Exploring Core Monomial Prediction Further: Weak-Key Superpoly Recovery for 852-Round Trivium

Jiahui He[1,2,3,4], Kai Hu[1,2,3,4](✉), and Guowei Liu[1,2,3,4]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China.
hejiahui2020@mail.sdu.edu.cn, kai.hu@sdu.edu.cn, guoweiliu@mail.sdu.edu.cn
[2] State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao, China.
[3] Quancheng Laboratory, Jinan 250103, China.
[4] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China.

Abstract. The cube attack is one of the most powerful attacks on stream ciphers, with recovering the superpoly as its key step. The core monomial prediction is the state-of-the-art technique for superpoly recovery, which can reach 851 rounds for Trivium thus far (EUROCRYPT 2024). The core monomial prediction heavily relies on the trail enumeration which is the bottleneck for its efficiency.

This paper further explores the potential of the core monomial prediction for Trivium by constructing a composite representation for the superpoly. This representation allows us to detect the algebraic structure of the superpoly under specific conditions on the intermediate variables, without the computational burden of trail enumerations. Leveraging these discovered conditions, we successfully recovered weak-key superpolies for 852-round Trivium, establishing the first cryptanalytic result against 852-round Trivium in the literature to date.

Keywords: Cube Attack, Core Monomial Prediction, Superpoly Recovery, Trivium

1 Introduction

Cube attack, proposed by Dinur and Shamir at EUROCRYPT 2009 [14], is one of the general cryptanalytic techniques for analyzing symmetric-key cryptosystems. Since its proposal, the cube attack has been successfully applied to various stream ciphers [3,15,18,34,30,41], hash functions [13,25,28,36] and authenticated encryptions [29,16,4,32,33]. The fundamental principle of cube attacks lies in the algebraic manipulation of a cipher's output bit represented as a multivariate Boolean polynomial. By strategically partitioning input public variables into cube variables and non-cube variables, and then summing the output over a

structure called a *cube* that consists of all possible values of the cube variables, each monomial in the polynomial that does not involve all cube variables disappears due to the property of XOR operation. This cube summation process isolates specific monomials in the polynomial, yielding a reduced polynomial called the superpoly that depends solely on key bits and non-cube variables. When non-cube variables are fixed to constants, this results in an equation of the key bits in the online phase. By collecting multiple such equations from different cubes, an attacker can construct a system of equations in the online phase that, when solved, reveals critical information about the secret key.

The superpoly recovery and equation solving are two crucial steps in the cube attacks that dominate the effectiveness of the attack. At an early stage, traditional cube attacks are mainly concerned with linear or quadratic superpolies recovered by experimental tests, so the sizes of cubes are largely confined. Breakthroughs have been made by Todo et al. in [44], where they introduced the bit-based division property [45] into the cube attack.

As a generalization of the integral property [26], the original division property was proposed by Todo at EUROCRYPT 2015 [43]. With division property, the integral properties can be captured in a more accurate manner. As one prominent application of the division property, the long-standing cipher MISTY1 was broken theoretically for the first time [42]. The original division property can only be applied to word-oriented primitives. Boura and Canteaut at CRYPTO 2016 introduced the parity set as another approach for division properties, and managed to detect longer integral distinguishers for PRESENT [8]. At FSE 2016, bit-based division property [45] was proposed by Todo et al. to detect integral characteristics for bit-based ciphers. As a result, the previous experimentally discovered integral distinguishers of SIMON32 and Simeck32 were proved theoretically for the first time. By encoding the propagation of the bit-based division property through basic operations as linear constraints, the MILP model for deducing the bit-based division property was proposed by Xiang et al. [50] at ASIACRYPT 2016. This automated method allows for tracing the propagation of the bit-based division property through block ciphers with large block sizes, and thus has been applied to improve the integral attacks on many ciphers [38,37,19,47].

When determining the existence of a monomial in the superpoly, the bit-based division property may produce false positives. Therefore, after the bit-based division property was introduced into the cube attack, cryptographers began to focus on improving the precision and efficiency of division property in the superpoly recovery. Wang et al. advanced this research field in [48] by taking the cancellation of constant 1 bits into account to improve the precision of the bit-based division property. At ASIACRYPT 2019, Wang et al. [49] recovered the exact superpoly for the first time with the pruning technique combined with the three-subset division property, but this approach relies on the assumption that almost all elements in the 1-subset can be pruned. The inaccuracy problem was finally resolved by Hao et al. in [20], where the three-subset division property without unknown subset was introduced. From a purely algebraic

perspective, the division property was interpreted as the monomial prediction technique by Hu et al. at ASIACRYPT 2020 [24]. Combined with the divide-and-conquer strategy, this technique was applied in a nested manner [23] to recover massive superpolies for stream ciphers. In order to alleviate the trail enumeration efficiency bottleneck inherent in the monomial prediction technique, He et al. proposed the core monomial prediction technique [22] at ASIACRYPT 2022 by sacrificing the accuracy of monomial prediction for efficiency. The perfect accuracy of core monomial prediction was proved by He at al. in [21], where a recursive framework for superpoly recovery was established based solely on the core monomial prediction technique. This framework extended the superpoly recovery for Trivium to the current 851 rounds, with a time cost of about 25 days. It is worth mentioning that Beyne and Verbauwhede have unified the division property, parity set and monomial prediction in the algebraic transition matrix [7] as an application of Beyne's geometric approach [5,6] to integral cryptanalysis.

Motivation. When applied to superpoly recovery, both the monomial prediction and core monomial prediction theories can achieve perfect accuracy by enumerating trails. Beyond superpoly recovery, monomial prediction has demonstrated utility in detecting algebraic properties of ciphers (e.g., algebraic degree [24,32]), while the potential value of core monomial prediction beyond superpoly recovery remains largely unexplored. Compared to monomial prediction, core monomial prediction tracks more detailed information about the structure of a cipher throughout the propagation process. This suggests that core monomial prediction should, intuitively, be capable of revealing algebraic properties of a cipher with greater precision and depth. The algebraic properties of a superpoly, as one of the most critical algebraic properties of a cipher, fundamentally determines the success of a cube attack. Therefore, exploring the relationship between the core monomial prediction and the algebraic properties of a superpoly is essential for advancing the cube attack.

Our Contributions. This paper aims to advance the cube attack by establishing the theoretical framework for detecting the algebraic properties of a superpoly based on the core monomial prediction. The key contributions are:

- We introduce a composite representation for the superpoly, where the intermediate variables are naturally derived from the structure of a cipher. Since this composite representation provides a more tractable and efficient perspective for analyzing the superpoly compared to treating the superpoly as a complex polynomial of key bits, we can detect potential simplifications in the algebraic structure of the superpoly under conditions on the intermediate variables, without the computational burden of enumerating trails.
- Based on the composite representation, we develop a greedy algorithm to search for optimal conditions that maximize the superpoly simplification. For the case when the concrete expression of the superpoly is not available, our algorithm employs a MILP-based method to efficiently estimate the effectiveness of each potential condition, which is solely based on the propagation

of core monomial prediction rather than the enumeration of core monomial trails. These identified conditions can be transformed into conditions on the key bits and non-cube variables, under which the superpoly can be recovered efficiently using a recursive framework similar to that in [21].

As a demonstration of the practical effectiveness of our approach, we successfully recover two weak-key superpolies for 852-round TRIVIUM under 2-bit and 3-bit conditions on the key bits, respectively. Utilizing these superpolies under the implementation-dependent setting, we are able to derive a weak-key cube attack against 852-round TRIVIUM on a key space of 2^{78} with a complexity of approximately 2^{77}, establishing the first cryptanalytic result on 852-round TRIVIUM in the literature.

The summary of our cube attack results is provided in Table 1. For verification and reproducibility purposes, we have made available the source code for optimal condition identification and superpoly recovery, along with the complete recovered weak-key superpolies, in our anonymous git repository

https://anonymous.4open.science/r/KSOPxz-sDcZ91aC0C3.

2 Background

In this section, we introduce some notations that will be used later and briefly review the basics of the core monomial prediction technique. The size of a set S is denoted as $|S|$. We use italic Latin or Greek letters to represent binary vectors, where each element can be either a binary value or a binary variable, depending on the context. The all-zeros vector and all-ones vector are denoted as $\mathbf{0}$ and $\mathbf{1}$, respectively, with their sizes inferred from the context. For a binary vector $x \in \mathbb{F}_2^n$, we use $x[i]$ to represent its i-th element, such that $x = (x[0], \ldots, x[n-1])$. For a set $I = \{i_0, \ldots, i_{|I|-1}\} \subseteq \{0, \ldots, n-1\}$, we use $x[I] = (x[i_0], \ldots, x[i_{|I|-1}]) \in \mathbb{F}_2^{|I|}$ to represent the sub-vector selected by I from $x \in \mathbb{F}_2^n$, and thus $x[I][j] = x[i_j]$ for $0 \leq j < |I|$. The Hamming weight of x is denoted by $\mathrm{Hw}(x)$. For two binary vectors $x, u \in \mathbb{F}_2^n$, we refer to the product $\pi(x, u) = \prod_{i=0}^{n-1} x[i]^{u[i]}$ as a *monomial*; we write $x \preceq u$ (resp. $x \succeq u$) if $x[i] \leq u[i]$ (resp. $x[i] \geq u[i]$) holds for all $i \in \{0, \ldots, n-1\}$; the concatenation of x and u is denoted by $x \| u$.

For a Boolean polynomial p of $x \in \mathbb{F}_2^n$, we denote the algebraic degree as $\mathrm{Deg}(p)$. Particularly, $\mathrm{Deg}(0) = -\infty$ and $\mathrm{Deg}(1) = 0$. For a monomial $\pi(x, u)$, we express p as $p = g(x) \cdot \pi(x, u) + h(x)$, such that $g(x)$ only contains variables $x[i]$ where $u[i] = 0$, and each monomial in $h(x)$ is not divisible by $\pi(x, u)$. We refer to g and h as the quotient and remainder of p with respect to $\pi(x, u)$, denoted as $p/\pi(x, u)$ and $p \bmod \pi(x, u)$, respectively. For a set $S = \{\pi(x, t_0), \ldots, \pi(x, t_{|S|-1})\}$ of monomials, we use $p \bmod \langle S \rangle$ to represent the remainder when dividing p by the ideal $\langle \pi(x, t_0), \ldots, \pi(x, t_{|S|-1}) \rangle$, which consists of all monomials in p that are not divisible by any monomial in S. When $S = \emptyset$, we assume $p \bmod \langle S \rangle = p$ and $p/\langle S \rangle = 0$.

Table 1: Summary of the cube attacks against TRIVIUM

Rounds	#Cube*	Cube size	Attack type	Data	Time	Key Space	Ref.
672	63	12	Cube	$2^{18.6}$	2^{17}	2^{80}	[14]
709	80	22-23	Cube	2^{23}	$2^{29.14}$	2^{80}	[31]
767	35	28-31	Cube	2^{31}	2^{45}	2^{80}	[14]
784	42	30-33	Cube	2^{33}	2^{39}	2^{80}	[18]
799	18	32-37	Cube	2^{38}	2^{62}	2^{80}	[18]
802	8	34-37	Cube	2^{37}	2^{72}	2^{80}	[51]
805	42	32-38	Cube	2^{38}	$2^{41.4}$	2^{80}	[52]
805	28	28	Correlation Cube	2^{28}	2^{73}	2^{80}	[30]
806	16	34-37	Cube	$2^{38.64}$	2^{64}	2^{80}	[52]
806	29	34-37	Cube	2^{39}	2^{39}	2^{80}	[40]
808	37	39-41	Cube	2^{44}	$2^{44.58}$	2^{80}	[40]
810	39	40-42	Cube	2^{44}	$2^{44.17}$	2^{80}	[27]
815	35	44-46	Cube	2^{47}	$2^{47.32}$	2^{80}	[11]
820	30	48-51	Cube	2^{53}	$2^{53.17}$	2^{80}	[11]
820	2^{13}	38	Correlation Cube	2^{51}	2^{60}	$2^{79.8}$	[46]
825	31	49-52	Cube	2^{53}	$2^{53.09}$	2^{80}	[27]
825	2^{12}	41	Correlation Cube	2^{53}	2^{60}	$2^{79.7}$	[46]
830	2^{13}	41	Correlation Cube	2^{54}	2^{60}	$2^{79.3}$	[46]
832	1	72	Cube	2^{72}	2^{79}	2^{80}	[44,49]
835	41	35	Correlation Cube	2^{35}	2^{75}	2^{80}	[30]
840	1	78	Cube	2^{78}	$2^{79.6}$	2^{80}	[20]
840	3	75	Cube	$2^{76.6}$	$2^{77.8}$	2^{80}	[24]
840	6	47-62	Cube	2^{62}	$2^{76.32}$	2^{80}	[23]
841	1	78	Cube	2^{78}	$2^{79.6}$	2^{80}	[20]
841	2	76	Cube	2^{77}	$2^{78.6}$	2^{80}	[24]
841	3	56-76	Cube	2^{76}	2^{78}	2^{80}	[23]
842	1	78	Cube	2^{78}	$2^{79.6}$	2^{80}	[20]
842	2	76	Cube	2^{77}	$2^{78.6}$	2^{80}	[24]
842	3	56-76	Cube	2^{76}	2^{78}	2^{80}	[23]
843	2	78	Cube	2^{78}	$2^{79.6}$	2^{80}	[40]
843*	5	56-76	Cube	2^{56}	2^{77}	2^{80}	[23]
844	28	37-38	Cube	$2^{42.8}$	2^{76}	2^{80}	[12]
844*	2	54-55	Cube	2^{56}	2^{78}	2^{80}	[23]
845*	2	54-55	Cube	2^{56}	2^{78}	2^{80}	[23]
846*	6	51-54	Cube	2^{51}	2^{79}	2^{80}	[22]
847*	2	52-53	Cube	2^{52}	2^{79}	2^{80}	[22]
848*	1	52	Cube	2^{52}	2^{79}	2^{80}	[22]
849*	2	44	Cube	2^{44}	2^{79}	2^{80}	[21]
850*	1	44	Cube	2^{44}	2^{79}	2^{80}	[21]
851*	1	44	Cube	2^{44}	2^{79}	2^{80}	[21]
852*	**1**	**43**	**Weak-Key Cube**	$\mathbf{2^{43}}$	$\mathbf{2^{76}}$	$\mathbf{2^{77}}$	**Sect. 4**
852*	**1**	**44**	**Weak-Key Cube**	$\mathbf{2^{44}}$	$\mathbf{2^{77}}$	$\mathbf{2^{78}}$	**Sect. 4**

* #Cube represents the number of cubes whose superpolies are recovered, but this may not be equal to the number of cubes eventually used in the key recovery attack.
* All of these cube attacks are performed under the implementation-dependent setting, where a TRIVIUM call is translated into a specific number of bit operations.

