7th Asian Workshop on Symmetric Key Cryptography (ASK2017)
9 Dec., 2017

Innovative R&D by NTT

Some Improvements of Non-Blackbox Cube Attacks

Yosuke Todo
NTT Secure Platform Laboratories

Today's Talk

1. Cube Attacks on Non-Blackbox Polynomials.
   - Proposed at CRYPTO 2017.
   - New generic tools for the cube attack.
2. Improvement 1.
   - Longer distinguisher is found when inactive bits are 0.
   - In detail, ePrint/2017/306.
3. Improvement 2.
   - Reduce the time complexity by exploiting low degree property of superpoly.
   - In detail, ePrint/2017/1063.

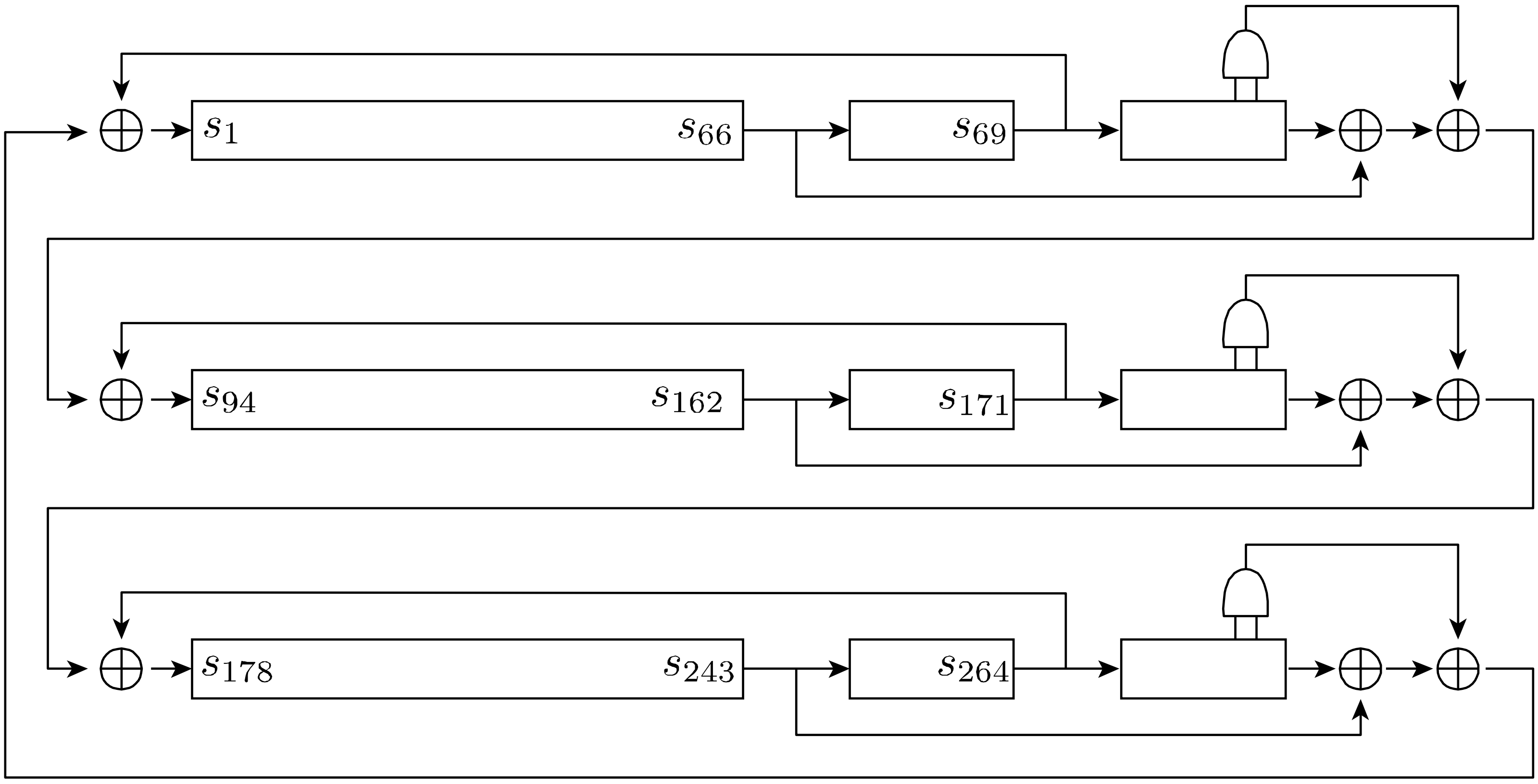
Cube Attacks on Non-Blackbox Polynomials (from CRYPTO2017)

Yosuke Todo, NTT Secure Platform Laboratories, Japan
Takanori Isobe, University of Hyogo, Japan
Yonglin Hao, Tsinghua Universtiy, China
Willi Meier, FHNW, Switzerland

