Revisiting Division Property Based Cube Attacks: Key-Recovery or Distinguishing Attacks?

Chen-Dong Ye and Tian Tian

Outline

Cube Attacks

The output bit z is a tweakable Boolean function f on secret key variables and IV variables, i.e., $z = f(\boldsymbol{x}, \boldsymbol{v})$.

- For a given public variable set $I = \{v_{i_1}, v_{i_2}, \dots, v_{i_d}\}$, f could be rewritten as

$$f(\boldsymbol{x}, \boldsymbol{v}) = t_I \cdot p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I) \oplus q(\boldsymbol{x}, \boldsymbol{v}).$$

  - $t_I = \prod_{j=1}^{d} v_{i_j}$
  - q is the sum of terms that miss at least one variable in I

- The basic idea of cube attacks

  $p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I) = \bigoplus_{(v_{i_1}, v_{i_2}, \dots, v_{i_d}) \in \mathbb{F}_2^d} f(\boldsymbol{x}, \boldsymbol{v})$

  - variables in I are called cube variables, the remaining variables in $\boldsymbol{v}$ are called non-cube variables
  - linear space C_I spanned by cube variables is called a cube
  - $p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I)$ is called the superpoly of I in f

Cube Attacks

The output bit z is a tweakable Boolean function f on secret key variables and IV variables, i.e., $z = f(\boldsymbol{x}, \boldsymbol{v})$.

- For a given public variable set $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$, f could be rewritten as

$$f(\boldsymbol{x}, \boldsymbol{v}) = t_I \cdot p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I) \oplus q(\boldsymbol{x}, \boldsymbol{v}).$$

  - $t_I = \prod_{j=1}^{d} v_{i_j}$
  - q is the sum of terms that miss at least one variable in I

- The basic idea of cube attacks

  $p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I) = \bigoplus_{(v_{i_1}, v_{i_2}, \ldots, v_{i_d}) \in \mathbb{F}_2^d} f(\boldsymbol{x}, \boldsymbol{v})$

  - variables in I are called cube variables, the remaining variables in $\boldsymbol{v}$ are called non-cube variables
  - linear space C_I spanned by cube variables is called a cube
  - $p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I)$ is called the superpoly of I in f

Cube Attacks

The output bit z is a tweakable Boolean function f on secret key variables and IV variables, i.e., $z = f(\boldsymbol{x}, \boldsymbol{v})$.

- For a given public variable set $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$, f could be rewritten as

$$f(\boldsymbol{x}, \boldsymbol{v}) = t_I \cdot p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I) \oplus q(\boldsymbol{x}, \boldsymbol{v}).$$

  - $t_I = \prod_{j=1}^{d} v_{i_j}$
  - q is the sum of terms that miss at least one variable in I

- The basic idea of cube attacks
  $p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I) = \bigoplus_{(v_{i_1}, v_{i_2}, \ldots, v_{i_d}) \in \mathbb{F}_2^d} f(\boldsymbol{x}, \boldsymbol{v})$
  - variables in I are called cube variables, the remaining variables in $\boldsymbol{v}$ are called non-cube variables
  - linear space C_I spanned by cube variables is called a cube
  - $p_I(\boldsymbol{x}, \boldsymbol{v} \setminus I)$ is called the superpoly of I in f

Cube Attacks and Cube Testers

- Off-line phase
  - independent of the secret key
  - find some useful superpolies to recover the secret key
- on-line phase
  - solve a system of equations derived from previously found superpolies under the real key

cube testers

Finding superpolies which could be distinguished from random polynomial, such as 0-constant polynomial(called zero-sum distinguishers).

Cube Attacks and Cube Testers

- Off-line phase
  - independent of the secret key
  - find some useful superpolies to recover the secret key
- on-line phase
  - solve a system of equations derived from previously found superpolies under the real key

cube testers

Finding superpolies which could be distinguished from random polynomial, such as 0-constant polynomial(called zero-sum distinguishers).

The Division Property Based Cube Attacks

- Originally, linearity tests are applied to find linear superpolies in cube attacks;
- Complexity: $c \times 2^{|I|}$, where I is a set of cube variables;
- $|I|$ is confined to around 40;

At CRYPTO 2017, Y. Todo et al applied the division property to cube attacks for the first time.

- Division property is used to analyse the algebraic normal form(ANF) of the output bit $f(\boldsymbol{x}, \boldsymbol{v})$.
- Cubes with large sizes could be used.