2.1 Möbius Transform

For a Boolean function $F(x)$ of n variables, we can represent its ANF as a binary vector of 2^n entries corresponding to its 2^n coefficients. The truth table of F, which is also represented as a binary vector of 2^n entries, can be computed from the ANF of F via the Möbius transform over $\mathbb{F}_2^n$, which is based on the following formula:

$$F(x[0], \ldots, x[n-1]) = x[0] \cdot F(x[1], \ldots, x[n-1]) + F_1(x[1], \ldots, x[n-1]).$$

Thus, we can compute the truth table of F by first computing the truth tables of F_0 and F_1 and then performing 2^{n-1} bit operations. Denoting the time complexity of the Möbius transform as $T(n)$, we have $T(n) = 2T(n-1) + 2^{n-1}$, thus $T(n) \leq n \cdot 2^n$. In this paper, we assume the Möbius transform requires $n \cdot 2^n$ bit operations and 2^n bits of memory.

2.2 Cube Attack

The concept of cube attack was proposed by Dinur and Shamir at EUROCRYPT 2009 [14] as an extension of the higher-order differential attack.

For a cipher that takes secret variables $k \in \mathbb{F}_2^n$ (the key) and public variables $v \in \mathbb{F}_2^m$ (the plaintext/IV) as input, the first output keystream bit can be expressed as a Boolean polynomial f of $k \| v$. Let $I \subseteq \{0, \ldots, m-1\}$ and $J \subseteq \{0, \ldots, m-1\}$ be two disjoint subsets corresponding to the selection of public variables, then we can algebraically represent f as

$$f(k\|v) = f/t_I \cdot t_I + f \bmod t_I, \tag{1}$$

where $t_I = \prod_{i \in I} v[i]$. The set I and the public variables $v[I]$ are referred to as the cube indices and cube variables, respectively.

Assigning zero values to $v[J]$, we can define a structure called cube, denoted as $C_I(J)$, consisting of $2^{|I|}$ binary vectors as follows:

$$C_I(J) = \{w \in \mathbb{F}_2^m : w[s] = 0/1 \text{ for } s \in I \bigwedge w[s] = 0 \text{ for } s \in J \bigwedge w[s] \text{ is the binary variable } v[s] \text{ for } s \notin (I \cup J)\}.$$

By substituting $v[J]$ with $\mathbf{0}$, we can derive a polynomial $p(k\|v)$ from f/t_I, which only relates to k and $v[\{0, \ldots, m-1\} \setminus (I \cup J)]$. We refer to this polynomial as the superpoly of the cube $C_I(J)$ in f, which can be computed by assuming f over the cube $C_I(J)$ as follows:

$$p(k\|v) = \sum_{w \in C_I(J)} f(k\|w). \tag{2}$$

When $I \cup J = \{0, \ldots, m-1\}$, the cube $C_I(J)$ is uniquely determined by I, in which case we may also refer to I as the "cube" without causing ambiguities. The size of I is also referred to as the dimension of the cube.

The whole process of a cube attack can be summarized as follows:

1. **Offline Phase: Superpoly Recovery.** Select the cube indices I and the set J. Recover the expression of the superpoly $p(k\|v)$.
2. **Online Phase: Parity Key Recovery.** Assign a specific value $\alpha \in \mathbb{F}_2^{m-|I|-|J|}$ to $v[\{0,\ldots,m-1\} \setminus (I \cup J)]$. Assume that the secret key is K. For each $w \in C_I(J)$, define $\beta \in \mathbb{F}_2^m$ as $\beta[I \cup J] = w[I \cup J] \bigwedge \beta[\{0,\ldots,m-1\} \setminus (I \cup J)] = \alpha$, and then query the value $f(K\|\beta)$ to the encryption oracle. Summing these values yields the value of the superpoly, which establishes an equation involving the secret key bits. Solving this equation provides partial information about the secret key.
3. **Brute Force.** Guess the remaining secret key bits to recover the full secret key.

A critical prerequisite for the successful implementation of a cube attack is the accurate recovery of the superpoly expression.

2.3 Core Monomial Prediction

Typically, the first output keystream bit is generated from k and v through multiple transformations. Thus, we can always represent the Boolean polynomial f in Eq. (1) as the composition of R vectorial Boolean functions, namely

$$f(k\|v) = f^{R-1} \circ \cdots \circ f^0(k\|v),$$

where each component function $f^i, 0 \le i < R$ is simple and its ANF can be easily computed (e.g., each f^i is a basic operation COPY, AND and XOR). For each i, where $0 \le i < R$, let x^i and x^{i+1} denote the input and output variables of f^i, respectively. By construction, $x^0 = k\|v$ and $x^R = (x^R[0])$, where $x^R[0]$ is the output keystream bit.

Flag Technique. For each binary variable in x^i, where $0 \le i \le R$, we assign a flag $y \in \{\delta, 1_c, 0_c\}$. As a result, all binary variables are categorized according to their assigned flags: those flagged with δ are referred to as δ-variables, those with 1_c as 1_c-variables, and those with 0_c as 0_c-variables.

Three operations $=, \times$ and $\oplus$ are defined for the flags, corresponding to the basic operations COPY, AND and XOR, respectively. The operation $=$ copies one flag to another, while $\times$ and $\oplus$ satisfy the following rules:

$$\begin{array}{lll} 1_c \times y = y \times 1_c = y & 1_c \oplus 1_c = 1_c & 0_c \oplus y = y \oplus 0_c = y \\ 0_c \times y = y \times 0_c = 0_c & \delta \oplus y = y \oplus \delta = \delta & \delta \times \delta = \delta \end{array},$$

where y can be any of $\{\delta, 1_c, 0_c\}$. In this paper, the flags are fixed as follows: initially, based on the cube $C_I(J)$, the flags of x^0 are determined by assigning δ flags to the cube variables, 1_c flags to k and $v[\{0,\ldots,m-1\} \setminus (I \cup J)]$, and 0_c flags to $v[J]$; then, the flags of x^i are computed from the flags of x^0 according to the operation rules of flags for $i > 0$.

Core Monomial Prediction. For $0 \le i \le R$, if a monomial $\pi(x^i, u^i)$ only relates to δ-variables, then we call it a *core monomial*. Now consider two core

monomials $\pi(x^{r_0}, t^{r_0})$ and $\pi(x^{r_1}, t^{r_1})$ where $0 \leq r_0 < r_1 \leq R$. We can express $\pi(x^{r_1}, t^{r_1})$ as a Boolean polynomial h of x^{r_0}. Let S be the set consisting of all δ-variables and 0_c-variables in x^{r_0}. If $(h/\pi(x^{r_0}, t^{r_0})) \bmod \langle S \rangle \neq 0$, then we say that t^{r_0} can propagate to t^{r_1} through $f^{r_1-1} \circ \cdots \circ f^{r_0}$ with a coefficient $(h/\pi(x^{r_0}, t^{r_0})) \bmod \langle S \rangle$. For brevity, we sometimes use the compact notation $\mathbb{C}_{t^{r_0}, t^{r_1}}(x^{r_0})$ to represent the coefficient $(h/\pi(x^{r_0}, t^{r_0})) \bmod \langle S \rangle$. Note that the notation $\mathbb{C}_{t^{r_0}, t^{r_1}}(x^{r_0})$ implicitly contains the condition that both $\pi(x^{r_0}, t^{r_0})$ and $\pi(x^{r_1}, t^{r_1})$ are core monomials.

We call the sequence $(t^{r_0}, t^{r_0+1}, \ldots, t^{r_1})$ a core monomial trail if $\mathbb{C}_{t^s, t^{s+1}}(x^s) \neq 0$ for all $s \in \{r_0, \ldots, r_1 - 1\}$. The contribution of this trail is defined as the polynomial of x^0 generated from the product of coefficients along this trail, namely $\prod_{s:r_0 \leq s < r_1} \mathbb{C}_{t^s, t^{s+1}}(f^{s-1} \circ \cdots \circ f^0(x^0))$, denoted as $\mathbb{C}_{(t^{r_0}, t^{r_0+1}, \ldots, t^{r_1})}(x^0)$, which is a polynomial of x^0. The propagation rule of core monomial prediction through a component function $f^i, 0 \leq i < R$ can be generally constructed as follows:

1. Initialize a set S.
2. For each t^i and t^{i+1} that satisfy $\mathbb{C}_{t^i, t^{i+1}}(x^i) \neq 0$, put the vector $t^i \| t^{i+1}$ into S.
3. With the help of mathematical tools (e.g., Sagemath), follow the methods in [35,39,9] to generate a set of linear constraints, such that the feasible solutions from these contraints are exactly the vector in S.

The concrete propagation rules of core monomial prediction through basic operations COPY, AND and XOR are provided in Appendix A. The following theorems illustrates how to recover the superpoly by enumerating core monomial trails.

Theorem 1 (Superpoly Recovery by Core Monomial Prediction [21]). *Recovering the superpoly $p(k\|v)$ in Eq. (2) can be reduced to computing $\mathbb{C}_{u^0, u^R}(x^0)$, where u^0 satisfies $u^0[i] = 1$ for all $i - n \in I$ and $u^0[i] = 1$ otherwise, and $u^R = (1) \in \mathbb{F}_2^1$ corresponds to x^R.*

In the rest of this paper, we always refer to $\mathbb{C}_{u^0, u^R}(x^0)$ defined in the above theorem as the superpoly of the cube $C_I(J)$.

Theorem 2 (Divide-and-Conquer Strategy by Core Monomial Prediction [21]). *For two core monomials $\pi(x^{r_0}, t^{r_0})$ and $\pi(x^{r_1}, t^{r_1})$, where $0 \leq r_0 < r_1 \leq R$, we can decompose $\mathbb{C}_{t^{r_0}, t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ according to the following formula:*

$$
\begin{aligned}
&\mathbb{C}_{t^{r_0}, t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0)) \\
&= \sum_{\substack{t^{r_0+1}: \\ \mathbb{C}_{t^{r_0}, t^{r_0+1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0)) \neq 0}} \left(\mathbb{C}_{t^{r_0}, t^{r_0+1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0)) \cdot \mathbb{C}_{t^{r_0+1}, t^{r_1}}(f^{r_0} \circ \cdots \circ f^0(x^0))\right) \\
&= \sum_{\substack{t^{r_1-1}: \\ \mathbb{C}_{t^{r_1-1}, t^{r_1}}(f^{r_1-2} \circ \cdots \circ f^0(x^0)) \neq 0}} \left(\mathbb{C}_{t^{r_0}, t^{r_1-1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0)) \cdot \mathbb{C}_{t^{r_1-1}, t^{r_1}}(f^{r_1-2} \circ \cdots \circ f^0(x^0))\right),
\end{aligned}
$$

where $\mathbb{C}_{t^{r_0}, t^{r_0+1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ in the first summation and $\mathbb{C}_{t^{r_1-1}, t^{r_1}}(f^{r_1-2} \circ \cdots \circ f^0(x^0))$ in the second summation can be precomputed.