Stream ciphers

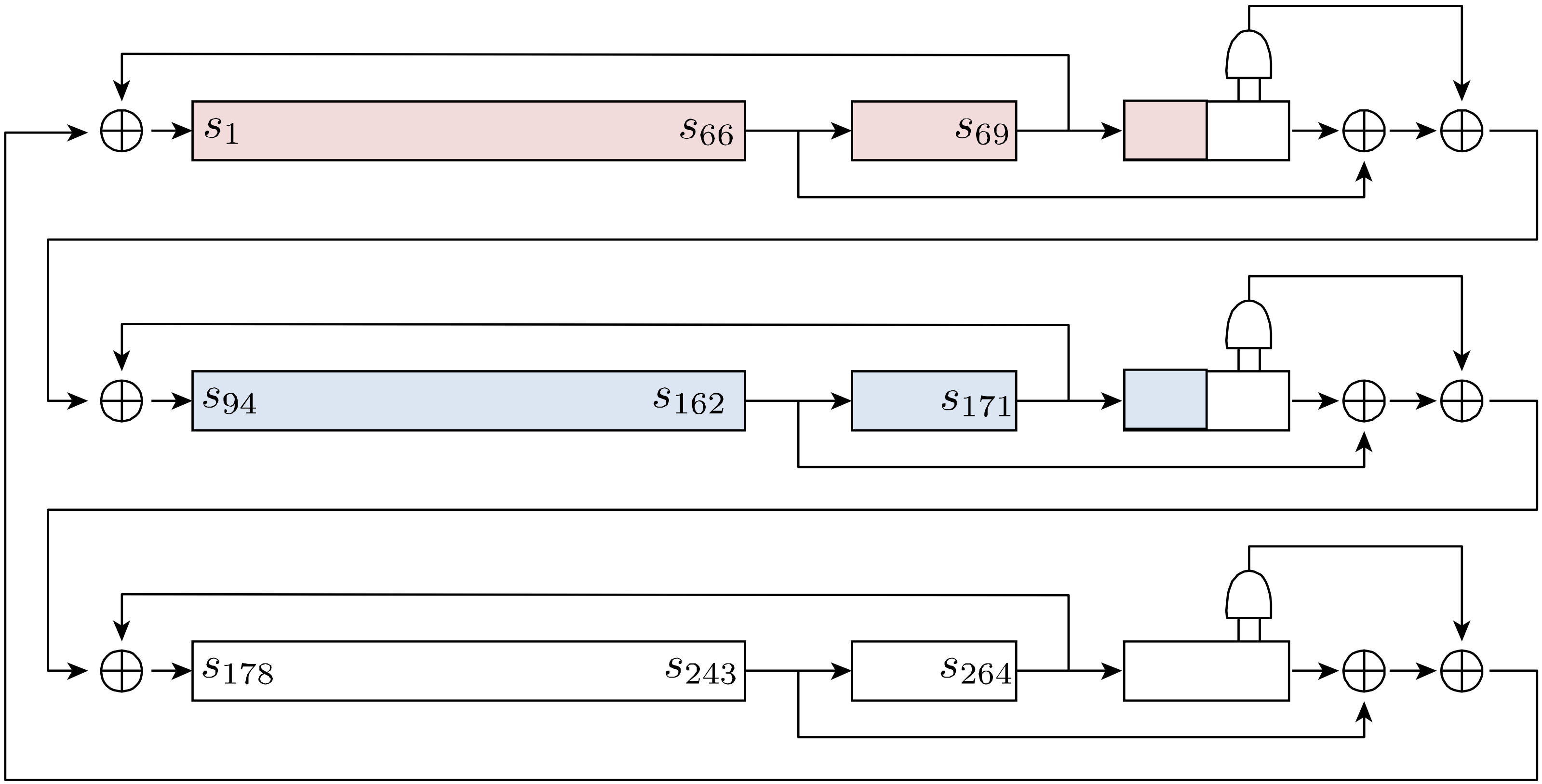
- Consists of two parts
  - Key initialization.
    - Secret key and public IV are loaded to the internal state.
    - Execute the update function iteratively w/o output of key-stream sequence.
  - Key-stream generation.
    - Update function outputs key-stream sequence.