The Division Property Based Cube Attacks

- Originally, linearity tests are applied to find linear superpolies in cube attacks;
- Complexity: $c \times 2^{|I|}$, where I is a set of cube variables;
- $|I|$ is confined to around 40;

At CRYPTO 2017, Y. Todo et al applied the division property to cube attacks for the first time.

- Division property is used to analyse the algebraic normal form(ANF) of the output bit $f(\boldsymbol{x}, \boldsymbol{v})$.
- Cubes with large sizes could be used.

The Development of the Division Property

- Division property, as a generalization of the integral property, was first proposed at EUROCRYPT 2015.
- At FSE 2016, bit-based division property was proposed to investigate integral characteristics for bit-based block ciphers.
- At ASIACRYPT 2016, Xiang et al. combine mixed integer linear programming (MILP) methods with division property. With the aid of MILP, bit-based division property could be applied widely.
- ...

The Development of the Division Property

- Division property, as a generalization of the integral property, was first proposed at EUROCRYPT 2015.
- At FSE 2016, bit-based division property was proposed to investigate integral characteristics for bit-based block ciphers.
- At ASIACRYPT 2016, Xiang et al. combine mixed integer linear programming (MILP) methods with division property. With the aid of MILP, bit-based division property could be applied widely.
- ...

The Development of the Division Property

- Division property, as a generalization of the integral property, was first proposed at EUROCRYPT 2015.
- At FSE 2016, bit-based division property was proposed to investigate integral characteristics for bit-based block ciphers.
- At ASIACRYPT 2016, Xiang et al. combine mixed integer linear programming (MILP) methods with division property. With the aid of MILP, bit-based division property could be applied widely.
- $\cdots$

The Development of the Division Property Based Cube Attacks

- At CRYPTO 2017, Y. Todo et al. proposed the division property based cube attacks.
- Soon after proposing division property based cube attacks, Y. Todo et al.: Considering the effect of non-cube variables which are set to 0
- At CRYPTO 2018, by proposing some new techniques, Wang et al. improved the division property based cube attacks.
  - Flag technique
  - Degree Evaluation Method
  - Precise/Relaxed Term Enumeration

The Development of the Division Property Based Cube Attacks

- At CRYPTO 2017, Y. Todo et al. proposed the division property based cube attacks.
- Soon after proposing division property based cube attacks, Y. Todo et al.: Considering the effect of non-cube variables which are set to 0
- At CRYPTO 2018, by proposing some new techniques, Wang et al. improved the division property based cube attacks.
  - Flag technique
  - Degree Evaluation Method
  - Precise/Relaxed Term Enumeration

The Development of the Division Property Based Cube Attacks

- At CRYPTO 2017, Y. Todo et al. proposed the division property based cube attacks.
- Soon after proposing division property based cube attacks, Y. Todo et al.: Considering the effect of non-cube variables which are set to 0
- At CRYPTO 2018, by proposing some new techniques, Wang et al. improved the division property based cube attacks.
  - Flag technique
  - Degree Evaluation Method
  - Precise/Relaxed Term Enumeration

Motivations

- Division property based cube attacks: For a cube set I, a set J including all the key variables in the superpoly could be figured out.
- The bit-based division property can not analyse the ANF of the output bit $f(\boldsymbol{x}, \boldsymbol{v})$ precisely, since it does not consider the terms vanished by OXR operation.
- Even though the set J is not empty, the superpoly p_I may be constant.

Motivations

- Division property based cube attacks: For a cube set I, a set J including all the key variables in the superpoly could be figured out.
- The bit-based division property can not analyse the ANF of the output bit $f(\boldsymbol{x}, \boldsymbol{v})$ precisely, since it does not consider the terms vanished by OXR operation.
- Even though the set J is not empty, the superpoly p_I may be constant.

Motivations

- Division property based cube attacks: For a cube set I, a set J including all the key variables in the superpoly could be figured out.
- The bit-based division property can not analyse the ANF of the output bit $f(\boldsymbol{x}, \boldsymbol{v})$ precisely, since it does not consider the terms vanished by OXR operation.
- Even though the set J is not empty, the superpoly p_I may be constant.

Motivations

- To keep the validity of key-recovery attacks: Weak Assumption

Assumption (Weak Assumption)

For a cube I, there are many values in the constant part of IV whose corresponding superpoly is not a constant function.