Theorem 3 (Coefficient Computation by Core Monomial Prediction [21]). *For two core monomials $\pi(x^{r_0}, t^{r_0})$ and $\pi(x^{r_1}, t^{r_1})$, where $0 \leq r_0 < r_1 \leq R$, if $r_1 = r_0 + 1$, then $\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ can be obtained immediately since the ANF of f^{r_0} can be easily computed; otherwise, we can compute $\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ as*

$$\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0)) = \sum_{\substack{t^{r_0+1},\ldots,t^{r_1-1}:\\ \forall s\in\{r_0,\ldots,r_1-1\},\mathbb{C}_{t^s,t^{s+1}}(x^s)\neq 0}} \mathbb{C}_{(t^{r_0},t^{r_0+1},\ldots,t^{r_1})}(x^0).$$

When implementing Theorem 3 in practice, we first construct a model to describe the propagation of core monomial prediction from t^{r_0} to t^{r_1}. By solving the model, we enumerate all core monomial trails from t^{r_0} and t^{r_1} and compute the contribution of each trail. Finally, $\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ is obtained by summing the contributions of all trails. The time cost of this process mainly comes from the model solving.

On the other hand, it was pointed out in [21] that for any core monomials $\pi(x^s, t^s)$ and $\pi(x^{s+1}, t^{s+1})$, where $0 \leq s < R$, the coefficient $\mathbb{C}_{t^s,t^{s+1}}(x^s)$ can only assume a finite number of possible expressions, according to the following rules.

Rule 1 (COPY [21]) *If f^s is the basic operation COPY, then for any core monomials $\pi(x^s, t^s)$ and $\pi(x^{s+1}, t^{s+1})$, $\mathbb{C}_{t^s,t^{s+1}}(x^s)$ must be 1 if $\mathbb{C}_{t^s,t^{s+1}}(x^s) \neq 0$.*

Rule 2 (AND [21]) *If f^s is the basic operation AND, where $x^{s+1}[0] = \prod_{j=0}^{n-1} x^s[j]$ and $x^{s+1}[j] = x^s[j+n-1]$ for $j > 0$. Then, for any core monomials $\pi(x^s, t^s)$ and $\pi(x^{s+1}, t^{s+1})$, if $\mathbb{C}_{t^s,t^{s+1}}(x^s) \neq 0$, then it can only be 1 or $\prod_{x^s[j] \text{ is a } 1_c\text{-variable}, j:0\leq j<n,} x^s[j]$.*

Rule 3 (XOR [21]) *If f^s is the basic operation XOR, where $x^{s+1}[0] = \bigoplus_{j=0}^{n-1} x^s[j]$ and $x^{s+1}[j] = x^s[j+n-1]$ for $j > 0$. Then, for any core monomials $\pi(x^s, t^s)$ and $\pi(x^{s+1}, t^{s+1})$, if $\mathbb{C}_{t^s,t^{s+1}}(x^s) \neq 0$, then it can only be 1 or $\bigoplus_{x^s[j] \text{ is a } 1_c\text{-variable}, j:0\leq j<n,} x^s[j]$.*

Rule 4 (S-box [21]) *If f^s is an n-bit S-box, then all the possible expressions that a coefficient $\mathbb{C}_{t^s,t^{s+1}}(x^s)$ can take can be precomputed by a $2^n \times 2^n$ table, where the entry at (u, w) is $\mathbb{C}_{u,w}(x^s)$ if both $\pi(x^s, u)$ and $\pi(x^{s+1}, w)$ are core monomials; otherwise it is 0.*

In [21], the superpoly recovery process employs a recursive framework. Initially, leveraging Theorem 1, the superpoly recovery is reduced into a coefficient computation problem. The framework then decomposes any unresolved problems into smaller, more tractable sub-problems according to Theorem 2. These sub-problems are efficiently addressed using Theorem 3 within a preset time limit. Any remaining unresolved sub-problems will be further decomposed and addressed, with this iterative process continuing until all problems are completely resolved. The details of this recursive framework is provided in Appendix B. The overall time cost of the framework is primarily determined by the repeated applications of Theorem 3, as this is the only step involving model solving.

As pointed out in [21], when i is sufficiently large, all state variables of x^i become δ-variables, in which case the propagation of core monomial prediction through $f^j, j \geq i$ becomes exactly equivalent to the propagation of monomial prediction proposed in [24]. Thus, the monomial prediction can be considered as a special case of the core monomial prediction. Naturally, the core monomial prediction inherits the ability of monomial prediction in detecting the algebraic properties of superpolies. Since the core monomial prediction tracks additional information related to the structure of a cipher (e.g., the flags of state bits) compared to the monomial prediction, a compelling research question emerges: can the core monomial prediction detect algebraic properties of a superpoly that remain invisible to the monomial prediction technique?

3 Conditional Superpoly

In this section, we introduce a composite representation for the superpoly based on the core monomial prediction technique. This composite representation can be used to simplify the superpoly by imposing conditions on the intermediate variables. To search for optimal conditions that maximize the superpoly simplification, we give a MILP-based method to efficiently detect the algebraic degree of the composite representation without requiring enumeration of core monomial trails. This specifies the role of core monomial prediction in detecting the algebraic structure of the superpoly.

3.1 Composite Representation for Superpoly

As mentioned earlier, for any core monomials $\pi(x^s, t^s)$ and $\pi(x^{s+1}, t^{s+1})$ where $0 \leq s < R$, there are only a limited number of possible expressions that $\mathbb{C}_{t^s,t^{s+1}}(x^s)$ can take. Consequently, the same holds for $\mathbb{C}_{t^s,t^{s+1}}(f^{s-1} \circ \cdots \circ f^0(x^0))$. For each f^s, where $0 \leq s < R$, define the set V^s as

$$V^s = \bigcup_{\substack{t^s,t^{s+1}:\\ \mathbb{C}_{t^s,t^{s+1}}(f^{s-1}\circ\cdots\circ f^0(x^0))\neq 0/1}} \{\mathbb{C}_{t^s,t^{s+1}}(f^{s-1} \circ \cdots \circ f^0(x^0))\}.$$

Once the cube $C_I(J)$ is selected, the set $\bigcup_{0\leq s<R} V^s$ can be determined immediately. Using the polynomials in this set, we can construct a composite representation for the contribution of each core monomial trail, which ultimately leads to a composite representation for the superpoly.

Proposition 1 (Composite Representation for Contribution of Core Monomial Trail). *Assuming that $\bigcup_{0\leq s<R} V^s = \{g_0, \ldots, g_{N-1}\}$, where each $g_j, 0 \leq j < N$ is a polynomial of x^0 (more precisely, k and the non-zero non-cube variables $v[\{0, \ldots, m-1\} \setminus (I \cup J)]$), we introduce N intermediate variables $Y \in \mathbb{F}_2^N$, build a map $Y = g(x^0) = (g_0(x^0), \ldots, g_{N-1}(x^0))$ between x^0 and Y, such that $Y[j] = g_j(x^0)$ for $0 \leq j < N$, and build a table T such that $T[g_j] = Y[j]$ for each $g_j, 0 \leq j < N$, $T[1] = 1$ and $T[0] = 0$. Then, for two*

core monomials $\pi(x^{r_0}, t^{r_0})$ *and* $\pi(x^{r_1}, t^{r_1})$*, where* $0 \leq r_0 < r_1 \leq R$*, the contribution* $\mathbb{C}_{(t^{r_0},\ldots,t^{r_1})}(x^0)$ *of a core monomial trail* $(t^{r_0}, \ldots, t^{r_1})$ *can be represented as* $h(g(x^0))$ *with* h *being a polynomial of* Y *defined as*

$$h(Y) = \prod_{s:r_0 \leq s < r_1} T[\mathbb{C}_{t^s,t^{s+1}}(f^{s-1} \circ \cdots \circ f^0(x^0))].$$

The polynomial $h(Y)$ *is referred to as the composite representation for the contribution* $\mathbb{C}_{(t^{r_0},\ldots,t^{r_1})}(x^0)$*, denoted as* $\widehat{\mathbb{C}}_{(t^{r_0},\ldots,t^{r_1})}(Y)$*.*

Proof. This follows directly from the definition of the contribution of a core monomial trail. □

Proposition 2 (Composite Representation for Coefficient). *Let the intermediate variables* Y *and the map* g *be defined as in Proposition 1. For two core monomials* $\pi(x^{r_0}, t^{r_0})$ *and* $\pi(x^{r_1}, t^{r_1})$ *where* $0 \leq r_0 < r_1 \leq R$*, the polynomial* $\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots f^0(x^0))$ *can be represented as* $h(g(x^0))$ *with* h *being defined as*

$$\begin{aligned} h(Y) &= \sum_{\substack{t^{r_0+1},\ldots,t^{r_1-1}: \\ \forall s \in \{r_0,\ldots,r_1-1\}, \mathbb{C}_{t^s,t^{s+1}}(x^s) \neq 0}} \widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1},\ldots,t^{r_1})}(Y), \\ &= \sum_{\substack{t^{r_0+1},\ldots,t^{r_1-1}: \\ \forall s \in \{r_0,\ldots,r_1-1\}, \mathbb{C}_{t^s,t^{s+1}}(f^{s-1} \circ \cdots \circ f^0(x^0)) \neq 0}} \widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1},\ldots,t^{r_1})}(Y) \end{aligned}$$

$h(Y)$ *is referred to as the composite representation for* $\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ *and is denoted as* $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$*.*

Proof. By replacing each core monomial trail appearing in Theorem 3 with its composite representation given in Proposition 1, we obtain this proposition. □

In practice, Proposition 2 can be implemented in a similar way to Theorem 3. The only difference is that, after enumerating all core monomial trails from t^{r_0} and t^{r_1} by solving the model, the composite representation for the contribution of each trail is computed. Thus, compared to implementing Theorem 3, the time cost of implementing Proposition 2 remains unchanged compared to Theorem 3, as the model-solving step remains the same.

Under the composite representation, the divide-and-conquer strategy in Theorem 2 remains applicable, as demonstrated by the following proposition.

Proposition 3 (Divide-and-Conquer Strategy under Composite Representation). *For two core monomials* $\pi(x^{r_0}, t^{r_0})$ *and* $\pi(x^{r_1}, t^{r_1})$*, where* $0 \leq r_0 < r_1 \leq R$*, we can decompose* $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ *according to the following formula:*

$$\begin{aligned} \widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y) &= \sum_{\substack{t^{r_0+1}: \\ \widehat{\mathbb{C}}_{t^{r_0},t^{r_0+1}}(Y) \neq 0}} \left(\widehat{\mathbb{C}}_{t^{r_0},t^{r_0+1}}(Y) \cdot \widehat{\mathbb{C}}_{t^{r_0+1},t^{r_1}}(Y)\right) \\ &= \sum_{\substack{t^{r_1-1}: \\ \widehat{\mathbb{C}}_{t^{r_1-1},t^{r_1}}(Y) \neq 0}} \left(\widehat{\mathbb{C}}_{t^{r_0},t^{r_1-1}}(Y) \cdot \widehat{\mathbb{C}}_{t^{r_1-1},t^{r_1}}(Y)\right), \end{aligned}$$

where $\widehat{\mathbb{C}}_{t^{r_0},t^{r_0+1}}(Y)$ *in the first summation and* $\widehat{\mathbb{C}}_{t^{r_1-1},t^{r_1}}(Y)$ *in the second summation can be precomputed.*

Proof. By replacing the coefficients in Theorem 2 with their composite representations, we obtain this proposition. □

For the superpoly $\mathbb{C}_{u^0,u^R}(x^0)$ defined in Theorem 1, we can immediately obtain a composite representation $\widehat{\mathbb{C}}_{u^0,u^R}(Y)$ for it according to Proposition 2. To illustrate this fact more clearly, we provide a simple example below.

Example 1. Consider $f = f^1 \circ f^0$ with secret variables $k \in \mathbb{F}_2^2$, public variables $v \in \mathbb{F}_2^5$, cube indices $I = \{0, 1\}$ ($v[0]$ and $v[1]$ are cube variables) and set $J = \{2\}$ ($v[2]$ is set to 0). The initial state is $x^0 = k \| v = (k[0], k[1], v[0], v[1], v[2], v[3], v[4])$, thus $k[0], k[1], v[3]$ and $v[4]$ are 1_c-variables, $v[0]$ and $v[1]$ are δ-variables, and $v[2]$ is a 0_c-variable. The first component function $f^0 : \mathbb{F}_2^6 \to \mathbb{F}_2^4$ is defined as

$$
\begin{aligned}
x^1[0] &= x^0[0] + x^0[2]x^0[4] = k[0] + v[0]v[2],\\
x^1[1] &= x^0[1] + x^0[3]x^0[6] = k[1] + v[1]v[4],\\
x^1[2] &= x^0[2]x^0[3] + x^0[3] + x^0[4]x^0[6] = v[0]v[1] + v[1] + v[2]v[4],\\
x^1[3] &= x^0[0] + x^0[1] = k[0] + k[1],
\end{aligned}
$$

The second component function $f^1 : \mathbb{F}_2^4 \to \mathbb{F}_2^1$ is defined as

$$x^2[0] = x^1[0] + x^1[0]x^1[1]x^1[2] + x^1[2]x^1[3].$$

According to the operation rules of flags, we can derive that $x^1[1], x^1[2]$ and $x^2[0]$ are δ-variables, $x^1[0]$ and $x^1[3]$ are 1_c-variables.