Example of Trivium : Internal State



state size = 288 bits

Example of Trivium : Key initialization



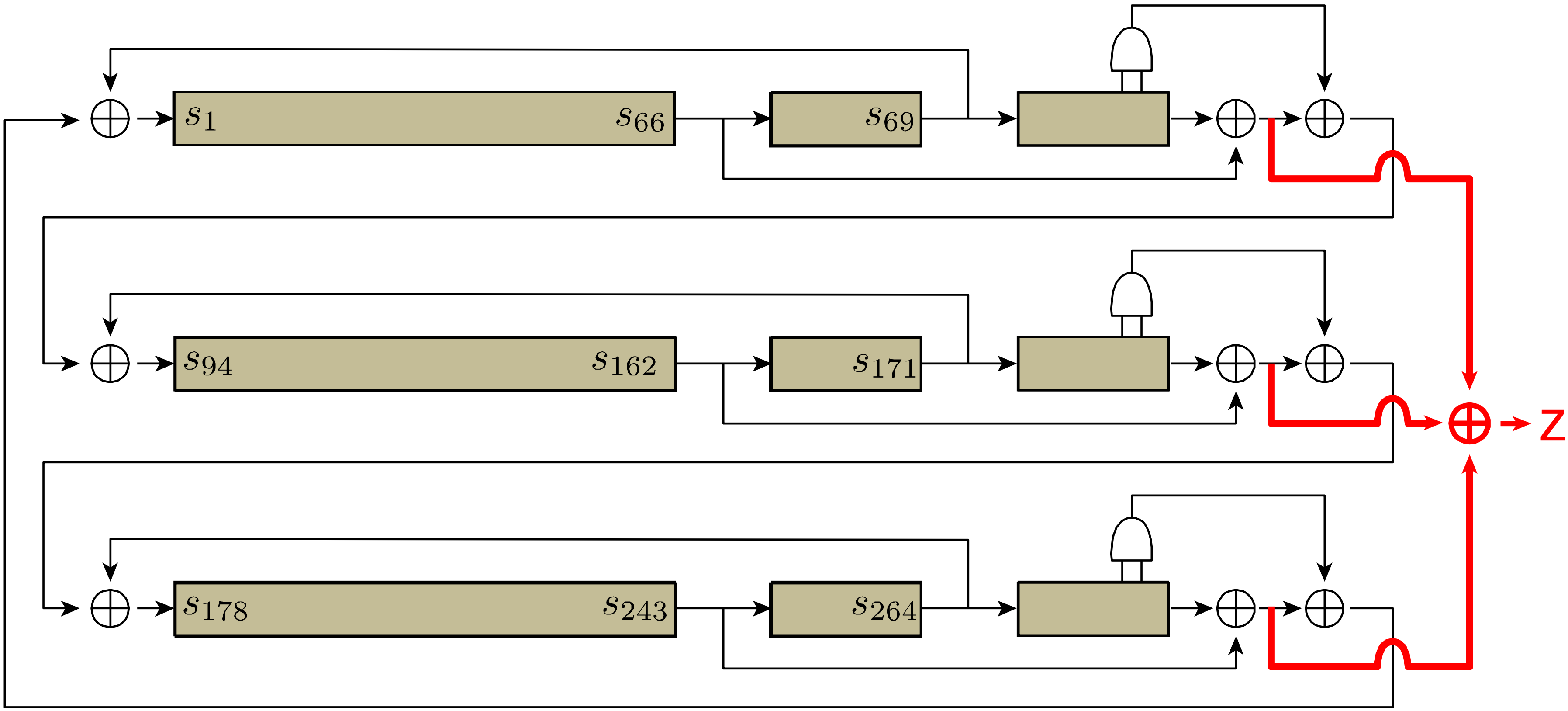
80-bit secret key

80-bit initialization vector

state size = 288 bits

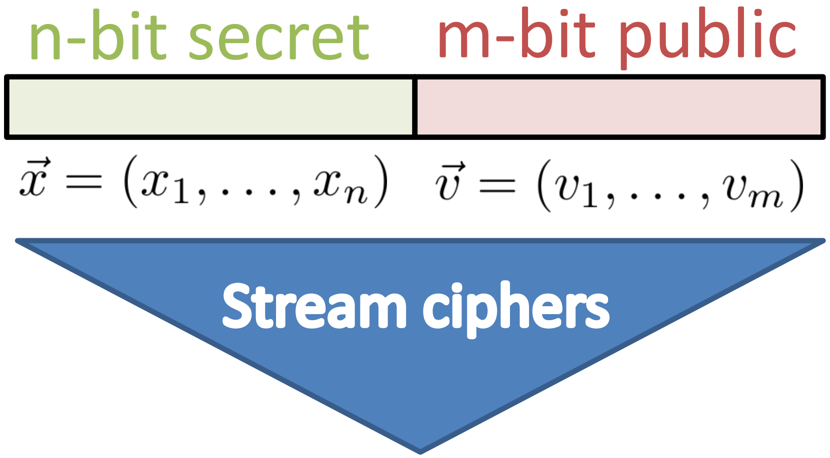
initialization = 1152 rounds

Example of Trivium : Output key stream



1 update function outputs 1-bit key stream.

Stream ciphers



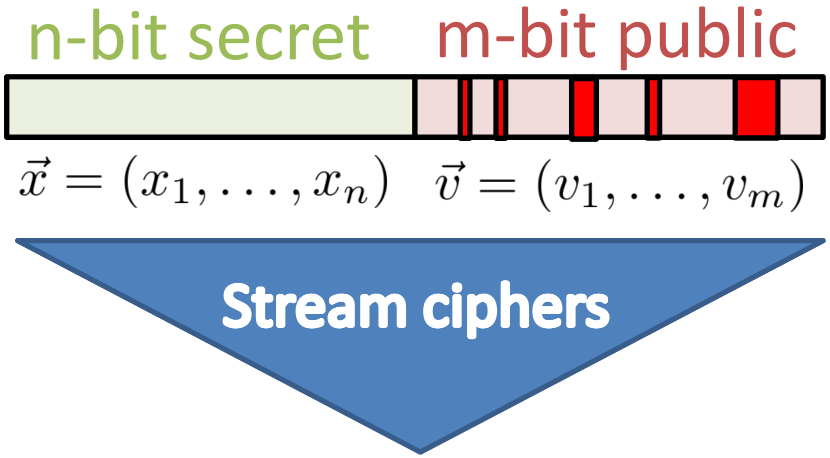
- $\vec{x}$ is n-bit secret variable.
- $\vec{v}$ is m-bit public variable.
- z is the first bit of the key stream.

