A Key-Recovery Attack on 855-round Trivium

Ximing Fu[1], Xiaoyun Wang[2,3,4(✉)], Xiaoyang Dong[2], and Willi Meier[5]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[2] Institute for Advanced Study, Tsinghua University, Beijing 100084, China
xiaoyunwang@mail.tsinghua.edu.cn
[3] School of Mathematics, Shandong University, Jinan 250100, China
[4] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
[5] FHNW, Windisch, Switzerland

Abstract. In this paper, we propose a key-recovery attack on Trivium reduced to 855 rounds. As the output is a complex Boolean polynomial over secret key and IV bits and it is hard to find the solution of the secret keys, we propose a novel nullification technique of the Boolean polynomial to reduce the output Boolean polynomial of 855-round Trivium. Then we determine the degree upper bound of the reduced nonlinear boolean polynomial and detect the right keys. These techniques can be applicable to most stream ciphers based on nonlinear feedback shift registers (NFSR). Our attack on 855-round Trivium costs time complexity 2^{77}. As far as we know, this is the best key-recovery attack on round-reduced Trivium. To verify our attack, we also give some experimental data on 721-round reduced Trivium.

Keywords: Trivium · Nullification technique · Polynomial reduction IV representation · Key-recovery attack

1 Introduction

Most symmetric cryptographic primitives can be described by boolean functions over secret variables and public variables. The secret variables are often key bits, the public variables are often plaintext bits for block ciphers and IV bits for stream ciphers. The ANF (algebraic normal form) representation of the output is usually very complex by repeatedly executing a simple iterative function, where the iterative function is a round function for block ciphers or a feedback function for stream ciphers based on nonlinear feedback shift registers. For stream ciphers, obtaining the exact output boolean functions is usually impossible. But if its degree is low, the cipher can not resist on many known attacks, such as higher order differential attacks [13,15], cube attacks [1,4], and integral attacks [14]. Hence, it is important to reduce the degree of polynomials for cryptanalysis of stream ciphers.

H. Shacham and A. Boldyreva (Eds.): CRYPTO 2018, LNCS 10992, pp. 160–184, 2018.
https://doi.org/10.1007/978-3-319-96881-0_6

Trivium, based on a nonlinear feedback shift register (NFSR), is one of the finalists by eSTREAM project and has been accepted as ISO standard [2,10]. Trivium has a simple structure, with only bit operations, so that it can be applicable to source restricted applications such as RFID. By iteratively using NFSR, the degree increases rapidly and the output is a complex boolean function over key and IV bits.

There have been lots of cryptanalysis of Trivium since its submission. The early results include the chosen IV statistical attack [6,7], which was applied to key-recovery attack on Trivium reduced to 672 rounds. Inspired by the message modification technique [20,21], Knellwolf et al. invented the conditional differential tool [11], which was applicable to distinguishing stream ciphers based on NFSR. In [12], Knellwolf et al. proposed a distinguishing attack on 961-round Trivium with practical complexity for weak keys.

Cube attacks are the major methods for recent cryptanalysis results of reduced round Trivium. In [4], Dinur and Shamir proposed a practical full key recovery on Trivium reduced to 767 rounds, using cube attacks. Afterwards, Aumasson et al. [1] provided the distinguishers of 790-round Trivium with complexity 2^{30}. Then Fouque and Vannet [8] provided a practical full key recovery for 784/799 rounds Trivium. Todo et al. [19] proposed a key-recovery attack on 832-round Trivium, where one equivalent bit can be recovered with complexity of around 2^{77}, combined with division property [18]. All of these attacks exploited low degree properties of the ANF of the output bit over IV bits. As though the degree is not low, i.e., the degree is equal to the number of variables, there is a possibility to construct distinguishers if there are missing (IV) terms. In [3,5], Dinur and Shamir exploited the density of IV terms, combined with nullification technique, and broke the full-round Grain128. Based on nullification technique [3,5], degree evaluation and IV representation techniques were proposed and the missing IV terms can be obtained with probability 1 [9]. The degree upper bounds of Trivium-like ciphers were obtained [16] using the degree evaluation techniques. Then a key-recovery attack on 835-round Trivium was proposed in [17] using correlation cube attack with a complexity of 2^{75}. Though the cube attack and cube tester tools can be applied to obtain the low-degree information, it is restricted by the computing ability. It is hard to execute cube tester programs of dimension more than 50 on a small cluster of cores.