From $u^0 = (0, 0, 1, 1, 0, 0, 0)$ to $u^2 = (1)$ there is two core monomial trails $(u^0, (0, 0, 1, 0), u^2)$ and $(u^0, (0, 1, 1, 0), u^2)$, and we can compute $\mathbb{C}_{(u^0,(0,0,1,0),u^2)} = 1 \cdot (k[0] + k[1])$ and $\mathbb{C}_{(u^0,(0,1,1,0),u^2)} = k[0] \cdot (k[1] + v[4])$. Summing these contributions gives the superpoly $k[0] + k[1] + k[0] \cdot (k[1] + v[4])$.

For f^0, we have

$$x^1[1]x^1[2] = (k[1] + v[4]) \cdot v[1] + (k[1] + v[4]) \cdot v[0]v[1],$$

thus $V^0 = \{k[1] + v[4]\}$; for f^1, we can compute $V^1 = \{k[0], k[0] + k[1]\}$. Thus, $V^0 \cup V^1 = \{k[0], k[0] + k[1], k[1] + v[4]\}$. We introduce intermediate variables $Y \in \mathbb{F}_2^3$ and build a map $Y = g(x^0) = (g_0(x^0), g_1(x^0), g_2(x^0))$ between x^0 and Y, such that $Y[0] = g_0(x^0) = k[0]$, $Y[1] = g_1(x^0) = k[0] + k[1]$ and $Y[2] = g_2(x^0) = k[1] + v[4]$. Then, we have $\widehat{\mathbb{C}}_{(u^0,(0,0,1,0),u^2)}(Y) = Y[1]$, $\widehat{\mathbb{C}}_{(u^0,(0,1,1,0),u^2)}(Y) = Y[0]Y[2]$, and the superpoly can be represented compositely as $\widehat{\mathbb{C}}_{u^0,u^2}(g(x^0))$, where $\widehat{\mathbb{C}}_{u^0,u^2}(Y) = Y[1] + Y[0]Y[2]$.

Combining Proposition 2 and Proposition 3, we can recover the composite representation for the superpoly by adapting each coefficient involved in the recursive framework presented in [21] (see Appendix B) to its composite form. As this adaptation does not impact the model-solving process, the additional time cost introduced by the adaptation is negligible.

3.2 Simplifying Superpoly Based on Composite Representation

Once the cube $C_I(J)$ is selected, the map $Y = g(x^0) = (g_0(x^0), \ldots, g_{N-1}(x^0))$ from Proposition 1 is fully determined. Thus, for the superpoly $\mathbb{C}_{u^0,u^R}(x^0)$ defined in Theorem 1, instead of working directly with its potentially complex expression as a polynomial of x^0, we shift our focus to its more manageable composite representation $\widehat{\mathbb{C}}_{u^0,u^R}(Y)$. More generally, we consider the composite representation $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ for any two arbitrarily selected core monomials $\pi(x^{r_0}, t^{r_0})$ and $\pi(x^{r_1}, t^{r_1})$, where $0 \leq r_0 < r_1 \leq R$.

Basic Strategy of Conditional Superpoly. Our goal is to identify a set of variables $S \subseteq \{Y[0], \ldots, Y[N-1]\}$ for $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$, where $|S|$ is upper bounded by a preset threshold ℓ, such that $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}$ can be simplified to $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle$, by imposing the conditions $Y[i] = 0$ for all $Y[i] \in S$, which can be transformed into conditions $g_i(x^0) = 0$. Under these conditions, our primary focus becomes recovering the expression of $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle$, thus significantly reducing the computational complexity. For each $Y[i] \in S$, $g_i(x^0)$ is a polynomial of the secret variables k and the non-zero non-cube variables $v[\{0, \ldots, m-1\} \setminus (I \cup J)]$, thus these conditions establish relationship between these variables. In the special case where $J = \{0, \ldots, m-1\} \setminus I$ (i.e., all non-cube variables are set to 0), these conditions reduce to weak-key conditions imposed solely on k. We give an example to show how to use the strategy above to simplify the superpoly $\mathbb{C}_{u^0,u^R}(x^0)$.

Example 2. In Example 1, the composite representation for the superpoly is computed as $\widehat{\mathbb{C}}_{u^0,u^2}(Y) = Y[1] + Y[0]Y[2]$. If we impose the condition $Y[0] = 0$ or $Y[2] = 0$, namely $k[0] = 0$ (a weak-key condition) or $k[1] + v[4] = 0$, then $\widehat{\mathbb{C}}_{u^0,u^2}(Y)$ simplifies to $Y[1]$, which can be expanded into the polynomial $k[0] + k[1]$ of x^0.

Remark 1. One might argue that the strategy outlined above does not always work, as it overlooks the potential inherent relationships among the polynomials $g_0, \ldots, g_{N-1}$. For instance, if $g_0(x^0) = k[0]$ and $g_1(x^0) = k[0] + 1$, then the conditions $g_0(x^0) = 0$ and $g_1(x^0) = 0$ are unlikely to hold simultaneously. While such cases may arise in practice, our experiments on the cipher Trivium show that they occur rarely. Therefore, for simplicity, we assume that the conditions $Y[i] = 0$ for all $Y[i] \in S$ can always hold simultaneously.

Algorithm 1 shows how to recover the simplified expression $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle$ by enumerating core monomial trails under the conditions $Y[j] = 0$ for all $Y[j] \in S$. Since these imposed conditions eliminate certain originally valid core monomial trails from t^{r_0} to t^{r_1}, the number of core monomial trails to be enumerated is reduced compared to recovering the full expression $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}$. Consequently, the MILP model in Algorithm 1 can be solved more efficiently than the original model that enumerates all core monomial trails from t^{r_0} to t^{r_1} without any conditions imposed. Under the conditions, the divide-and-conquer strategy in

Proposition 3 is reformulated as follows:

$$\begin{aligned}\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle &= \sum_{\substack{t^{r_0+1}:\\ \widehat{\mathbb{C}}_{t^{r_0},t^{r_0+1}}(Y)\neq 0}} \left(\widehat{\mathbb{C}}_{t^{r_0},t^{r_0+1}} \bmod \langle S\rangle \cdot \widehat{\mathbb{C}}_{t^{r_0+1},t^{r_1}} \bmod \langle S\rangle\right) \\ &= \sum_{\substack{t^{r_1-1}:\\ \widehat{\mathbb{C}}_{t^{r_1-1},t^{r_1}}(Y)\neq 0}} \left(\widehat{\mathbb{C}}_{t^{r_0},t^{r_1-1}} \bmod \langle S\rangle \cdot \widehat{\mathbb{C}}_{t^{r_1-1},t^{r_1}} \bmod \langle S\rangle\right),\end{aligned} \tag{3}$$

where $\widehat{\mathbb{C}}_{t^{r_0},t^{r_0+1}} \bmod \langle S\rangle$ in the first summation and $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1-1}} \bmod \langle S\rangle$ in the second summation can be precomputed. Thus, once the set S of conditional variables is determined, we can recover the expression $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle$ using a recursive framework similar to the one in [21]. In particular, the recursive framework used to recover the expression $\widehat{\mathbb{C}}_{u^0,u^R} \bmod \langle S\rangle$ for the superpoly $\mathbb{C}_{u^0,u^R}(x^0)$ defined in Theorem 1 is detailed in Appendix C.

Criteria for Selecting Conditional Variables. The remaining question is: how do we choose the set S such that $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ is simplified as much as possible? To this end, we need a criterion to measure and compare the effectiveness of each variable $Y[i], 0 \le i < N$ in terms of simplifying $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$. Generally speaking, for a polynomial h of Y and any variable $Y[i], 0 \le i < N$, the more complex $h/Y[i]$ is and the less complex $h \bmod Y[i]$ is, the more effective $Y[i]$ is in terms of simplifying h. Thus, we propose the following two candidate criteria to quantify the complexity of $h/Y[i]$ and $h \bmod Y[i]$.

Definition 1 ("Number of Monomials" Criterion). *Let h be a polynomial of Y. When the concrete expression of h is available, we choose the number of monomials in the quotient $h/Y[i]$ and that in the remainder $h \bmod Y[i]$ for each $Y[i], 0 \le i < N$ as the criterion, referred to as the "Number of Monomials" criterion. Under this criterion, for two variables $Y[i]$ and $Y[j]$, where $0 \le i < j < N$, we compare their effectiveness in terms of simplifying $h(Y)$ as follows:*

- *$Y[j]$ (resp. $Y[i]$) is more effective than $Y[i]$ (resp. $Y[j]$) in terms of simplifying $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ if $h \bmod Y[j]$ (resp. $h \bmod Y[i]$) contains fewer monomials than $h \bmod Y[i]$ (resp. $h \bmod Y[j]$).*
- *If the two remainders have the same number of monomials, then the two variables are considered equally effective in terms of simplifying $h(Y)$.*

Definition 2 ("Algebraic Degree" Criterion). *Let h be a polynomial of Y. When the concrete expression of h is not available, we choose $Deg(h/Y[i])$ and $Deg(h \bmod Y[i])$ for each $Y[i], 0 \le i < N$ as the criterion, referred to as the "Algebraic Degree" criterion. Under this criterion, for two variables $Y[i]$ and $Y[j]$, where $0 \le i < j < N$, we compare their effectiveness in terms of simplifying $h(Y)$ as follows:*

- *$Y[i]$ (resp. $Y[j]$) is more effective than $Y[j]$ (resp. $Y[i]$) if $Deg(h \bmod Y[i]) < Deg(h \bmod Y[j])$ (resp. $Deg(h \bmod Y[j]) < Deg(h \bmod Y[i])$).*

Algorithm 1: Recover $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle$ by enumerating core monomial trails

Input: The composite representation $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ to be simplified, the set S of conditional variables
Output: The simplified expression $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle$

Initialize an empty model $\mathcal{M}$
$\mathcal{M}$.variables $\leftarrow$ a vector $\mathtt{t}^{r_0}$ of MILP binary variables with the same size as x^{r_0}
$\mathcal{M}$.constraints $\leftarrow \mathtt{t}^{r_0} = t^{r_0}$
foreach $s \in \{r_0, \ldots, r_1 - 1\}$ **do**
    Initialize a set V of binary vectors
    foreach *u and w corresponding to x^s and x^{s+1} that satisfy $\widehat{\mathbb{C}}_{u,w}(Y) \notin S$ and $\widehat{\mathbb{C}}_{u,w}(Y) \neq 0$* **do**
        Add the binary vector $u\|w$ into the set V
    end
    $\mathcal{M}$.variables $\leftarrow$ a vector $\mathtt{t}^{s+1}$ of MILP binary variables with the same size as x^{s+1}
    Use mathematical tools (e.g., Sagemath) to generate linear constraints for $\mathtt{t}^s$ and $\mathtt{t}^{s+1}$, such that the feasible solutions of $\mathtt{t}^s\|\mathtt{t}^{s+1}$ from these constraints are exactly the binary vectors in V, and then add these constraints to $\mathcal{M}$
end
$\mathcal{M}$.constraints $\leftarrow \mathtt{t}^{r_1} = t^{r_1}$
Configure the MILP solver (e.g., Gurobi) to enumerate all possible solutions of $\mathcal{M}$
if *All the solutions are computed* **then**
    Initialize a polynomial $p = 0$ of Y
    foreach *Solution of $\mathcal{M}$* **do**
        Read the values of $\mathtt{t}^{r_0+1}, \ldots, \mathtt{t}^{r_1-1}$ as $t^{r_0+1}, \ldots, t^{r_1-1}$
        Compute $\widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1}\ldots,t^{r_1})}(Y)$ manually
        Update $p = p + \widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1}\ldots,t^{r_1})}$
    end
end
return p as $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle$

– *If the remainders have the same algebraic degree, the variable with the higher-degree quotient is considered more effective.*
– *If both the remainders and quotients have the same degree, the two variables are considered equally effective in terms of simplifying $h(Y)$.*

After choosing the criterion, we can determine the set S by the selection procedure shown in Algorithm 2, which is based on a greedy algorithm. To intuitively demonstrate the effectiveness of the conditions determined by the selection procedure, we give a concrete example related to Trivium.

Example 3. For the cipher Trivium, the 849-round superpoly of the cube $C_{I_0}(J)$, where the cube indices I_0 is shown in Table 2 and $J = \{0, \ldots, 79\} \setminus I_0$, was pre-

Algorithm 2: The procedure for selecting the conditional variables used to simplify the composite representation of a coefficient

Input: The composite representation $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ to be simplified, the upper bound ℓ for $|S|$

Output: The set S of selected conditional variables

Initialize $S \leftarrow \emptyset$ and $h(Y) \leftarrow \widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$

while $|S| < \ell$ ***and*** $h(Y) \neq 0$ ***and*** $h(Y) \neq 1$ **do**

  Identify the most effective variable(s) in terms of simplifying $h(Y)$ from the variables not in S. If there are more than one such variables, randomly choose one

  Assume the selected variable is $Y[j]$, then update $S = S \cup \{Y[j]\}$

  $h \leftarrow h \bmod Y[j]$

end

return S

sented in [21]. We recover the composite representation of the 849-round superpoly as a polynomial of $Y \in \mathbb{F}_2^{484}$, which contains 3 340 488 monomials. Under the "Number of Monomials" criterion, we invoke Algorithm 2 by setting ℓ to 15. Finally, the algorithm returns a set of size 14. The set corresponds to 14 weak-key conditions as follows:

$$
\begin{aligned}
&k[62] = 0, && k[78]k[79] + k[53] = 0,\\
&k[78] + k[76]k[77] + k[51] = 0, && k[44] + k[42]k[43] + k[17] = 0,\\
&k[76] + k[74]k[75] + k[49] = 0, && k[42] + k[40]k[41] + k[15] = 0,\\
&k[46] + k[44]k[45] + k[19] = 0, && k[70] + k[68]k[69] + k[43] = 0,\\
&k[59] + k[57]k[58] + k[32] = 0, && k[61] + k[59]k[60] + k[34] = 0,\\
&k[74] + k[72]k[73] + k[47] = 0, && k[63] + k[61]k[62] + k[36] = 0,\\
&k[29] + k[27]k[28] + k[2] = 0, && 1 + k[67] + k[25] + k[23]k[24] = 0.
\end{aligned}
$$

The first condition reduces the number of monomials in the composite representation from 3 340 488 to 1 429 535; the second one reduces the number of monomials from 1 429 535 to 657 932. These 14 conditions are clearly independent of each other. After imposing the 14 weak-key conditions, the superpoly simplifies to the constant 0.

