SR problem



$\alpha^{r_s,\delta}$
$\alpha^{r_s,1_c}$
t^{r_s}
t^{r_e}
$\mathsf{SR}_{\alpha^{r_s,\delta},\alpha^{r_s,1_c}}\langle t^{r_e}, t^{r_s}\rangle$
Solution
$\mathsf{Sol}_{\alpha^{r_s,\delta},\alpha^{r_s,1_c}}\langle t^{r_e}, t^{r_s}\rangle = \sum_{w\in W} \pi(x^{r_s}, w)$

$$W = \{w \mid w \preceq \alpha^{r_s,1_c}, \pi(x^{r_s}, w) \cdot \pi(x^{r_s}, t^{r_s}) \to \pi(x^{r_e}, t^{r_e})\}$$

Formalization

Assigning different values to the flag masks can lead to different propagation of CMP.

Relation between SR problem and CMP

$\mathsf{Sol}_{\boldsymbol{\alpha}^{r_s,\delta},\boldsymbol{\alpha}^{r_s,1_c}}\langle \boldsymbol{t}^{r_e}, \boldsymbol{t}^{r_s}\rangle$ is not 0 if and only if $\pi(\boldsymbol{x}^{r_s}, \boldsymbol{t}^{r_s}) \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e})$.

Reducing superpoly recovery to solving SR problem

Let $\boldsymbol{M}^{0,\delta}, \boldsymbol{M}^{0,1_c}$ and $\boldsymbol{M}^{0,0_c}$ take the particular values $\boldsymbol{\gamma}^{0,\delta}, \boldsymbol{\gamma}^{0,1_c}$ and $\boldsymbol{\gamma}^{0,0_c}$ respectively, where

$\gamma^{0,\delta}[j] = 1$ if and only if $x^0[j]$ is loaded with a cube varibale.

$\gamma^{0,1_c}[j] = 1$ if and only if $x^0[j]$ is loaded with a constant 1 or a secret variable.

$\gamma^{0,0_c} = \neg(\gamma^{0,\delta} \vee \gamma^{0,1_c})$.

We compute the values of $\boldsymbol{M}^{r,\delta}, \boldsymbol{M}^{r,1_c}, \boldsymbol{M}^{r,0_c}$ as $\boldsymbol{\gamma}^{r,\delta}, \boldsymbol{\gamma}^{r,1_c}, \boldsymbol{\gamma}^{r,0_c}$. Then computing $\mathsf{Coe}\langle z, t_I\rangle$ can be reduced to solving instances of the form $\mathsf{SR}_{\boldsymbol{\gamma}^{0,\delta},\boldsymbol{\gamma}^{0,1_c}}\langle \boldsymbol{t}^{r}, \boldsymbol{t}^{0}\rangle$.

Solve SR Problem Based on CMP Theory

Contribution of core monomial trail

$$\mathsf{Contr}\langle \ell \rangle = \mathsf{Sol}\langle t^{r_s+1}, t^{r_s}\rangle \times \mathsf{Sol}\langle t^{r_s+2}, t^{r_s+1}\rangle \times \cdots \times \mathsf{Sol}\langle t^{r_e-1}, t^{r_e-2}\rangle \times \mathsf{Sol}\langle t^{r_e}, t^{r_e-1}\rangle$$

$$\ell = \boxed{\pi(x^{r_s}, t^{r_s})} \xrightarrow{\mathcal{C}} \boxed{\pi(x^{r_s+1}, t^{r_s+1})} \xrightarrow{\mathcal{C}} \boxed{\quad\cdots\quad} \xrightarrow{\mathcal{C}} \boxed{\pi(x^{r_e-1}, t^{r_e-1})} \xrightarrow{\mathcal{C}} \boxed{\pi(x^{r_e}, t^{r_e})}$$

Solving SR problem by core monomial trails

Let $\boldsymbol{M}^{r_s,\delta}, \boldsymbol{M}^{r_s,1_c}$ take arbitrary values $\boldsymbol{\alpha}^{r_s,\delta}, \boldsymbol{\alpha}^{r_s,1_c}$ and $\boldsymbol{M}^{r_s,0_c} = \neg(\boldsymbol{\alpha}^{r_s,\delta} \vee \boldsymbol{\alpha}^{r_s,1_c})$. Given an instance $\mathsf{SR}\langle \boldsymbol{t}^{r_e}, \boldsymbol{t}^{r_s}\rangle$, if $\pi(\boldsymbol{x}^{r_s}, \boldsymbol{t}^{r_s}) \overset{\mathcal{C}}{\not\rightsquigarrow} \pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e})$, the solution of this instance is 0; otherwise, we can calculate the solution of this instance by

$$\mathsf{Sol}\langle \boldsymbol{t}^{r_e}, \boldsymbol{t}^{r_s}\rangle = \sum_{\ell \in \pi(\boldsymbol{x}^{r_s}, \boldsymbol{t}^{r_s}) \overset{\mathcal{C}}{\bowtie} \pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e})} \mathsf{Expr}\left\langle \mathsf{Contr}\langle \ell \rangle, \boldsymbol{x}^{r_s}\right\rangle .$$