$$z = f(\vec{x}, \vec{v}) = \bigoplus_{\vec{u} \in \mathbb{F}_2^{n+m}} a_{\vec{u}}^{f} \cdot \vec{x}^{\vec{u}} \cdot \vec{v}^{\vec{v}}$$

ex) $z = x_1 x_2 \oplus x_1 v_1 \oplus v_2 v_3$

Idea of the cube attack [DS09]

$$t_I = v_{i_1} \times \cdots \times v_{i_{|I|}}$$



- Let $I = \{i_1, \ldots, i_{|I|}\}$ be the indices of active bits.
- Let C_I be a set of $2^{|I|}$ values where v_i $(i \in I)$ is active.

$$z = f(\vec{x}, \vec{v}) = t_I \cdot p_I(\vec{x}, \vec{v}) + q_I(\vec{x}, \vec{v})$$

$$\bigoplus_{v \in C_I} z = p_I(\vec{x}, \vec{v})$$

Attackers recover secret variable $\vec{x}$ by analyzing $p_I(\vec{x}, \vec{v})$.

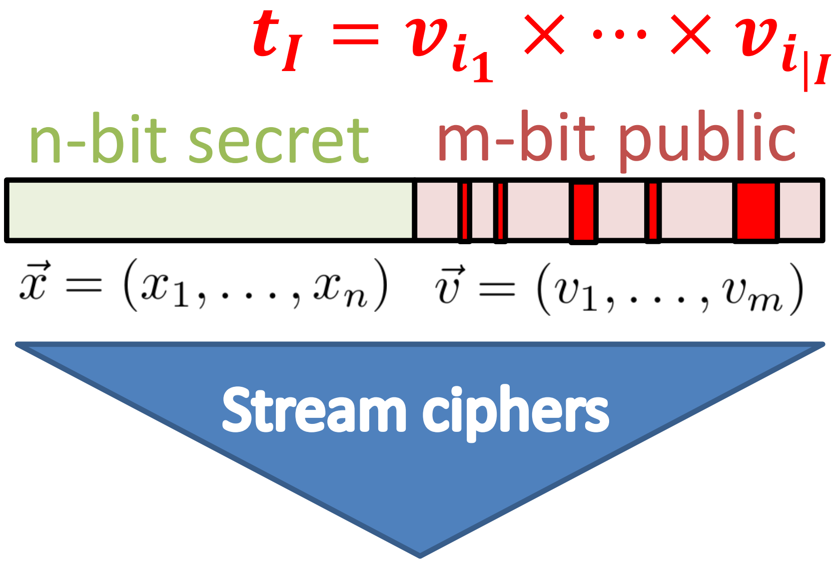
Concrete example

$$
\begin{aligned}
&f(v_1, v_2, v_3, x_1, x_2) \\
&= v_1 v_2 v_3 + v_1 v_2 x_1 + v_2 x_1 x_2 + v_1 v_2 + v_2 + v_3 x_2 + x_2 + 1 \\
&= v_1 v_2 (v_3 + x_1 + 1) + (v_2 x_1 x_2 + v_3 x_2 + v_2 + x_2 + 1)
\end{aligned}
$$

$$
\begin{cases}
t_I = v_1 v_2 \\
p_I(\vec{x}) = v_3 + x_1 + 1 \\
q_I(\vec{x}) = v_2 x_1 x_2 + v_3 x_2 + v_2 + x_2 + 1
\end{cases}
$$

$$
\bigoplus_{(v_1, v_2) \in \{0,1\}^2} f(\vec{v}, \vec{x}) = v_3 + x_1 + 1
$$

Unfortunately…



- Let $I = \{i_1, \ldots, i_{|I|}\}$ be the indices of active bits.
- Let C_I be a set of $2^{|I|}$ values where v_i $(i \in I)$ is active.

$$z = f(\vec{x}, \vec{v}) = t_I \cdot p_I(\vec{x}, \vec{v}) + q_I(\vec{x}, \vec{v})$$

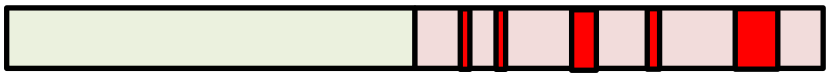
We cannot decompose $f(\vec{x}, \vec{v})$ because real stream cipher is complicated.

Experimental balckbox analysis

- How to recover $p_I(\vec{x}, \vec{v})$.
  1. Assume that p_I is linear function.
  2. Randomly choose $\vec{x}$.
     iteratively compute $\bigoplus_{\vec{v} \in C_I} f(\vec{x}, \vec{v}) = p_I(\vec{x}, \vec{v})$.
  3. Execute linearly test on many $\vec{x}$.
     Recover p_I under the assumption that it's linear.

- Drawback
  - The cube size is limited in the range of experimental, e.g., $|C_I| \leq 40$.