Similarly, for the 841-round superpoly of $C_{I_0}(J)$, which contains 257 877 monomials, we recover its composite representation as a polynomial of Y, which contains 9 monomials. We invoke Algorithm 2 by setting ℓ to 3. Finally, the algorithm returns a set of size 1. The set corresponds to a single weak-key condition $k[63] + k[61]k[62] + k[36] = 0$. Imposing this condition simplifies the superpoly to the constant 0. More interestingly, during Algorithm 2, we found another two conditions $k[50] + k[48]k[49] + k[23] = 0$ and $k[78] + k[76]k[77] + k[51] = 0$ that are equally effective as $k[63] + k[61]k[62] + k[36] = 0$. That is to say, if we compute the value of the 841-round superpoly as 1 in the online phase, then we can immediately obtain three bits of information about the secret

key, namely $k[63] + k[61]k[62] + k[36] = 1, k[50] + k[48]k[49] + k[23] = 1$ and $k[78] + k[76]k[77] + k[51] = 1$.

In Line 3 of Algorithm 2, we must identify the most effective variable in terms of simplifying $h(Y)$ from the remaining unselected variables. This presents a significant computational challenge when operating under the "Algebraic Degree" criterion: in Line 3 of Algorithm 2, how can we efficiently determine the algebraic degrees of $h/Y[i]$ and $h \bmod Y[i]$ for each unselected variable $Y[i] \notin S$, when computing the explicit polynomial expression of $h(Y)$ is infeasible?

3.3 Search for Effective Conditional Variables under "Algberaic Degree" Criterion

We focus on deriving upper bounds for the algebraic degrees of $h/Y[i]$ and $h \bmod Y[i]$. The following proposition shows how to achieve this based on core monomial trails.

Proposition 4 (Upper Bounds Based on Core Monomial Trails). *For a composite representation $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ and a set $S \subseteq \{Y[0], \ldots, Y[N-1]\}$ of variables, we define the polynomial $h = \widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle$. Then, for any variable $Y[i] \notin S$, we can upper bound $Deg(h/Y[i])$ as follows:*

$$Deg(h/Y[i]) \leq \max_{\substack{t^{r_0+1},\ldots,t^{r_1-1}:\forall s\in\{r_0,\ldots,r_1-1\},\\ \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq 0 \wedge \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\notin S;\\ \exists s\in\{r_0,\ldots,r_1-1\},\ s.t.\ \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)=Y[i]}} \sum_{\substack{s:r_0\leq s<r_1,\\ \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq Y[i]}} Deg\left(\widehat{\mathbb{C}}_{t^s,t^{s+1}}\right). \tag{4}$$

Similarly, we can upper bound $Deg(h \bmod Y[i])$ as follows:

$$Deg(h \bmod Y[i]) \leq \max_{\substack{t^{r_0+1},\ldots,t^{r_1-1}:\\ \forall s\in\{r_0,\ldots,r_1-1\},\widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq 0\\ \wedge \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\notin S \wedge \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq Y[i]}} \sum_{s:r_0\leq s<r_1} Deg\left(\widehat{\mathbb{C}}_{t^s,t^{s+1}}\right). \tag{5}$$

Proof. According to Proposition 2, we have

$$h = \sum_{\substack{t^{r_0+1},\ldots,t^{r_1-1}:\\ \forall s\in\{r_0,\ldots,r_1-1\},\widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq 0}} \widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1},\ldots,t^{r_1})} \bmod \langle S \rangle.$$

Thus, we can express $h/Y[i]$ as

$$h/Y[i] = \sum_{\substack{t^{r_0+1},\ldots,t^{r_1-1}:\forall s\in\{r_0,\ldots,r_1-1\},\\ \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq 0 \wedge \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\notin S;\\ \exists s\in\{r_0,\ldots,r_1-1\},\ \text{s.t.}\ \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)=Y[i]}} \widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1},\ldots,t^{r_1})}/Y[i].$$

Since $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1},\ldots,t^{r_1})}/Y[i]\right) \leq \sum_{\substack{s:r_0\leq s<r_1,\\ \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq Y[i]}} \mathrm{Deg}\left(\widehat{\mathbb{C}}_{t^s,t^{s+1}}\right)$, we obtain Inequality (4). Similarly, we can express $h \bmod Y[i]$ as

$$h \bmod Y[i] = \sum_{\substack{t^{r_0+1},\ldots,t^{r_1-1}:\\ \forall s\in\{r_0,\ldots,r_1-1\},\widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq 0\\ \bigwedge \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\notin S \bigwedge \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y)\neq Y[i]}} \widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1},\ldots,t^{r_1})} \bmod Y[i].$$

Since $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{(t^{r_0},t^{r_0+1},\ldots,t^{r_1})} \bmod Y[i]\right) \leq \sum_{s:r_0\leq s<r_1} \mathrm{Deg}\left(\widehat{\mathbb{C}}_{t^s,t^{s+1}}\right)$, we obtain Inequality (5). □

For the polynomial h in Line 3 of Algorithm 2, we obviously have $h = \widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle$. However, we can not directly apply the proposition above to upper bound $\mathrm{Deg}\left(h/Y[i]\right)$ and $\mathrm{Deg}\left(h \bmod Y[i]\right)$ for an unselected variable $Y[i]$, because we are considering the case when computing the concrete expression of h is impractical (i.e., enumerating core monomial trails is computationally prohibitive). Thus, we translate Inequality (4) into an automated approach using MILP models, which is shown in Algorithm 3. In Lines 6–16 of the algorithm, the model not only introduces two vectors $\mathtt{t}^s, \mathtt{t}^{s+1}$ of MILP variables to trace the propagation of core monomial prediction through each $f^s, r_0 \leq s < r_1$ under the conditions $Y[j] = 0$ for all $Y[j] \in S$, but also incorporates two MILP variables $(\mathtt{a}, \mathtt{b})$ to encode whether $\widehat{\mathbb{C}}_{\mathtt{t}^s,\mathtt{t}^{s+1}}(Y) = Y[i]$ and to record the algebraic degree of $\widehat{\mathbb{C}}_{\mathtt{t}^s,\mathtt{t}^{s+1}}(Y)$ (excluding $Y[i]$) when the transition from $\mathtt{t}^s$ to $\mathtt{t}^{s+1}$ is a valid propagation. The constraint $L \geq 1$ in the model encodes the condition $\exists s \in \{r_0,\ldots,r_1-1\}$, s.t. $\widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y) = Y[i]$ under the max operator in Inequality (4). The automated method for deriving the upper bound of $\mathrm{Deg}\left(h \bmod Y[i]\right)$ in Inequality (5) is nearly identical to Algorithm 3, with the exception that the constraint $L \geq 1$ is replaced by $L = 0$, which encodes the condition $\forall s \in \{r_0,\ldots,r_1-1\}, \widehat{\mathbb{C}}_{t^s,t^{s+1}}(Y) \neq Y[i]$ under the max operator in Inequality (5).

For the polynomial h in Line 3 of Algorithm 2, we can naturally apply Algorithm 3 to derive upper bounds of $\mathrm{Deg}\left(h/Y[i]\right)$ and $\mathrm{Deg}\left(h \bmod Y[i]\right)$ for each unselected variable $Y[i]$. These bounds can then be used as estimates for $\mathrm{Deg}\left(h/Y[i]\right)$ and $\mathrm{Deg}\left(h \bmod Y[i]\right)$ in Line 3 of Algorithm 2 to identify the most effective variable(s) under the "Algebraic Degree" criterion. However, to make the estimates more accurate, we further adopt a divide-and-conquer strategy based on the following proposition.

Proposition 5 (Divide-and-Conquer Strategy When Searching for Conditional Variables). *For the polynomial $h = \widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle$ in Line 3 of Algorithm 2, we choose an intermediate number j between r_0 and r_1. Then, for*

Algorithm 3: The automated method for deriving the upper bound of $\mathrm{Deg}\,(h/Y[i])$ based on core monomial trails

Input: The composite representation $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ to be simplified, the set S consisting of the conditional variables that have been selected, the polynomial $h = \widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle$, the variable $Y[i]$ for which the effectiveness will be evaluated

Output: The upper bound of $\mathrm{Deg}\,(h/Y[i])$

Initialize an empty MILP model $\mathcal{M}$
$\mathcal{M}$.variables $\leftarrow$ a vector $\mathtt{t}^{r_0}$ of MILP binary variables with the same size as x^{r_0}
$\mathcal{M}$.constraints $\leftarrow \mathtt{t}^{r_0} = t^{r_0}$
Initialize a linear expression $L = 0$
Initialize a linear expression $Obj = 0$
foreach $s \in \{r_0, \ldots, r_1 - 1\}$ **do**
  Initialize a set V of binary vectors
  foreach *u and w corresponding to x^s and x^{s+1} that satisfy $\widehat{\mathbb{C}}_{u,w}(Y) \notin S$ and $\widehat{\mathbb{C}}_{u,w}(Y) \neq 0$* **do**
    Compute a two-bit value $(a, b) \in \mathbb{F}_2^2$ by

$$a = \begin{cases} 0, & \text{if } \widehat{\mathbb{C}}_{u,w}(Y) \neq Y[i] \\ 1, & \text{if } \widehat{\mathbb{C}}_{u,w}(Y) = Y[i] \end{cases}; b = \begin{cases} 0, & \text{if } \widehat{\mathbb{C}}_{u,w}(Y) = Y[i] \\ \mathrm{Deg}\left(\widehat{\mathbb{C}}_{u,w}\right), & \text{if } \widehat{\mathbb{C}}_{u,w}(Y) \neq Y[i] \end{cases}.$$

    Add the binary vector $u\|w\|(a, b)$ into the set V
  end
  $\mathcal{M}$.variables $\leftarrow$ a vector $\mathtt{t}^{s+1}$ of MILP binary variables with the same size as x^{s+1}
  $\mathcal{M}$.variables $\leftarrow$ a vector $(\mathtt{a}, \mathtt{b})$ containing two MILP binary variables
  Generate linear constraints for $\mathtt{t}^s, \mathtt{t}^{s+1}$ and $(\mathtt{a}, \mathtt{b})$ such that the feasible solutions of $\mathtt{t}^s\|\mathtt{t}^{s+1}\|(\mathtt{a}, \mathtt{b})$ from these constraints are exactly the binary vectors in V, and then add these constraints to $\mathcal{M}$
  $L = L + \mathtt{a}$
  $Obj = Obj + \mathtt{b}$
end
$\mathcal{M}$.constraints $\leftarrow \mathtt{t}^{r_1} = t^{r_1}$
$\mathcal{M}$.constraints $\leftarrow L \geq 1$
Set the objective function of $\mathcal{M}$ to maximize Obj
Optimize the model $\mathcal{M}$
if *the model is successfully optimized* **then**
  return the value of Obj as the upper bound of $\mathrm{Deg}\,(h/Y[i])$
end

each unselected variable $Y[i] \notin S$, we can decompose $Deg\,(h/Y[i])$ as follows:

$$\begin{aligned} &Deg\,(h/Y[i]) \\ \leq \max_{\substack{t^j: \\ \widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)\neq 0}} \max \Bigg\{ &Deg\left(\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S\rangle\right)/Y[i]\right) + Deg\left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S \cup \{Y[i]\}\rangle\right), \\ &Deg\left(\left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle\right)/Y[i]\right) + Deg\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \cup \{Y[i]\}\rangle\right), \\ &Deg\left(\left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle\right)/Y[i]\right) + Deg\left(\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S\rangle\right)/Y[i]\right) \Bigg\}. \end{aligned}$$

Similarly, we can decompose $Deg\,(h \bmod Y[i])$ *as follows:*

$$\begin{aligned} &Deg\,(h \bmod Y[i]) \\ &\leq \max_{\substack{t^j: \\ \widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)\neq 0}} \left(Deg\left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S \cup \{Y[i]\}\rangle\right) + Deg\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \cup \{Y[i]\}\rangle\right)\right). \end{aligned}$$

In the above two inequalities, we can precompute $Deg\left(\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S\rangle\right)/Y[i]\right)$ *and* $Deg\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \cup \{Y[i]\}\rangle\right)$ *accurately for each* t^j *satisfying* $\widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y) \neq 0$.