Equivalence Between CMP and MP

Iterative variable substitution algorithm

$$\begin{tikzcd} \boxed{\pi(x^j, t^j)} \arrow[r, "\mathcal{C}"] & \boxed{\pi(x^{j+1}, t^{j+1})} \arrow[l, bend right, "c = \mathsf{Sol}\langle t^{j+1}, t^j \rangle"'] \end{tikzcd}$$

$$\boldsymbol{c}^j.\,\mathrm{append}(c) \qquad T[c] = \mathsf{Sol}\langle t^{j+1}, t^j \rangle$$

One-to-one correspondence between core monomial trail and monomial trail

$$\pi(\boldsymbol{x}^{r_s}, \boldsymbol{t}^{r_s}) \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{r_s+1}, \boldsymbol{t}^{r_s+1}) \xrightarrow{\mathcal{C}} \cdots \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e}) \Leftrightarrow$$

$$\pi(\boldsymbol{c}^{r_s} \| \cdots \| \boldsymbol{c}^{r_e-1} \| \boldsymbol{x}^{r_s}, \boldsymbol{w}^{r_s} \| \cdots \| \boldsymbol{w}^{r_e-1} \| \boldsymbol{t}^{r_s})$$

$$\to \pi(\boldsymbol{c}^{r_s+1} \| \cdots \| \boldsymbol{c}^{r_e-1} \| \boldsymbol{x}^{r_s+1}, \boldsymbol{w}^{r_s+1} \| \cdots \| \boldsymbol{w}^{r_e-1} \| \boldsymbol{t}^{r_s+1}) \to \cdots$$

$$\to \pi(\boldsymbol{c}^{r_e-1} \| \boldsymbol{x}^{r_e-1}, \boldsymbol{w}^{r_e-1} \| \boldsymbol{t}^{r_e-1})$$

$$\to \pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e}).$$

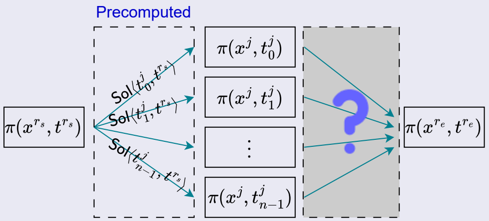
Forward and Backward Expansion

Divide-and-conquer

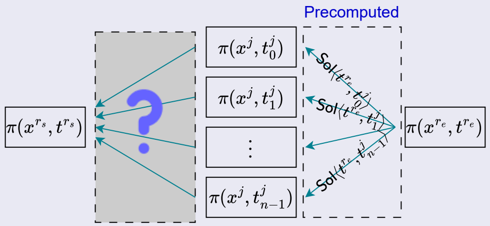
Given an instance $\mathsf{SR}_{\boldsymbol{\alpha}^{r_s,\delta},\boldsymbol{\alpha}^{r_s,1_c}}\langle \boldsymbol{t}^{r_e}, \boldsymbol{t}^{r_s}\rangle$, we have

$$\mathsf{Sol}\langle \boldsymbol{t}^{r_e}, \boldsymbol{t}^{r_s}\rangle = \sum_{\boldsymbol{t}^j} \mathsf{Expr}\left\langle \mathsf{Sol}\langle \boldsymbol{t}^{r_e}, \boldsymbol{t}^{j}\rangle, \boldsymbol{x}^{r_s}\right\rangle \cdot \mathsf{Sol}\langle \boldsymbol{t}^{j}, \boldsymbol{t}^{r_s}\rangle,$$

where the summation is over all $\boldsymbol{t}^j$'s that satisfy $\pi(\boldsymbol{x}^{r_s}, \boldsymbol{t}^{r_s}) \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{j}, \boldsymbol{t}^{j})$ and $\pi(\boldsymbol{x}^{j}, \boldsymbol{t}^{j}) \xrightarrow{\mathcal{C}} \pi(\boldsymbol{x}^{r_e}, \boldsymbol{t}^{r_e})$.