Motivation



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

$$z = f(\vec{x}, \vec{v})$$

Experimental cube attack

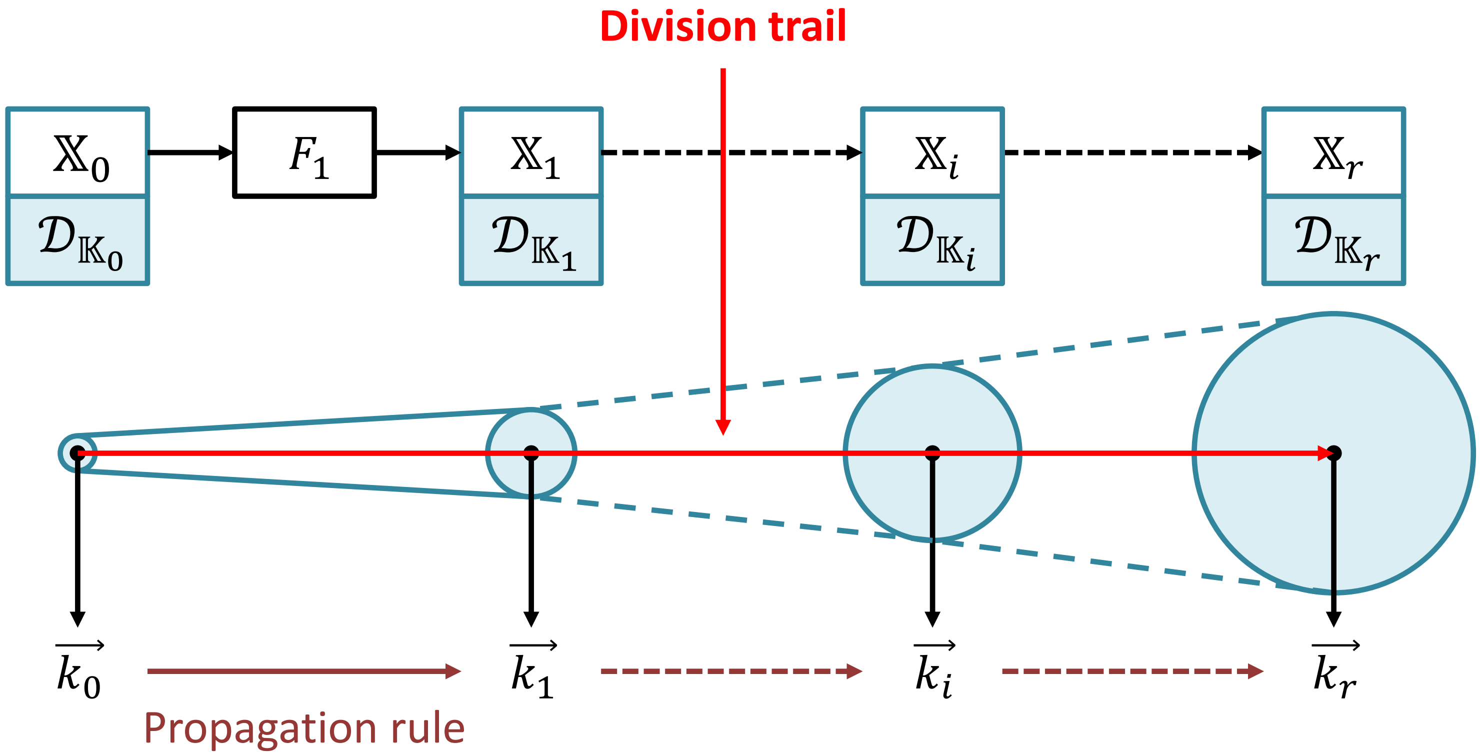
- Iterate linearly test experimentally.
- Recover the ANF of superpoly in real.

Theoretical cube attack

- Analyze the structure of superpoly.
- Evaluate the ub to recover its ANF.

We use the ***division property*** as a tool to analyze the sturcture of the superpoly.

Division Property



If there is NOT division trail $\vec{k}_0 \xrightarrow{f = F_r \circ \cdots \circ F_1} \vec{1}$,
the output of the Boolean function f is balanced.

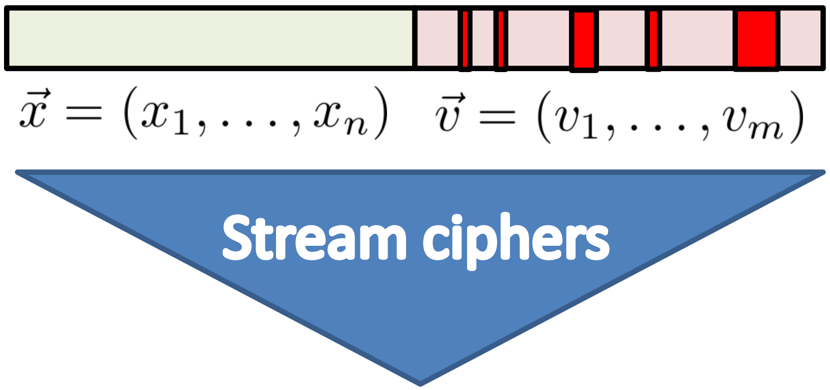
※ $\vec{v}^{\vec{k}_0} = t_I$.