Proof. According to Proposition 3, we have

$$\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} = \sum_{t^j:\widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)\neq 0} \left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \cdot \widehat{\mathbb{C}}_{t^j,t^{r_1}}\right).$$

Thus, we can express $h/Y[i]$ as

$$\begin{aligned} h/Y[i] = \sum_{\substack{t^j: \\ \widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)\neq 0}} &\Bigg(\left(\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S\rangle\right)/Y[i]\right) \cdot \left(\left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle\right)/Y[i]\right) \\ &+ \widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \cup \{Y[i]\}\rangle \cdot \left(\left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle\right)/Y[i]\right) \\ &+ \widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S \cup \{Y[i]\}\rangle \cdot \left(\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S\rangle\right)/Y[i]\right) \Bigg). \end{aligned}$$

This gives the decomposition of $\mathrm{Deg}\,(h/Y[i])$. For $h \bmod Y[i]$, we have

$$h \bmod Y[i] = \sum_{\substack{t^j: \\ \widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)\neq 0}} \left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \cup \{Y[i]\}\rangle \cdot \widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S \cup \{Y[i]\}\rangle\right),$$

which gives the decomposition of $\mathrm{Deg}\,(h \bmod Y[i])$. □

Applying the proposition above, the final procedure for estimating the algebraic degrees of $h/Y[i]$ and $h \bmod Y[i]$ in Line 3 of Algorithm 2 is shown in Algorithm 4.

Search for Effective Conditional Variables to Simplify Superpoly. Finally, the whole procedure of simplifying the superpoly $\mathbb{C}_{u^0,u^R}(x^0)$ can be summarized as follows:

1. Determine the maximum number ℓ of conditional variables to search. Under the "Algebraic Degree" criterion, apply Algorithm 2 to compute the set S of conditional variables, with the input being $\widehat{\mathbb{C}}_{u^0,u^R}(Y)$ and ℓ.
2. Under the conditions $Y[j] = 0$ for all $Y[j] \in S$, compute the concrete expression of $\widehat{\mathbb{C}}_{u^0,u^R} \bmod \langle S\rangle$ using a recursive framework (see Appendix C) based on the core monomial prediction technique. Expand this result into a polynomial in terms of x^0, yielding the desired conditional superpoly.

Algorithm 4: The procedure for estimating $\mathrm{Deg}\,(h/Y[i])$ and $\mathrm{Deg}\,(h \bmod Y[i])$ with a divide-and-conquer strategy

Input: The composite representation $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}}(Y)$ to be simplified, the set S consisting of the conditional variables that have been selected, the polynomial $h = \widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle$, the variable $Y[i]$ for which the effectiveness will be evaluated

Output: The estimates of $\mathrm{Deg}\,(h/Y[i])$ and $\mathrm{Deg}\,(h \bmod Y[i])$

Choose an intermediate number j (usually close to r_1) between r_0 and r_1. For Trivium, we will choose $j = r_1 - 300$

Following the propagation rules of core monomial prediction, compute a set T consisting of all t^j that can propagate to t^{r_1}

Initialize $d_0 = -\infty$ and $d_1 = -\infty$

foreach $t^j \in T$ **do**

  Compute $\widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)$ using Proposition 2

  if $\widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y) \neq 0$ ***and*** $\widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)$ *does not involve any variable in* S **then**

    Based on the expression of $\widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)$, accurately compute $A_1 = \mathrm{Deg}\left(\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \rangle\right)/Y[i]\right)$ and $A_0 = \mathrm{Deg}\left(\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \cup \{Y[i]\} \rangle\right)$

    Using the automated method in Algorithm 3, compute the upper bounds of $\mathrm{Deg}\left(\left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S \rangle\right)/Y[i]\right)$ and $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S \cup \{Y[i]\} \rangle\right)$ as B_1 and B_0, respectively

    if $d_1 > \max\{A_1 + B_1, A_0 + B_1, A_1 + B_0\}$ **then**

      Update $d_1 = \max\{A_1 + B_1, A_0 + B_1, A_1 + B_0\}$

    end

    if $d_0 > A_0 + B_0$ **then**

      Update $d_0 = A_0 + B_0$

    end

  end

end

return d_1 *and* d_0 *as the estimates of* $Deg\,(h/Y[i])$ *and* $Deg\,(h \bmod Y[i])$

4 Application to 852-Round Trivium

We apply the procedure of simplifying superpoly mentioned above to the Trivium cipher. As a result, we discovered 2-bit and 3-bit weak-key conditions for 852-round Trivium under two different cubes with the non-cube variables being set as 0, and successfully recover the weak-key superpolies under these conditions. All experiments are conducted using the Gurobi Solver (version 9.1.2) on a workstation equipped with high-speed processors (totally 32 cores and 64 threads). The source code and the recovered weak-key superpolies are available in our git repository.

Table 2: The two cubes we chose for weak-key superpoly recovery

I	Indices	Size
I_0	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	44
I_1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15, 17, 19, 22, 24, 26, 28, 30, 32, 34, 37, 39, 41, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 64, 67, 69, 71, 73, 75, 77, 79	43

4.1 Weak-Key Superpoly Recovery for 852-Round Trivium

Trivium is a lightweight, synchronous stream cipher developed by De Cannière and Preneel [10]. It has been recognized as one of the Profile 2 Algorithms in the eSTREAM portfolio [1] and has also been standardized as ISO/IEC International Standard 29192-3 [2], solidifying its position as a reliable cryptographic primitive.

The internal state of the cipher consists of 288 bits, represented as $s = (s[0], s[1], \ldots, s[287])$, and is divided into three separate registers. During initialization, the 80-bit secret key K is loaded into the first register, while the 80-bit initialization vector (IV) is loaded into the second register. The remaining bits are set to 0, except for the last three bits of the third register, which are initialized to 1. The initial state is formally given as:

$$\begin{aligned}
(s[0], s[1], \ldots, s[92]) &\leftarrow (K[0], K[1], \ldots, K[79], 0, \ldots, 0) \\
(s[93], s[94], \ldots, s[176]) &\leftarrow (IV[0], IV[1], \ldots, IV[79], 0, \ldots, 0) \\
(s[177], s[178], \ldots, s[287]) &\leftarrow (0, 0, \ldots, 0, 1, 1, 1)
\end{aligned}$$

The state update process is defined by the following pseudo-code:

$$\begin{aligned}
t_1 &\leftarrow s[65] \oplus s[90] \cdot s[91] \oplus s[92] \oplus s[170], \\
t_2 &\leftarrow s[161] \oplus s[174] \cdot s[175] \oplus s[176] \oplus s[263], \\
t_3 &\leftarrow s[242] \oplus s[285] \cdot s[286] \oplus s[287] \oplus s[68], \\
(s[0], s[1], \ldots, s[92]) &\leftarrow (t_3, s[0], s[1], \ldots, s[91]), \\
(s[93], s[94], \ldots, s[176]) &\leftarrow (t_1, s[93], s[94], \ldots, s[175]), \\
(s[177], s[178], \ldots, s[287]) &\leftarrow (t_2, s[177], s[178], \ldots, s[286]).
\end{aligned}$$

After completing 1152 initialization rounds, the cipher produces one keystream bit per update. The keystream bit is computed as:

$$z = s[65] \oplus s[92] \oplus s[161] \oplus s[176] \oplus s[242] \oplus s[287].$$

Our cube attacks target the initialization phase of Trivium.

Weak-Key Superpoly Recovery. In [21], the authors successfully recovered the superpoly of a cube I_0 (see Table 2) for 851 rounds of Trivium. We adjust I_0 slightly and obtain another cube I_1 (see Table 2). In this work, we adopt these two cubes and fix all non-cube variables to the constant 0.

For the cube $C_{I_0}(J)$, where $J = \{0, \ldots, 79\} \setminus I_0$, let the superpoly be denoted as $\mathbb{C}_{u^0,u^{852}}(x^0)$ as in Theorem 1. We compute the intermediate variables $Y = (g_0(x^0), \ldots, g_{483}(x^0)) \in \mathbb{F}_2^{484}$. Then under the "Algebraic Degree" criterion, we apply Algorithm 2 with $\ell = 2$ to search for conditional variables of Y to simplify the composite representation $\widehat{\mathbb{C}}_{u^0,u^{852}}(Y)$. As a result, the algorithm returns a set $S = \{Y[287], Y[190]\}$, where $Y[287]$ and $Y[190]$ correspond to $g_{287} = k[62]$ and $g_{190} = k[42]+k[40]k[41]+k[15]$, respectively. The first identified conditional variable is $Y[287]$, for which we estimate $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{u^0,u^{852}}/Y[287]\right)$ and $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod Y[287]\right)$ as both 23 using Algorithm 3. For the second conditional variable $Y[190]$, we estimate $\mathrm{Deg}\left(\left(\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod Y[287]\right)/Y[190]\right)$ and $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod \langle S\rangle\right)$ as 21 and 22, respectively, using Algorithm 3. Thus, imposing the condition $Y[190] = 0$ and $Y[287] = 0$ reduces the algebraic degree of $\widehat{\mathbb{C}}_{u^0,u^{852}}$ from approximately 24 to 22. Under the conditions $Y[190] = 0$ and $Y[287] = 0$, we recover the expression of $\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod \langle S\rangle$ using the recursive framework in Appendix C. Expanding the result into a polynomial of x^0 gives the weak-key superpoly of $C_{I_0}(J)$ under the conditions $k[62] = 0$ and $k[42]+k[40]k[41]+k[15] = 0$. The details of the weak-key superpoly are provided in Table 3.

For the cube $C_{I_1}(J)$, where $J = \{0, \ldots, 79\} \setminus I_1$, let the superpoly be denoted as $\mathbb{C}_{u^0,u^{852}}(x^0)$ as in Theorem 1. We compute the intermediate variables $Y = (g_0(x^0), \ldots, g_{491}(x^0)) \in \mathbb{F}_2^{492}$. Then under the "Algebraic Degree" criterion, we apply Algorithm 2 with $\ell = 3$ to search for conditional variables of Y to simplify the composite representation $\widehat{\mathbb{C}}_{u^0,u^{852}}(Y)$. The algorithm returns a set $S = \{Y[303], Y[142], Y[189]\}$, where $Y[303]$, $Y[142]$ and $Y[189]$ correspond to $g_{303} = k[58]$, $g_{142} = k[59]+k[57]k[58]+k[32]$ and $g_{189} = k[42]+k[40]k[41]+k[15]$, respectively. For the first identified conditional variable $Y[303]$, we estimate $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{u^0,u^{852}}/Y[303]\right)$ and $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod Y[303]\right)$ as 23 and 24, respectively, using Algorithm 3. For the second identified conditional variable $Y[142]$, $\mathrm{Deg}\left(\left(\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod Y[303]\right)/Y[142]\right)$ and $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod \langle\{Y[303], Y[142]\}\rangle\right)$ are estimated as 22 and 23, respectively. For the last identified conditional variable $Y[189]$, we estimate $\mathrm{Deg}\left(\left(\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod \langle\{Y[303], Y[142]\}\rangle\right)/Y[189]\right)$ and $\mathrm{Deg}\left(\widehat{\mathbb{C}}_{u^0,u^{852}} \bmod \langle S\rangle\right)$ as 21 and 22, respectively. That is to say, imposing the conditions $Y[303] = 0$, $Y[142] = 0$ and $Y[189] = 0$ reduces the algebraic degree of $\widehat{\mathbb{C}}_{u^0,u^{852}}$ from approximately 24 to 22. Under the conditions $k[58] = 0$, $k[59] + k[57]k[58] + k[32] = 0$ and $k[42] + k[40]k[41] + k[15] = 0$, we recover the weak-key superpoly of $C_{I_1}(J)$, whose details are provided in Table 3.

Due to memory constraints of our workstation in storing the fully expanded forms of the two weak-key superpolies, we evaluated the algebraic degrees, the number of monomials and the number of involved key bits for these two weak-key superpolies without considering the cancellation of monomials. Thus, the values presented in Table 3 represent only upper bounds of these parameters.

Table 3: Details related to the weak-key superpolies recovered for TRIVIUM

I	#Conditions*	Rounds	Time Cost	Balancedness§	#Monomials⋆	#Key Bits⋆	Degree⋆
I_0	2	852	376 hours	0.50	4 751 202 049 559	78	35
I_1	3	852	256 hours	0.50	778 123 908 999	77	34

§ The balancedness of each superpoly is estimated by testing 2^{15} random keys.
* The number of bit conditions imposed on the key.
⋆ An upper bound on the number of monomials, the number of involved key bits or the algebraic degree.

For instance, the polynomial of k generated from $Y[0]Y[1]Y[2]$ with $Y[0] = k[0]k[1] + 1, Y[1] = k[2]k[3] + k[0]k[1]$ and $Y[2] = k[5] + k[6]$ would be counted as containing $2 \times 2 \times 2 = 8$ monomials in our evaluation.