Precomputed
$\pi(x^{r_s}, t^{r_s})$
$\mathsf{Sol}\langle t_0^j, t^{r_s}\rangle$
$\mathsf{Sol}\langle t_1^j, t^{r_s}\rangle$
$\mathsf{Sol}\langle t_{n-1}^j, t^{r_s}\rangle$
$\pi(x^j, t_0^j)$
$\pi(x^j, t_1^j)$
$\vdots$
$\pi(x^j, t_{n-1}^j)$
?
$\pi(x^{r_e}, t^{r_e})$

Forward Expansion

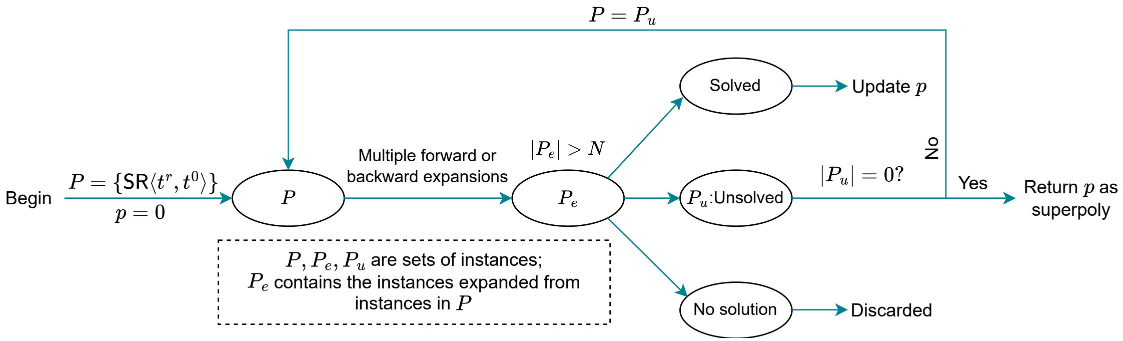

Precomputed
$\pi(x^{r_s}, t^{r_s})$
?
$\pi(x^j, t_0^j)$
$\pi(x^j, t_1^j)$
$\vdots$
$\pi(x^j, t_{n-1}^j)$
$\mathsf{Sol}\langle t^{r_e}, t_0^j\rangle$
$\mathsf{Sol}\langle t^{r_e}, t_1^j\rangle$
$\mathsf{Sol}\langle t^{r_e}, t_{n-1}^j\rangle$
$\pi(x^{r_e}, t^{r_e})$

Backward Expansion

Meet-in-the-middle (MITM) Framework

Applying forward and backward expansion interchangeably and recursively

Both the forward expansion and backward expansion divide an instance into multiple simpler instances. If they are applied interchangeably and recursively, we can develop a meet-in-the-middle framework.



$P = P_u$
Begin
$P = \{\mathsf{SR}\langle t^r, t^0\rangle\}$
$p = 0$
P
Multiple forward or backward expansions
$|P_e| > N$
P_e
Solved
Update p
P_u:Unsolved
$|P_u| = 0?$
No
Yes
Return p as superpoly
No solution
Discarded
P, P_e, P_u are sets of instances;
P_e contains the instances expanded from instances in P

New Results of Superpoly Recovery

- Summary of our new results of the superpoly recovery. #Cube means the number of cubes whose superpolies are recovered.

Cipher	Rounds	#Cube	Cube size	Overall time cost	Balancedness	#Involved key bits
Trivium	849	2	44	76 hours	0.50	80
	850	1	44	81 hours	0.50	80
	851	1	44	600 hours	0.50	80
Grain-128AEAD	192	1	94	9 days	0.49	128
Kreyvium	896	1	123	180 hours	0.50	30
	896	1	124	58 hours	0	0
	897	1	124	67 hours	0.50	60
	898	1	124	182 hours	0.50	75
	899	1	124	272 hours	0.50	121

- The balancedness is estimated with experiments on 2^{15} randomly-generated keys
- Conducted by Gurobi Solver (version 9.1.2) on a work station with 2×AMD EPYC 7302 16-core (32 siblings) Processor 3.3 GHz, (totally 64 threads), 256G RAM, and Ubuntu 20.10.

Key Recovery Attack

Based on memory-efficient variant of Möbius transform proposed by Dinur at EUROCRYPT 2021. For the equation $p(\boldsymbol{k}) = a$ established in the online phase of cube attack:

1. Guess m key bits.
2. Simplify p to a polynomial p' of unguessed key bits.
3. Perform Möbius transform on p'.
4. Exhaustive search.

Results

Cipher	Rounds	Type	Time	Memory (bits)	Reference
Trivium	849	Cube	2^{79}	2^{42}	This Work
	850	Cube	2^{79}	2^{46}	This Work
	851	Cube	2^{79}	2^{49}	This Work
Kreyvium	896	Cube	2^{127}	Practical	This Work
	897	Cube	2^{127}	Practical	This Work
	898	Cube	2^{127}	Practical	This Work
	899	Cube	2^{127}	Practical	This Work
	900	Cube	$2^{127.58}$	Practical	Previous Work[2]

[2] Hao Fan, Yonglin Hao, Qingju Wang, Xinxin Gong, and Lin Jiao: Key filtering in cube attacks from the implementation aspect. CANS 2023

Conclusion

- Refine the CMP theory to be perfectly accurate.
- Superpoly recovery and cube attacks for up to 851 rounds of TRIVIUM and 899 rounds of Kreyvium.

Thanks for your attention!

hejiahui2020@mail.sdu.edu.cn