How to analyze division trails?

- Programming from scratch.
  - Depth/Breadth First Search.
- CP-based approaches.
  - Mixed Integer Linear Programming.
  - SAT solver.
  - Constraint Programming.

Zero-sum distinguisher

$t_I = v_{i_1} \times \cdots \times v_{i_{|I|}}$



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

$$z = f(\vec{x}, \vec{v})$$

Division property

$$(\vec{0}, \vec{k}), \quad \vec{v}^{\vec{k}} = t_I$$

No division trail.

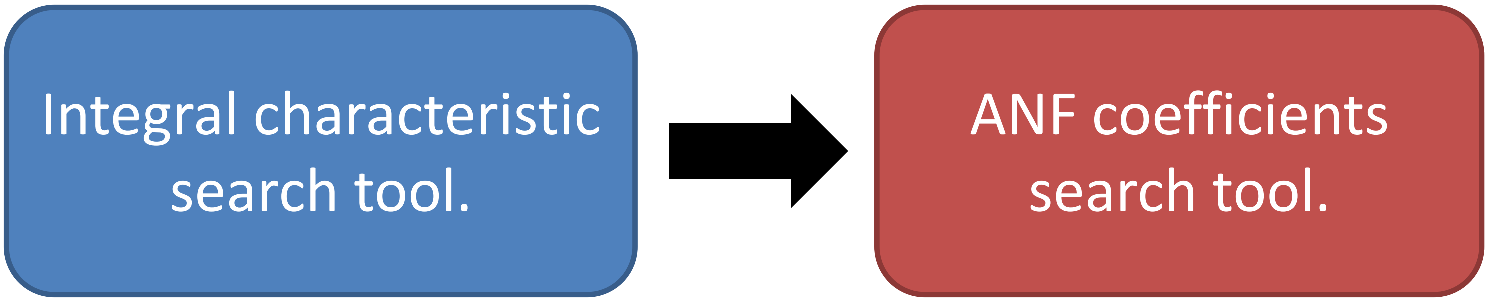
$$1$$

$$\bigoplus_{v \in C_I} f(\vec{x}, \vec{v}) = 0$$

Zero-sum distinguisher is trivially application.

How to recover the ANF.

- The role of division property.



- We revisit what the division property can do.

What division property can do

- Assuming there is NOT trail $\vec{k} \xrightarrow{f(\vec{x})} 1$,

$$\bigoplus_{C_I} f(\vec{x}) = p(\vec{x}) = \bigoplus_{\vec{u} \in \mathbb{F}_2^n | \vec{u} \succeq \vec{k}} a_{\vec{u}}^f \cdot \vec{x}^{\vec{u} \oplus \vec{k}}$$

is always zero for any $\vec{x}$.

- In other words,
  - $a_{\vec{u}}^f$ is always 0 for any $\vec{u} \succcurlyeq \vec{k}$.
- Division property can be used to analyze ANF coefficients.

Extension to key recovery.

- Assuming there is NOT trail $(\vec{e}_j, \vec{k}) \xrightarrow{f(\vec{x},\vec{v})} 1$, $a^f_{\vec{u}}$ is always 0 for any $\vec{u} \succcurlyeq (\vec{e}_j || \vec{k})$.
- Then,

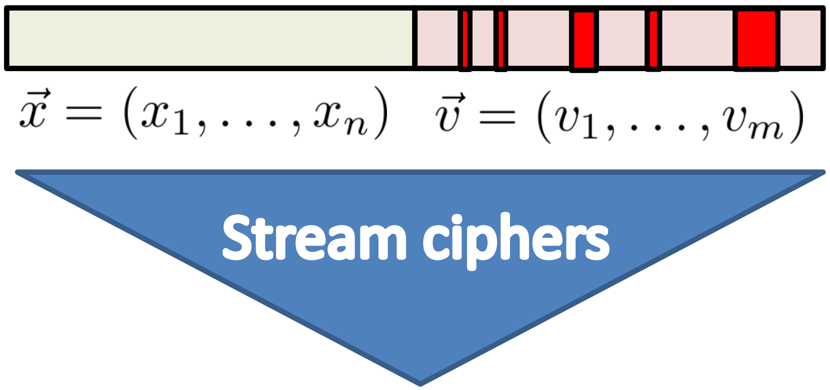
$$\bigoplus_{C_I} f(\vec{x}, \vec{v}) = p(\vec{x}, \vec{v}) = \bigoplus_{\vec{u} \in \mathbb{F}_2^{n+m} | \vec{u} \succeq (\vec{0} || \vec{k}_I)} a^f_{\vec{u}} \cdot (\vec{x} || \vec{v})^{\vec{u} \oplus (\vec{0} || \vec{k}_I)}$$

$$= \bigoplus_{\vec{u} \in \mathbb{F}_2^{n+m} | \vec{u} \succeq (\vec{0} || \vec{k}_I), u_j = 0} a^f_{\vec{u}} \cdot (\vec{x} || \vec{v})^{\vec{u} \oplus (\vec{0} || \vec{k}_I)}.$$

- The superpoly is independent of x_j becase $x_j^{u_j} = x_j^0 = 1$.