However, Weak Assumption does not always hold. It indicates that some so-called key-recovery attacks may be distinguishing attacks only.

Our Contribution

We propose a new method which is able to recover the superpoly $p_I(\boldsymbol{x}, \boldsymbol{v})$ of I in the output $z(\boldsymbol{x}, \boldsymbol{v})$.

- For I_1, we recover the superpoly $p_{I_1}(\boldsymbol{x}, \boldsymbol{v})$ of I_1 in the output bit of the 832-round Trivium, given by $p_{I_1}(\boldsymbol{x}, \boldsymbol{v}) = v_{68}v_{78} \cdot (x_{58} \oplus v_{70}) \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$.
  - The 80-bit key could be recovered in less than $2^{79} + 2^{73}$ requests.
- For the cubes proposed in [WHT+18], we prove that their superpolies in the output bit of 833-, 835-, 836- and 839-round Trivium are 0-constant. Hence, the key-recovery attacks are all distinguishing attack actually.

Our Contribution

We propose a new method which is able to recover the superpoly $p_I(\boldsymbol{x}, \boldsymbol{v})$ of I in the output $z(\boldsymbol{x}, \boldsymbol{v})$.

- For I_1, we recover the superpoly $p_{I_1}(\boldsymbol{x}, \boldsymbol{v})$ of I_1 in the output bit of the 832-round Trivium, given by $p_{I_1}(\boldsymbol{x}, \boldsymbol{v}) = v_{68}v_{78} \cdot (x_{58} \oplus v_{70}) \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$.
  - The 80-bit key could be recovered in less than $2^{79} + 2^{73}$ requests.
- For the cubes proposed in [WHT$^+$18], we prove that their superpolies in the output bit of 833-, 835-, 836- and 839-round Trivium are 0-constant. Hence, the key-recovery attacks are all distinguishing attack actually.

Detailed Results

Table 1: Results on Trivium variants with up to 839 rounds

Rounds	Cube	"Involved" Key Variables	Exact Superpoly
832	I_1	$x_{34}, x_{58}, x_{59}, x_{60}, x_{61}$ [TIHM17]	p_{I_1}
833	I_2	$x_{49}, x_{58}, x_{60}, x_{64}, x_{74}, x_{75}, x_{76}$ [WHT+18]	0-constant
833	I_3	x_{60} [WHT+18]	0-constant
835	I_4	x_{57} [WHT+18]	0-constant
836	I_5	x_{57} [WHT+18]	0-constant
839	I_6	x_{61} [WHT+18]	0-constant

$p_{I_1} = v_{68}v_{78} \cdot (x_{58} \oplus v_{70}) \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$

$I_1 = \{1, 2, \ldots, 65, 67, 69, \ldots, 79\}$

$I_2 = \{1, 2, \ldots, 67, 69, 71, \ldots, 79\}$

$I_3 = \{1, 2, \ldots, 69, 71, 73, \ldots, 79\}$

$I_4 = \{1, 2, 3, 4, 6, 7, \ldots, 50, 52, 53, \ldots, 64, 66, 67, \ldots, 80\}$

$I_5 = \{1, \ldots, 11, 13, \ldots, 42, 44, \ldots, 80\}$

$I_6 = \{1, \ldots, 33, 35, \ldots, 46, 48, \ldots, 80\}$

The Bit-based Division Property

Definition (**Bit-Based Division Property**)

Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{K}$ be a set whose elements take an n-dimensional bit vector. When the multiset $\mathbb{X}$ has the division property $D_{\mathbb{K}}^{1^n}$, it fulfills the following conditions:

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}} = \begin{cases} unknown & \text{if there exists } \boldsymbol{k} \text{ in } \mathbb{K} \text{ s.t. } \boldsymbol{u} \succeq \boldsymbol{k}, \\ 0 & \text{otherwise.} \end{cases}$$

where $\boldsymbol{u} \succeq \boldsymbol{k}$ if and only if $u_i \geq k_i$ for all i and $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=1}^{n} x_i^{u_i}$.