4.2 Weak-Key Cube Attack against Trivium

We adopt the following key-recovery strategy from [21] to extract the key information from an equation $p(k) = a, a \in \mathbb{F}_2$ established by a weak-key superpoly p:

1. Select m key bits involved in p to guess.
2. For each guess, reduce p to a polynomial p' of the remaining unguessed key bits, and then construct a truth table of p' using the Möbius transform. In the truth table of p', if there is any entry whose value is a, then together with the guessed value, we obtain a candidate solution for $p(k) = a$, whose correctness can be checked by performing an additional encryption call.

The memory complexity of this key-recovery strategy comprises the storage requirements for polynomials p and p', plus the memory needed to execute the Möbius transform. The time complexity is primarily determined by three operations: reducing polynomial p to p', performing the Möbius transform, and conducting additional encryption calls to verify candidate solutions.

Consider the case when p is the weak-key superpoly of I_0 in Table 3. The number of monomials in p is upper bounded by 4 751 202 049 559 without considering the cancellation of monomials. Thus, we can estimate the actual number of monomials in this superpoly as $\frac{4\,751\,202\,049\,559}{10} \approx 2^{38.79}$. Since storing each monomial requires 80 bits, the memory of storing this weak-key superpoly is approximately $2^{38.79} \times 80 \approx 2^{45.11}$ bits, and the time complexity of reducing p to p' requires $2^{45.11}$ bit operations. We choose $m = 32$, then the memory complexity of the key-recovery strategy above is $2 \times 2^{45.11} + 2^{78-32} \approx 2^{47.06}$ bits, and the time complexity is $2^{32} \times (2^{45.11} + 46 \times 2^{46}) \approx 2^{83.54}$ bits operations plus 2^{77} encryption calls, which is slightly more than 2^{77} TRIVIUM calls if we treat an 852-round TRIVIUM call as $852 * 9 = 7668$ bit operations. As pointed out by [17], this is an implementation-dependent attack, as we convert a TRIVIUM call into a specific number of bit operations.

For the case when p is the weak-key superpoly of I_1 in Table 3, we choose $m = 34$. Following the complexity analysis above, the final memory complexity is about $2^{44.27}$ bits, while the time complexity is slightly more than 2^{76} TRIVIUM calls.

5 Conclusion

In this paper, we expand the capabilities of the core monomial prediction technique by introducing a composite representation for the superpoly. This representation enables us to identify potential simplifications in the algebraic structure of the superpoly under specific conditions on intermediate variables, while avoiding the computational overhead of trail enumeration. We propose a greedy algorithm that efficiently searches for optimal conditions to maximize superpoly simplification. Through this algorithm, we successfully identified 2-bit and 3-bit conditions for the superpoly, resulting in weak-key superpoly recovery and the corresponding weak-key cube attack on 852-round Trivium.

References

1. eSTREAM: the ECRYPT stream cipher project (2018). https://www.ecrypt.eu.org/stream/. Accessed: 2021-03-23.
2. ISO/IEC 29192-3:2012: Information technology Security techniques Lightweight cryptography part 3: Stream ciphers. https://www.iso.org/standard/56426.html.
3. Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round MD6 and trivium. In Orr Dunkelman, editor, *FSE 2009*, volume 5665 of *LNCS*, pages 1–22. Springer, 2009.
4. Jules Baudrin, Anne Canteaut, and Léo Perrin. Practical cube attack against nonce-misused ascon. *IACR Trans. Symmetric Cryptol.*, 2022(4):120–144, 2022.
5. Tim Beyne. A geometric approach to linear cryptanalysis. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 36–66. Springer, 2021.
6. Tim Beyne. A geometric approach to symmetric-key cryptanalysis. 2023.
7. Tim Beyne and Michiel Verbauwhede. Integral cryptanalysis using algebraic transition matrices. *IACR Trans. Symmetric Cryptol.*, 2023(4):244–269, 2023.
8. Christina Boura and Anne Canteaut. Another view of the division property. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 654–682. Springer, 2016.
9. Christina Boura and Daniel Coggia. Efficient MILP modelings for sboxes and linear layers of SPN ciphers. *IACR Trans. Symmetric Cryptol.*, 2020(3):327–361, 2020.
10. Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *LNCS*, pages 244–266. Springer, 2008.
11. Cheng Che and Tian Tian. An experimentally verified attack on 820-round trivium. In Yi Deng and Moti Yung, editors, *Information Security and Cryptology - 18th International Conference, Inscrypt 2022, Beijing, China, December 11-13, 2022, Revised Selected Papers*, volume 13837 of *Lecture Notes in Computer Science*, pages 357–369. Springer, 2022.

12. Cheng Che and Tian Tian. A new correlation cube attack based on division property. In Leonie Simpson and Mir Ali Rezazadeh Baee, editors, *Information Security and Privacy - 28th Australasian Conference, ACISP 2023, Brisbane, QLD, Australia, July 5-7, 2023, Proceedings*, volume 13915 of *Lecture Notes in Computer Science*, pages 53–71. Springer, 2023.
13. Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Cube attacks and cube-attack-like cryptanalysis on the round-reduced keccak sponge function. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 733–761. Springer, 2015.
14. Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.
15. Itai Dinur and Adi Shamir. Breaking Grain-128 with dynamic cube attacks. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 167–187. Springer, 2011.
16. Xiaoyang Dong, Zheng Li, Xiaoyun Wang, and Ling Qin. Cube-like attack on round-reduced initialization of ketje sr. *IACR Trans. Symmetric Cryptol.*, 2017(1):259–280, 2017.
17. Hao Fan, Yonglin Hao, Qingju Wang, Xinxin Gong, and Lin Jiao. Key filtering in cube attacks from the implementation aspect. In Jing Deng, Vladimir Kolesnikov, and Alexander A. Schwarzmann, editors, *Cryptology and Network Security - 22nd International Conference, CANS 2023, Augusta, GA, USA, October 31 - November 2, 2023, Proceedings*, volume 14342 of *Lecture Notes in Computer Science*, pages 293–317. Springer, 2023.
18. Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 502–517. Springer, 2013.
19. Yuki Funabiki, Yosuke Todo, Takanori Isobe, and Masakatu Morii. Improved integral attack on HIGHT. In *ACISP 2017*, pages 363–383, 2017.
20. Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128AEAD. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 466–495. Springer, 2020.
21. Jiahui He, Kai Hu, Hao Lei, and Meiqin Wang. Massive superpoly recovery with a meet-in-the-middle framework - improved cube attacks on trivium and kreyvium. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 368–397. Springer, 2024.
22. Jiahui He, Kai Hu, Bart Preneel, and Meiqin Wang. Stretching cube attacks: Improved methods to recover massive superpolies. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 537–566. Springer, 2022.

23. Kai Hu, Siwei Sun, Yosuke Todo, Meiqin Wang, and Qingju Wang. Massive superpoly recovery with nested monomial predictions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 392–421. Springer, 2021.
24. Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 446–476. Springer, 2020.
25. Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. Conditional cube attack on reduced-round keccak sponge function. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 259–288, 2017.
26. Lars R. Knudsen and David A. Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.
27. Hao Lei, Jiahui He, Kai Hu, and Meiqin Wang. More balanced polynomials: Cube attacks on 810- and 825-round trivium with practical complexities. *IACR Cryptol. ePrint Arch.*, page 1237, 2023.
28. Zheng Li, Wenquan Bi, Xiaoyang Dong, and Xiaoyun Wang. Improved conditional cube attacks on keccak keyed modes with MILP method. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 99–127. Springer, 2017.
29. Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. Conditional cube attack on round-reduced ASCON. *IACR Trans. Symmetric Cryptol.*, 2017(1):175–202, 2017.
30. Meicheng Liu, Jingchun Yang, Wenhao Wang, and Dongdai Lin. Correlation cube attacks: From weak-key distinguisher to key recovery. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 715–744. Springer, 2018.
31. Piotr Mroczkowski and Janusz Szmidt. The cube attack on stream cipher Trivium and quadraticity tests. *Fundam. Informaticae*, 114(3-4):309–318, 2012.
32. Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. Misuse-free key-recovery and distinguishing attacks on 7-round ascon. *IACR Trans. Symmetric Cryptol.*, 2021(1):130–155, 2021.
33. Raghvendra Rohit and Santanu Sarkar. Diving deep into the weak keys of round reduced ascon. *IACR Trans. Symmetric Cryptol.*, 2021(4):74–99, 2021.
34. Md. Iftekhar Salam, Harry Bartlett, Ed Dawson, Josef Pieprzyk, Leonie Simpson, and Kenneth Koon-Ho Wong. Investigating cube attacks on the authenticated encryption stream cipher ACORN. In Lynn Batten and Gang Li, editors, *ATIS 2016*, volume 651 of *Communications in Computer and Information Science*, pages 15–26, 2016.

35. Yu Sasaki and Yosuke Todo. New algorithm for modeling s-box in MILP based differential and division trail search. In Pooya Farshim and Emil Simion, editors, *SecITC 2017*, volume 10543 of *LNCS*, pages 150–165. Springer, 2017.
36. Ling Song, Jian Guo, Danping Shi, and San Ling. New MILP modeling: Improved conditional cube attacks on keccak-based constructions. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 65–95. Springer, 2018.
37. Ling Sun, Wei Wang, and Meiqin Wang. Automatic search of bit-based division property for ARX ciphers and word-based division property. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017*, volume 10624 of *LNCS*, pages 128–157. Springer, 2017.
38. Ling Sun, Wei Wang, and Meiqin Wang. Milp-aided bit-based division property for primitives with non-bit-permutation linear layers. *IET Information Security*, 14(1):12–20, 2020.
39. Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 158–178. Springer, 2014.
40. Yao Sun. Automatic search of cubes for attacking stream ciphers. *IACR Trans. Symmetric Cryptol.*, 2021(4):100–123, 2021.
41. Wil Liam Teng, Md. Iftekhar Salam, Wei-Chuen Yau, Josef Pieprzyk, and Raphaël C.-W. Phan. Cube attacks on round-reduced tinyjambu. *IACR Cryptol. ePrint Arch.*, page 1164, 2021.
42. Yosuke Todo. Integral cryptanalysis on full MISTY1. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015*, volume 9215 of *LNCS*, pages 413–432, 2015.
43. Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015.
44. Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017*, volume 10403 of *LNCS*, pages 250–279. Springer, 2017.
45. Yosuke Todo and Masakatu Morii. Bit-based division property and application to Simon family. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 357–377. Springer, 2016.
46. Jianhua Wang, Lu Qin, and Baofeng Wu. Correlation cube attack revisited - improved cube search and superpoly recovery techniques. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part III*, volume 14440 of *Lecture Notes in Computer Science*, pages 190–222. Springer, 2023.
47. Qingju Wang, Lorenzo Grassi, and Christian Rechberger. Zero-sum partitions of PHOTON permutations. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 279–299. Springer, 2018.

48. Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018*, volume 10991 of *LNCS*, pages 275–305. Springer, 2018.
49. SenPeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. MILP-aided method of searching division property using three subsets and applications. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019*, volume 11923 of *LNCS*, pages 398–427. Springer, 2019.
50. Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 648–678. Springer, 2016.
51. Chen-Dong Ye and Tian Tian. A new framework for finding nonlinear superpolies in cube attacks against trivium-like ciphers. In Willy Susilo and Guomin Yang, editors, *ACISP 2018*, volume 10946 of *LNCS*, pages 172–187. Springer, 2018.
52. Chen-Dong Ye and Tian Tian. A practical key-recovery attack on 805-round trivium. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 187–213. Springer, 2021.

Appendix

A Propagation Rules of Core Monomial Prediction

Rule 1 (COPY [22]) *Let* $f^i : \mathbb{F}_2^{n_{in}} \to \mathbb{F}_2^{n_{out}}, 0 \le i < R$ *be the basic operation COPY* ($n_{out} = n_{in} + \ell - 1$) *with* $x^i = (x^i[0], x^i[1], \ldots, x^i[n_{in} - 1])$ *and* $x^{i+1} = (x^i[0], \ldots, x^i[0], x^i[1], \ldots, x^i[n_{in} - 1])$, *where the first* ℓ *bits of* x^{i+1} *are generated from* $x^i[0]$ *by COPY. Given two core monomials* $\pi(x^i, t^i)$ *and* $\pi(x^{i+1}, t^{i+1})$, $\mathbb{C}_{t^i, t^{i+1}}(x^i) \neq 0$ *if and only if*

- $t^{i+1}[\ell + j - 1] = t^i[j]\ \forall\ 1 \le j \le n_{in} - 1$.
- $t^{i+1}[0] \vee t^{i+1}[1] \vee \cdots \vee t^{i+1}[\ell - 1] = t^i[0]$.

Rule 2 (AND [22]) *Let* $f^i : \mathbb{F}_2^{n_{in}} \to \mathbb{F}_2^{n_{out}}, 0 \le i < R$ *be the basic operation AND* ($n_{out} = n_{in} - \ell + 1$) *with* $x^i = (x^i[0], x^i[1], \ldots, x^i[n_{in} - 1])$ *and* $x^{i+1} = (x^i[0] \wedge x^i[1] \wedge \cdots \wedge x^i[\ell - 1], x^j[\ell], \ldots, x^i[n_{in} - 1])$. *Given two core monomials* $\pi(x^i, t^i)$ *and* $\pi(x^{i+1}, t^{i+1})$, $\mathbb{C}_{t^i, t^{i+1}}(x^i) \neq 0$ *if and only if*

- $t^{i+1}[j - \ell + 1] = t^i[j]\ \forall\ \ell \le j \le n_{in} - 1$.
- $t^{i+1}[0] = t^i[0] \vee t^i[1] \vee \cdots \vee t^i[\ell - 1]$.
- $\forall\ j \in \{0, 1, \ldots, \ell - 1\}$, *if* $x^i[j]$ *is a* δ*-variable, then* $t^{i+1}[0] = t^i[j]$.