Summary of division property-based cube

$t_I = v_{i_1} \times \cdots \times v_{i_{|I|}}$



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

$z = f(\vec{x}, \vec{v})$

Division property

$(\vec{e}_j, \vec{k}), \quad \vec{v}^{\vec{k}} = t_I$

1

No division trail.

x_j is not involved to $\bigoplus_{v \in C_I} f(\vec{x}, \vec{v})$

By repeating this procedure,
we can distinguish which secret-key bits are involved.

NTT

Applications.

Applications	Previous Best	New Best
Trivium	799	832
Grain128a	177	183
ACORN	503	704
Kreyvium	--	872

※Applications to Kreyvium are explained the full version (ePrint/2017/306)

1st Improvement.
Exploiting constant-0 cubes.
(from ePrint/2017/306)

Yosuke Todo, NTT Secure Platform Laboratories, Japan

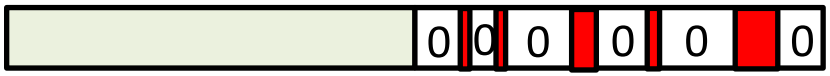
Takanori Isobe, University of Hyogo, Japan

Yonglin Hao, Tsinghua Universtiy, China

Willi Meier, FHNW, Switzerland

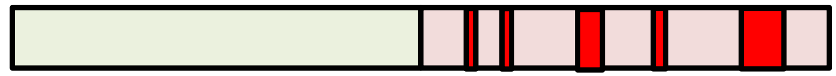
Motivation

We want to fill the gap from other works.



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

Non-active bits are always 0 in many previous cubes.



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

Non-active bits are any value in our cubes.

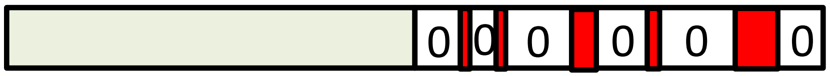
$$f(v_1, v_2, v_3, x_1, x_2)$$
$$= v_1 v_2 (v_3 + x_1 + v_3 x_2 + 1) + (v_2 x_1 x_2 + v_3 x_2 + v_2 + x_2 + 1)$$

$$p(v_3, x_1, x_2) = v_3 + x_1 + v_3 x_2 + 1$$
$$p(0, x_1, x_2) = x_1 + 1$$

NTT

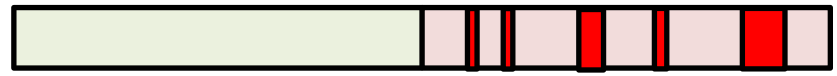
Motivation

We want to fill the gap from other works.



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

Non-active bits are always 0 in many previous cubes.



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

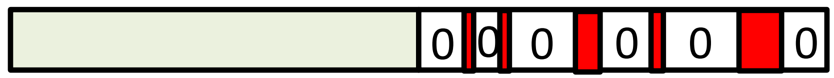
Non-active bits are any value in our cubes.

- 0-constant cubes bring more powerful attack generally.
- Liu's cube (at CRYPTO17) also uses 0-constant cube.

We need a new technique to exploit 0-constant cube with the division property.

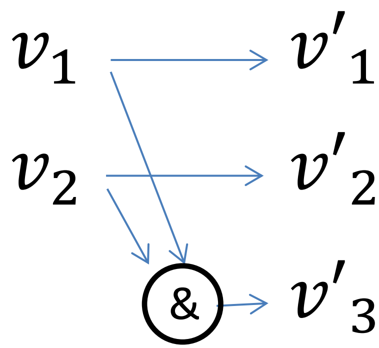
Exploiting the constant 0

- Non-cube bits are 0.



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

  - If non-cube bit is fixed to 0, the propagation of the division property is restricted.

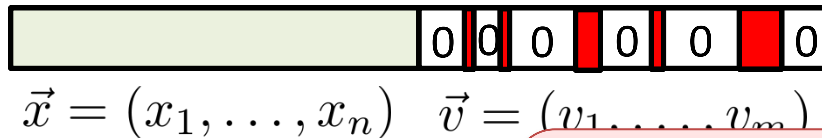


(0,0) → (0,0,0)
(1,0) → (1,0,0), (0,0,1)
(0,1) → (0,1,0), (0,0,1)
(1,1) → (1,1,0), (0,0,1)

※Similar technique was already used by Sun et al's work in the context of the integral distinguisher (ePrint/2016/1101).

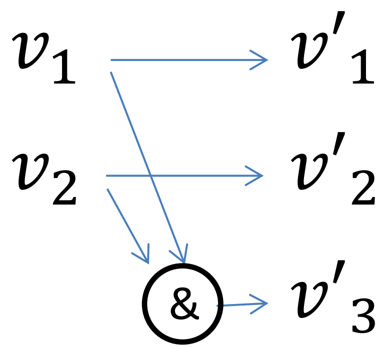
Exploiting the constant 0

- Non-cube bits are 0.