The Division Trail

Definition (Division Trail [XZBL16])

Let us consider the propagation of the division property $\{\boldsymbol{k}\} = \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \cdots \to \mathbb{K}_r$. Moreover, for any vector $\boldsymbol{k}_{i+1}^* \in \mathbb{K}_{i+1}$, there must exist a vector $\boldsymbol{k}_i^* \in \mathbb{K}_i$ such that $\boldsymbol{k}_i^*$ can propagate to $\boldsymbol{k}_{i+1}^*$ by the propagation rules of division property. Furthermore, for
$(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$ if $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$ for $i \in \{0, 1, \ldots, r-1\}$, we call $\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_r$ an r-round division trail.

The Basic of Division Property Based Cube Attacks

Lemma ([TIHM17])

Let $f(\boldsymbol{x})$ be a polynomial from $\mathbb{F}_2^n$ to $\mathbb{F}_2$ and $a_{\boldsymbol{u}}^f$ be the ANF coefficients. Let $\boldsymbol{k}$ be an n-dimensional bit vector. If there is no division trail such that $\boldsymbol{k} \xrightarrow{f} 1$, then $a_{\boldsymbol{u}}^f$ is always 0 for $\boldsymbol{u} \succeq \boldsymbol{k}$.

The Basic of Division Property Based Cube Attacks

Proposition ([TIHM17])

Let $f(\boldsymbol{x}, \boldsymbol{v})$ be a polynomial, where $\boldsymbol{x}$ and $\boldsymbol{v}$ denote the secret and public variables, respectively. For a set of indices $I = \{i_1, i_2, \ldots, i_d\} \subset \{1, 2, \ldots, m\}$, let C_I be a set where $\{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$ traverse all $2^{|I|}$ values and the other public variables are set to constants. Let $\boldsymbol{k}_I$ be an m-dimensional bit vector such that $\boldsymbol{v}^{\boldsymbol{k}_I} = t_I = v_{i_1} v_{i_2} \cdots v_{i_d}$, i.e., $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. If there is no division trail such that $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{f} 1$, then x_j is not involved in the superpoly of the cube C_I.

The Division Property Based Cube Attacks

For a cube I, with the above proposition, we could obtain a set J of key bits, "involved" in the superpoly of I in f.

phase 1 Try some randomly chosen assignments of non-cube variables until the proper assignments such that the corresponding superpoly p_I is non-constant are found.

phase 2 Set non-cube variables to the previous found assignments and calculate the superpoly under the real key, denoted by a. Then, only the values of key such that $p_I = a$ are reserved.

phase 3 Guess the remaining secret bits to recover the entire secret key.

Actually, the set of key bits that the superpoly depends on is a sub set of J.

The Division Property Based Cube Attacks

For a cube I, with the above proposition, we could obtain a set J of key bits, "involved" in the superpoly of I in f.

- **phase 1** Try some randomly chosen assignments of non-cube variables until the proper assignments such that the corresponding superpoly p_I is non-constant are found.
- **phase 2** Set non-cube variables to the previous found assignments and calculate the superpoly under the real key, denoted by a. Then, only the values of key such that $p_I = a$ are reserved.
- **phase 3** Guess the remaining secret bits to recover the entire secret key.

Actually, the set of key bits that the superpoly depends on is a sub set of J.

The Division Property Based Cube Attacks

For a cube I, with the above proposition, we could obtain a set J of key bits, "involved" in the superpoly of I in f.

- **phase 1** Try some randomly chosen assignments of non-cube variables until the proper assignments such that the corresponding superpoly p_I is non-constant are found.
- **phase 2** Set non-cube variables to the previous found assignments and calculate the superpoly under the real key, denoted by a. Then, only the values of key such that $p_I = a$ are reserved.
- **phase 3** Guess the remaining secret bits to recover the entire secret key.

Actually, the set of key bits that the superpoly depends on is a sub set of J.

The Division Property Based Cube Attacks

For a cube I, with the above proposition, we could obtain a set J of key bits, "involved" in the superpoly of I in f.

- **phase 1** Try some randomly chosen assignments of non-cube variables until the proper assignments such that the corresponding superpoly p_I is non-constant are found.
- **phase 2** Set non-cube variables to the previous found assignments and calculate the superpoly under the real key, denoted by a. Then, only the values of key such that $p_I = a$ are reserved.
- **phase 3** Guess the remaining secret bits to recover the entire secret key.

Actually, the set of key bits that the superpoly depends on is a sub set of J.