Rule 3 (XOR [22]) *Let* $f^i : \mathbb{F}_2^{n_{in}} \to \mathbb{F}_2^{n_{out}}, 0 \le i < R$ *be the basic operation XOR* ($n_{out} = n_{in} - \ell + 1$) *with* $x^i = (x^i[0], x^i[1], \ldots, x^i[n_{in} - 1])$ *and* $x^{i+1} = (x^i[0] \oplus x^i[1] \oplus \cdots \oplus x^i[\ell - 1], x^j[\ell], \ldots, x^i[n_{in} - 1])$. *Given two core monomials* $\pi(x^i, t^i)$ *and* $\pi(x^{i+1}, t^{i+1})$, $\mathbb{C}_{t^i, t^{i+1}}(x^i) \neq 0$ *if and only if*

- $t^{i+1}[j - \ell + 1] = t^i[j]\ \forall\ \ell \le j \le n_{in} - 1$.
- *If there exist a* δ*-variable and a* 1_c*-variable among* $x^i[0], \ldots, x^i[\ell - 1]$, *then* $t^{i+1}[0] \ge t^i[0] + t^i[1] + \cdots + t^i[\ell - 1]$; *otherwise* $t^{i+1}[0] = t^i[0] + t^i[1] + \cdots + t^i[\ell - 1]$.

B Recursive Framework for Superpoly Recovery in [21]

Essentially, Theorem 2 provides two perspectives for decomposing $\mathbb{C}_{t^{r_0}, t^{r_1}}(f^{r_0 - 1} \circ \cdots \circ f^0(x^0))$ as follows:

- Backward expansion: we choose a number j (usually close to r_1) between r_0 and r_1, precompute $\mathbb{C}_{t^j, t^{r_1}}(f^{j-1} \circ \cdots \circ f^0(x^0))$ for each t^j that satisfies $\mathbb{C}_{t^j, t^{r_1}}(f^{j-1} \circ \cdots \circ f^0(x^0)) \neq 0$, and then focus on computing $\mathbb{C}_{t^{r_0}, t^j}(f^{r_0 - 1} \circ \cdots \circ f^0(x^0))$ for each such t^j.
- Forward expansion: we choose a number j (usually close to r_0) between r_0 and r_1, precompute $\mathbb{C}_{t^{r_0}, t^j}(f^{r_0 - 1} \circ \cdots \circ f^0(x^0))$ for each t^j that satisfies $\mathbb{C}_{t^{r_0}, t^j}(f^{r_0 - 1} \circ \cdots \circ f^0(x^0)) \neq 0$, and then focus on computing $\mathbb{C}_{t^j, t^{r_1}}(f^{j-1} \circ \cdots \circ f^0(x^0))$ for each such t^j.

According to [21], the superpoly $\mathbb{C}_{u^0, u^R}(x^0)$ defined in Theorem 1 can be computed by the following recursive framework:

1. Prepare a hash table P whose key is a pair of binary vectors and value is a Boolean polynomial of x^0, and initialize P as $P[(u^0, u^R)] = 1$. Initialize $r_0 = 0, r_1 = R$. Prepare a binary variable d to represent the direction of the expansion and initialize $d = 1$. Initialize a Boolean polynomial $p = 0$ to store the results.
2. If $r_e < B$ ($B = 350$ for TRIVIUM), we flip the value of d. Prepare an empty hash table P_e of the same type as P.
3. If $d = 0$, we use forward expansion. Namely, we choose a number between j (close to r_0) between r_0 and r_1, and then for each pair (t^{r_0}, t^{r_1}) as a key of P:
   (a) By constructing a MILP model to describe the propagation of core monomial prediction through $f^{j-1} \circ \cdots \circ f^{r_0}$ starting from t^{r_0}, determine all t^j that satisfy the condition that there exists at least one core monomial trail from t^{r_0} to t^j.
   (b) For each such t^j, compute $\mathbb{C}_{t^{r_0},t^j}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ using Theorem 3, and if $\mathbb{C}_{t^{r_0},t^j}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ is not 0, we consider two cases: if the pair (t^j, t^{r_1}) is already a key of P_e, we update P_e by $P_e[(t^j, t^{r_1})] = P_e[(t^j, t^{r_1})] + \mathbb{C}_{t^{r_0},t^j}(f^{r_0-1} \circ \cdots \circ f^0(x^0)) \cdot P[(t^{r_0}, t^{r_1})]$; otherwise we add the pair (t^j, t^{r_1}) to P_e by letting $P_e[(t^j, t^{r_1})] = \mathbb{C}_{t^{r_0},t^j}(f^{r_0-1} \circ \cdots \circ f^0(x^0)) \cdot P[(t^{r_0}, t^{r_1})]$.
   (c) Let $P = P_e$ and update $r_0 = j$.
4. If $d = 1$, we use backward expansion. Namely, we choose a number between j (close to r_1) between r_0 and r_1, and then for each pair (t^{r_0}, t^{r_1}) as a key of P:
   (a) By constructing a MILP model to describe the propagation of core monomial prediction through $f^{r_1-1} \circ \cdots \circ f^j$ ending at t^j, determine all t^j that satisfy the condition that there exists at least one core monomial trail from t^j to t^{r_1}.
   (b) For each such t^j, compute $\mathbb{C}_{t^j,t^{r_1}}(f^{j-1} \circ \cdots \circ f^0(x^0))$ using Theorem 3, and if $\mathbb{C}_{t^j,t^{r_1}}(f^{j-1} \circ \cdots \circ f^0(x^0))$ is not 0, we consider two cases: if the pair (t^{r_0}, t^j) is already a key of P_e, we update P_e by $P_e[(t^{r_0}, t^j)] = P_e[(t^{r_0}, t^j)] + \mathbb{C}_{t^j,t^{r_1}}(f^{j-1} \circ \cdots \circ f^0(x^0)) \cdot P[(t^{r_0}, t^{r_1})]$; otherwise we add the pair (t^{r_0}, t^j) to P_e by letting $P_e[(t^{r_0}, t^j)] = \mathbb{C}_{t^j,t^{r_1}}(f^{j-1} \circ \cdots \circ f^0(x^0)) \cdot P[(t^{r_0}, t^{r_1})]$.
   (c) Let $P = P_e$ and update $r_1 = j$.
5. For each pair (t^{r_0}, t^{r_1}) as a key of P, if $P[(t^{r_0}, t^{r_1})]$ is 0, we remove (t^{r_0}, t^{r_1}) from the keys of P.
6. If the size of P is not larger than N ($N = 50\,000$ for TRIVIUM), we jump to Step 2; otherwise we start to compute the coefficient $\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ for each pair (t^{r_0}, t^{r_1}) in P and prepare an empty hash table P_u of the same type as P.
7. For each pair (t^{r_0}, t^{r_1}) as a key of P, we compute the coefficient $\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ using Theorem 3 within a preset time limit. If the coefficient is computed within the time limit, we obtain $\mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0))$ and update p by $p = p + \mathbb{C}_{t^{r_0},t^{r_1}}(f^{r_0-1} \circ \cdots \circ f^0(x^0)) \cdot P[(t^{r_0}, t^{r_1})]$; if the coefficient is determined to be 0, we discard it; if the coefficient is not computed within the time limit, we add the pair to P_u by letting $P_u[(t^{r_0}, t^{r_1})] = P[(t^{r_0}, t^{r_1})]$.

8. If the size of P_u is 0, then p is returned as the final result, namely the superpoly; otherwise we let $P = P_u$ and jump back to Step 2.

C Recursive Framework for Recovering the Composite Representation of Superpoly

Essentially, Eq. (3) provides two perspectives for decomposing $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S\rangle$ as follows:

- Backward expansion: we choose a number j (usually close to r_1) between r_0 and r_1, precompute $\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S\rangle$ for each t^j that satisfies $\widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y) \neq 0$, and then focus on computing $\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle$ for each such t^j that satisfies $\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S\rangle \neq 0$.
- Forward expansion: we choose a number j (usually close to r_0) between r_0 and r_1, precompute $\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle$ for each t^j that satisfies $\widehat{\mathbb{C}}_{t^{r_0},t^j}(Y) \neq 0$, and then focus on computing $\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S\rangle$ for each such t^j that satisfies $\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle \neq 0$.

For the superpoly $\mathbb{C}_{u^0,u^R}(x^0)$ defined in Theorem 1, the expression $\widehat{\mathbb{C}}_{u^0,u^R} \bmod \langle S\rangle$ can be computed by the following recursive framework:

1. Prepare a hash table P whose key is a pair of binary vectors and value is a Boolean polynomial of Y, and initialize P as $P[(u^0,u^R)] = 1$. Initialize $r_0 = 0, r_1 = R$. Prepare a binary variable d to represent the direction of the expansion and initialize $d = 1$. Initialize a Boolean polynomial $p = 0$ to store the results.
2. If $r_e < B$ ($B = 350$ for Trivium), we flip the value of d. Prepare an empty hash table P_e of the same type as P.
3. If $d = 0$, we use forward expansion. Namely, we choose a number between j (close to r_0) between r_0 and r_1, and then for each pair (t^{r_0},t^{r_1}) as a key of P:
   (a) By constructing a MILP model to describe the propagation of core monomial prediction through $f^{j-1}\circ\cdots\circ f^{r_0}$ starting from t^{r_0}, determine all t^j that satisfy the condition that there exists at least one core monomial trail from t^{r_0} to t^j.
   (b) For each such t^j, compute $\widehat{\mathbb{C}}_{t^{r_0},t^j}(Y)$ and then $\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle$ using Proposition 2, and if $\widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle$ is not 0, we consider two cases: if the pair (t^j,t^{r_1}) is already a key of P_e, we update P_e by $P_e[(t^j,t^{r_1})] = P_e[(t^j,t^{r_1})] + \widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle \cdot P[(t^{r_0},t^{r_1})]$; otherwise we add the pair (t^j,t^{r_1}) to P_e by letting $P_e[(t^j,t^{r_1})] = \widehat{\mathbb{C}}_{t^{r_0},t^j} \bmod \langle S\rangle \cdot P[(t^{r_0},t^{r_1})]$.
   (c) Let $P = P_e$ and update $r_0 = j$.
4. If $d = 1$, we use backward expansion. Namely, we choose a number between j (close to r_1) between r_0 and r_1, and then for each pair (t^{r_0},t^{r_1}) as a key of P:

(a) By constructing a MILP model to describe the propagation of core monomial prediction through $f^{r_1-1} \circ \cdots \circ f^j$ ending at t^j, determine all t^j that satisfy the condition that there exists at least one core monomial trail from t^j to t^{r_1}.

(b) For each such t^j, compute $\widehat{\mathbb{C}}_{t^j,t^{r_1}}(Y)$ and then $\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \rangle$ using Proposition 2, and if $\widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \rangle$ is not 0, we consider two cases: if the pair (t^{r_0}, t^j) is already a key of P_e, we update P_e by $P_e[(t^{r_0}, t^j)] = P_e[(t^{r_0}, t^j)] + \widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \rangle \cdot P[(t^{r_0}, t^{r_1})]$; otherwise we add the pair (t^{r_0}, t^j) to P_e by letting $P_e[(t^{r_0}, t^j)] = \widehat{\mathbb{C}}_{t^j,t^{r_1}} \bmod \langle S \rangle \cdot P[(t^{r_0}, t^{r_1})]$.

(c) Let $P = P_e$ and update $r_1 = j$.

5. For each pair (t^{r_0}, t^{r_1}) as a key of P, if $P[(t^{r_0}, t^{r_1})]$ is 0, we remove (t^{r_0}, t^{r_1}) from the keys of P.
6. If the size of P is not larger than N ($N = 50\,000$ for TRIVIUM), we jump to Step 2; otherwise we start to compute the expression $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle$ for each pair (t^{r_0}, t^{r_1}) in P and prepare an empty hash table P_u of the same type as P.
7. For each pair (t^{r_0}, t^{r_1}) as a key of P, we compute the expression $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle$ using Algorithm 1 within a preset time limit. If the expression is computed within the time limit, we obtain $\widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle$ and update p by $p = p + \widehat{\mathbb{C}}_{t^{r_0},t^{r_1}} \bmod \langle S \rangle \cdot P[(t^{r_0}, t^{r_1})]$; if the expression is determined to be 0, we discard it; if the expression is not computed within the time limit, we add the pair to P_u by letting $P_u[(t^{r_0}, t^{r_1})] = P[(t^{r_0}, t^{r_1})]$.
8. If the size of P_u is 0, then p is returned as the final result, namely the composite representation of the superpoly under the conditions $Y[j] = 0$ for all $Y[j] \in S$; otherwise we let $P = P_u$ and jump back to Step 2.