  - If non-cube bit is fixed t
    the division property is

Condition : $v_2 = 0$



(0,0) → (0,0,0)

(1,0) → (1,0,0), ~~(0,0,1)~~

~~(0,1) → (0,1,0), (0,0,1)~~

~~(1,1) → (1,1,0), (0,0,1)~~

impossible propagation

※Similar technique was already used by Sun et al's work in the context of the integral distinguisher (ePrint/2016/1101).

Summary of distinguishing attacks.

Applications	rounds	cube size	type	method
Trivium	837	37	zero sum	Liu & ours
	838	38	zero sum	ours
	842	37	biased sum	experimental (Liu)
Kreyvium	872	61	zero sum	Liu & ours
	873	62	zero sum	ours

- We can revisit Meicheng Liu's result.
- We can improve the zero-sum distinguisher on Trivium and Kreyvium from Liu's result.
- We haven't tried experimental approaches.
  - There is the possibility 38-dimensinal cube derives stronger biased sum distinguisher.

Comparison between Liu's result

	Liu's algorithm	Division property	Comment
Complexity	**WIN**	LOSE	We need to ask for solver's help to evaluate the division trails.
Accuracy	LOSE	**WIN** (w/ improved technique.)	I find some instances that division property is better than Liu's algorithm.
Flexibility	LOSE	**WIN**	Division property is applicable to arbitrary ciphers.

- Recommendation.
  - If the solver can stop, division property is better.
  - Otherwise, e.g., the state size is too large, we have to use Liu's algorithm.

NTT

Innovative R&D by NTT

2^{nd} *Improvement.* Exploiting Low Degree Property of Superpoly *(ePrint/2017/1063)*

Qingju Wang, DTU, Denmark
Yonglin Hao, Tsinghua Universtiy, China
Yosuke Todo, NTT Secure Platform Laboratories, Japan
Chaoyun Li, KU Leuven, Belgium
Takanori Isobe, University of Hyogo, Japan
Willi Meier, FHNW, Switzerland

Motivation

Experimental cube attack.

- Superpoly is assumed as linear or quadratic.
  - Experimental cube recovers superpoly efficiently by exploiting this low degree property.

Take one step further!!

- We also exploit this low-degree property with the division property.
  - The upper bound of the degree on superpoly is estimated.
  - The time complexity is more reduced.

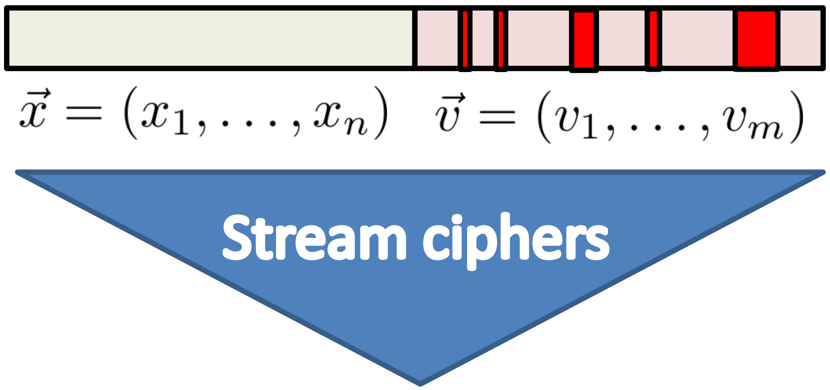
If the superpoly is low degree,...

- If the degree is at most d.
  - We don't need to evaluate the ANF coefficients whose degree of monomials is more than d.
  - The time complexity is reduced from

$$2^{|I|+|J|} \text{ to } 2^{|I|} \times \sum_{i=0}^{d} \binom{|J|}{i}$$

How degree is evaluated?

$t_I = v_{i_1} \times \cdots \times v_{i_{|I|}}$



$\vec{x} = (x_1, \ldots, x_n)$ $\vec{v} = (v_1, \ldots, v_m)$

$z = f(\vec{x}, \vec{v})$

Division property

$(\vec{\ell}, \vec{k}), \quad \vec{v}^{\vec{k}} = t_I$

1

No division trail.

under this condition

$\text{maximize} \sum_{j \in J} \ell_j$

This maximum value corresponds the upper bound of the algebraic degree of the superpoly.

Applications and results

Applications	rounds	cube size	|J|	time	ref.
Trivium	832	72	5	2^{77}	crypto17
	839	78	1	2^{79}	ePrint/2017/1063
Kreyvium	872	85	39	2^{124}	ePrint/2017/306
	888	102	36	2^{111.38}	ePrint/2017/1063

- Focus on 888-round attack on Kreyvium.
  - The number of involved secret variables is 36.
  - Previous estimations requires 2^{138} complexity.
  - However, since the degree of superpoly is at most 2, we can dramatically reduce the complexity.

Conclusion

- Division property based cube attacks
  - A new generic framework to evaluate the security against cube attacks.
  - It brings best key-recovery attacks against Trivium, Grain128a, ACORN, Kreyvium.
- Further improvements
  - Exploiting constant-0 cube brings more powerful superpoly recovery attacks.
  - Exploiting low degree property of the superpoly reduce the time complexity to recover the superpoly.