The Division Property Based Cube Attacks

For a cube I, with the above proposition, we could obtain a set J of key bits, "involved" in the superpoly of I in f.

phase 1 Try some randomly chosen assignments of non-cube variables until the proper assignments such that the corresponding superpoly p_I is non-constant are found.

phase 2 Set non-cube variables to the previous found assignments and calculate the superpoly under the real key, denoted by a. Then, only the values of key such that $p_I = a$ are reserved.

phase 3 Guess the remaining secret bits to recover the entire secret key.

Actually, the set of key bits that the superpoly depends on is a sub set of J.

Revisiting Division Property Based Cube Attacks

A problem in division property based cube attacks

The key-recovery attacks may reduce to distinguishing attacks.

Solution

Computing the exact ANF of the superpoly of a given cube I.

Main Idea

Expressing z as a polynomial on the initial state $s^{(0)}$ iteratively and discard the terms where the superpoly of I is 0-constant in each iteration.

Revisiting Division Property Based Cube Attacks

A problem in division property based cube attacks

The key-recovery attacks may reduce to distinguishing attacks.

Solution

Computing the exact ANF of the superpoly of a given cube I.

Main Idea

Expressing z as a polynomial on the initial state $s^{(0)}$ iteratively and discard the terms where the superpoly of I is 0-constant in each iteration.

Revisiting Division Property Based Cube Attacks

A problem in division property based cube attacks

The key-recovery attacks may reduce to distinguishing attacks.

Solution

Computing the exact ANF of the superpoly of a given cube I.

Main Idea

Expressing z as a polynomial on the initial state $s^{(0)}$ iteratively and discard the terms where the superpoly of I is 0-constant in each iteration.

Two New Lemmas - I

Assuming z is expressed as $z = g_t(s^{(t)})$.

Lemma

Let I be a cube indies set. Let $u = \prod_{j=1}^{h} s_{i_j}^{(t)}$ be a term in $T(g_t)$. If the internal state $(s_1^{(t)}, s_2^{(t)}, \ldots, s_n^{(t)})$ does not have division property $\mathcal{D}^{1^n}_{(w_1,w_2,\ldots,w_n)}$ for each $(w_1, w_2, \ldots, w_n)$ such that $\prod_{i=1}^{n}(s_i^{(t)})^{w_i} | u$, then the superpoly of I in u is 0-constant.

An Invalid Term

For $u \in T(g_t)$, if the superpoly of I in u is 0-constant, then u is called an invalid term.

Two New Lemmas - I

Assuming z is expressed as $z = g_t(s^{(t)})$.

Lemma

Let I be a cube indies set. Let $u = \prod_{j=1}^{h} s_{i_j}^{(t)}$ be a term in $T(g_t)$. If the internal state $(s_1^{(t)}, s_2^{(t)}, \ldots, s_n^{(t)})$ does not have division property $\mathcal{D}_{(w_1,w_2,\ldots,w_n)}^{1^n}$ for each $(w_1, w_2, \ldots, w_n)$ such that $\prod_{i=1}^{n}(s_i^{(t)})^{w_i} | u$, then the superpoly of I in u is 0-constant.

An Invalid Term

For $u \in T(g_t)$, if the superpoly of I in u is 0-constant, then u is called an invalid term.

Two New Lemmas - II

Lemma

Let I be a cube set. Assume that the output bit z is presented as a polynomial on $s^{(t)}$, i.e., $z = g_t(s^{(t)})$. Then, according to Lemma 4, $g_t(s^{(t)})$ could be rewritten as $g_t(s^{(t)}) = g_t^1(s^{(t)}) \oplus g_t^2(s^{(t)})$, where each term $u \in T(g_t^2(s^{(t)}))$ is an invalid term for I. Then, the superpoly of I in $z = g_t(s^{(t)})$ is exactly the superpoly of I in $g_t^1(s^{(t)})$.

Accordingly, for a cube set I, g_t could be divide into two parts.

Main idea

Express the output z as a polynomial of the internal state, i.e. compute a polynomial g_t such that $z = g_t(s^{(t)})$.

- Discard invalid terms: the superpoly of I in an invalid term is 0-constant.
- A reduced polynomial g_t^1 could be obtained, where the superpoly of I in g_t is equal to that of I in g_t^1.
- Express g_t^1 as a polynomial on $s^{(t-n_t)}$, and repeat the above procedure.

When reaching the initial internal state $s^{(0)}$, the superpoly could be recovered according to the initialization way.

Main idea

Express the output z as a polynomial of the internal state, i.e. compute a polynomial g_t such that $z = g_t(s^{(t)})$.

- Discard invalid terms: the superpoly of I in an invalid term is 0-constant.
- A reduced polynomial g_t^1 could be obtained, where the superpoly of I in g_t is equal to that of I in g_t^1.
- Express g_t^1 as a polynomial on $s^{(t-n_t)}$, and repeat the above procedure.

When reaching the initial internal state $s^{(0)}$, the superpoly could be recovered according to the initialization way.

Main idea

Express the output z as a polynomial of the internal state, i.e. compute a polynomial g_t such that $z = g_t(s^{(t)})$.

- Discard invalid terms: the superpoly of I in an invalid term is 0-constant.
- A reduced polynomial g_t^1 could be obtained, where the superpoly of I in g_t is equal to that of I in g_t^1.
- Express g_t^1 as a polynomial on $s^{(t-n_t)}$, and repeat the above procedure.

When reaching the initial internal state $s^{(0)}$, the superpoly could be recovered according to the initialization way.

How to discard the invalid terms?

- Lemma 2: use MILP-aided division property to remove invalid terms
  - When the number of terms is large, the computing complexity is high.
- $\deg_I(u) < |I| \rightarrow u$ is an invalid term.
  - Use degree evaluation method based on numeric mapping to remove invalid terms.

Solution

First using the numeric mapping based method to discard invalid terms, then utilizing the MILP-aided method to discard invalid terms.

How to discard the invalid terms?

- Lemma 2: use MILP-aided division property to remove invalid terms
  - When the number of terms is large, the computing complexity is high.
- $\deg_I(u) < |I| \rightarrow u$ is an invalid term.
  - Use degree evaluation method based on numeric mapping to remove invalid terms.

Solution

First using the numeric mapping based method to discard invalid terms, then utilizing the MILP-aided method to discard invalid terms.

How to discard the invalid terms?

- Lemma 2: use MILP-aided division property to remove invalid terms
  - When the number of terms is large, the computing complexity is high.
- $\deg_I(u) < |I| \rightarrow u$ is an invalid term.
  - Use degree evaluation method based on numeric mapping to remove invalid terms.

Solution

First using the numeric mapping based method to discard invalid terms, then utilizing the MILP-aided method to discard invalid terms.

How to determine n_t?

- n_t is first set to 300
  - Only need to build MILP model tracing the propagation of division property through $r - 300$ rounds.
  - The scale of the MILP model could be reduced.
- set n_t such that $|T(g_{t-n_t})|$ is not very large.

See Algorithm 5 in our manuscript for details.

How to determine n_t?

- n_t is first set to 300
  - Only need to build MILP model tracing the propagation of division property through $r - 300$ rounds.
  - The scale of the MILP model could be reduced.
- set n_t such that $|T(g_{t-n_t})|$ is not very large.

See Algorithm 5 in our manuscript for details.

Trivium

Trivium: a bit oriented stream cipher designed by Cannière and Preneel.

- a Galois nonlinear feedback shift register with 3 quadratic feedback functions
- supports an 80-bit key and an 80-bit IV, 1152 initialization rounds
- one of the eSTREAM hardware-oriented finalists
- an International Standard under ISO/IEC 29192-3:2012

Experimental Verification I

Cube set: $I = \{v_1, v_{11}, v_{21}, v_{31}, v_{41}, v_{51}, v_{61}, v_{71}\}$.

- The superpoly of I in z_{591} is recovered.
- Different superpolies could be obtained by setting different values of non-cube variables.
  - $\boldsymbol{IV}$[1] =0x00000000000080040010,

$$p_I^{591}(\boldsymbol{x}, \boldsymbol{IV}) = x_{23}x_{24} \oplus x_{25} \oplus x_{67}.$$

  - $\boldsymbol{IV}$=0x00200000000020000040,

$$p_I^{591}(\boldsymbol{x}, \boldsymbol{IV}) = x_{66} \cdot (x_{23}x_{24} \oplus x_{25} \oplus x_{67} \oplus 1).$$

  - $\boldsymbol{IV}$=0x00000000000000000000,

$$p_I^{591}(\boldsymbol{x}, \boldsymbol{0}) = 0.$$

[1] $\boldsymbol{IV} = v_{80}||v_{79}|| \cdots ||v_1$

Experimental Verification II

Cube set: $I = \{v_1, v_{11}, v_{21}, v_{31}, v_{41}, v_{51}, v_{61}, v_{71}\}$.

- The superpoly of I in z_{586} is recovered.
- By appending some noncube variables to the set of cube variables, some simple superpolies could be obtained.

Table 2: New cubes and the corresponding superpolies

new cubes indies set	superpoly
$I \cup \{32, 37, 42, 50, 73\}$	x_{58}
$I \cup \{32, 37, 42, 49, 50, 70\}$	$x_{60} \oplus 1$
$I \cup \{32, 37, 50, 70, 73\}$	$x_{30} \oplus x_{55}x_{56} \oplus x_{57}$
$I \cup \{23, 24, 32, 42\}$	$x_{65}x_{66} \oplus x_{40} \oplus x_{67}$
$I \cup \{23, 24, 42\}$	$(x_{45}x_{46} \oplus x_{20} \oplus x_{47}) \cdot (x_{65}x_{66} \oplus x_{40} \oplus x_{67})$

Results on 832-round Trivium

For the cube used in [Todo17] to attack 832-round Trivium, we recover its superpoly which is given by

$$p_{I_1}(\boldsymbol{x}, \boldsymbol{v}) = v_{68}v_{78} \cdot (x_{58} \oplus v_{70}) \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61}). \quad (1)$$

- Under different assignments of non-cube variables, different equations could be obtained.
  - $p_{I_1}(\boldsymbol{x}, \boldsymbol{IV}) = x_{58} \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$, where $\boldsymbol{IV}$ =0x20080000000000000000
  - $p_{I_1}(\boldsymbol{x}, \boldsymbol{IV}) = (x_{58} \oplus 1) \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$, where $\boldsymbol{IV}$ =0x20280000000000000000

Results on 832-round Trivium

For the cube used in [Todo17] to attack 832-round Trivium, we recover its superpoly which is given by

$$p_{I_1}(\boldsymbol{x}, \boldsymbol{v}) = v_{68}v_{78} \cdot (x_{58} \oplus v_{70}) \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61}). \quad (1)$$

- Under different assignments of non-cube variables, different equations could be obtained.
  - $p_{I_1}(\boldsymbol{x}, \boldsymbol{IV}) = x_{58} \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$, where $\boldsymbol{IV}$ =0x20080000000000000000
  - $p_{I_1}(\boldsymbol{x}, \boldsymbol{IV}) = (x_{58} \oplus 1) \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$, where $\boldsymbol{IV}$ =0x20280000000000000000

A Key-recovery Attack on 832-round Trivium

With the two superpolies $x_{58} \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$, and $(x_{58} \oplus 1) \cdot (x_{59}x_{60} \oplus x_{34} \oplus x_{61})$.

- $x_{59}x_{60} \oplus x_{34} \oplus x_{61}$ could be recovered.
- x_{58} could be recovered when $x_{59}x_{60} \oplus x_{34} \oplus x_{61} = 1$.

For 832-round Trivium, the 80-bit key could be recovered in less than $2^{79} + 2^{73}$ requests.

Results on up to 839-round Trivium

For the cubes used to do "key-recovery" attacks against Trivium in [WHT+18], their superpolies are all 0-constant. Hence, such "key-recovery" attacks are all distinguishing attacks in fact.

Table 3: Results on Trivium variants with up to 839 rounds

Rounds	Cube	"Involved" Key Variables	Exact Superpoly
833	I_2	$x_{49}, x_{58}, x_{60}, x_{64}, x_{74}, x_{75}, x_{76}$ [WHT+18]	0-constant
833	I_3	x_{60} [WHT+18]	0-constant
835	I_4	x_{57} [WHT+18]	0-constant
836	I_5	x_{57} [WHT+18]	0-constant
839	I_6	x_{61} [WHT+18]	0-constant

$I_2 = \{1, 2, \ldots, 67, 69, 71, \ldots, 79\}$, $I_3 = \{1, 2, \ldots, 69, 71, 73, \ldots, 79\}$
$I_4 = \{1, 2, 3, 4, 6, 7, \ldots, 50, 52, 53, \ldots, 64, 66, 67, \ldots, 80\}$
$I_5 = \{1, \ldots, 11, 13, \ldots, 42, 44, \ldots, 80\}$
$I_6 = \{1, \ldots, 33, 35, \ldots, 46, 48, \ldots, 80\}$

Thanks for your attention

Questions?

Our Email: ye_chendong@126.com, tiantian_d@126.com