In this paper, we focus on the cryptanalysis on round-reduced Trivium. We first propose a novel observation of the Boolean polynomial and invent a new nullification technique for reducing the output Boolean polynomial. After nullification, we determine the degree upper bound of the reduced polynomial, which can serve as the distinguishers. In this process, large quantities of state terms arise to be processed. We present a series of techniques to help discard monomials, including degree evaluation and degree reduction techniques. Based on these reduction techniques for boolean polynomials, we propose the first key-recovery attack on 855-round Trivium with time complexity 2^{77}. We summarize the related results in Table 1.

Table 1. Some related key-recovery results for reduced round Trivium.

Rounds	Complexity	Ref.
736	2^{30}	[4]
767	2^{36}	[4]
799	Practical	[8]
832	2^{77}	[19]
835	2^{75}	[17]
855	2^{77}	Sect. 4

The rest of the paper is organised as follows. In Sect. 2, some basic related preliminaries will be shown. The basic techniques used in this paper and the attack framework will be introduced in Sect. 3. Based on the Boolean polynomial reduction techniques and IV representation, a key recovery attack on 855-round Trivium is proposed in Sect. 4, combined with a new nullification technique. Finally, Sect. 5 summarizes the paper.

2 Preliminaries

In this section, some basic notations used in this paper are introduced in the following subsections.

2.1 Notations

ANF	the Algebraic Normal Form
IV bit	public variables of Trivium
IV term	product of certain IV bits
state bit	internal state bit in the initialization of Trivium stream cipher
state term	product of certain state bits, IV bits or key bits

2.2 Brief Description of Trivium

Trivium can be described by a 288-bit nonlinear feedback shift register s_i $(1 \leq i \leq 288)$. During the initialization stage, s_1 to s_{80} are set to 80 key bits, s_{94} to s_{173} are set with 80 IV bits, s_{286}, s_{287}, s_{288} are set to 1s and the other state bits are set to zeros, i.e.,

$$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_0, \ldots, K_{79}, 0, \ldots, 0)$$
$$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_0, \ldots, IV_{79}, 0, \ldots, 0)$$
$$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1).$$

Then the NFSR is updated for 1152 rounds with the following updating function, i.e.,

for $i \leftarrow 1 : 4 \cdot 288$ **do**
  $t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$
  $t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$
  $t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

After the initialization, the output bits o_i can be generated by the following functions.

for $i \leftarrow 1 : N$ **do**
  $t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$
  $t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$
  $t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$
  $o_i \leftarrow s_{66} + s_{93} + s_{162} + s_{177} + s_{243} + s_{288}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

Then the message can be encrypted by exclusive-or with o_i. To outline our technique more conveniently, we describe Trivium using the following iterative expression. We use s_w^r $(0 \leq w \leq 2)$ shown in Eq. 1 to illustrate r-round $(1 \leq r \leq 1152)$ s_1, s_{94} and s_{178} separately. Let z_r denote the output bit after r rounds of initialization. Then the initialization process can be illustrated by the following formula

$$\begin{aligned} s_0^r &= s_2^{r-66} + s_2^{r-109} s_2^{r-110} + s_2^{r-111} + s_0^{r-69}, \\ s_1^r &= s_0^{r-66} + s_0^{r-91} s_0^{r-92} + s_0^{r-93} + s_1^{r-78}, \\ s_2^r &= s_1^{r-69} + s_1^{r-82} s_1^{r-83} + s_1^{r-84} + s_2^{r-87}. \end{aligned} \tag{1}$$

The s_w^r $(0 \leq w \leq 2)$ is denoted as internal **state bit** in this paper. The multiplication of state bits $\prod_{i \in I, j \in J} s_i^j$ is denoted as a **state term**. The output can be described using the state terms as $z_r = s_0^{r-65} + s_0^{r-92} + s_1^{r-68} + s_1^{r-83} + s_2^{r-65} + s_2^{r-110}$.

2.3 Representation of Boolean Functions for Stream Ciphers

Supposing that there are m IV bits, i.e., $v_0, v_1, \ldots, v_{m-1}$ and n key bits, i.e., $k_0, k_1, \ldots, k_{n-1}$, the Algebraic Normal Form (ANF) of the internal state bit or output bit s could be written as the following style:

$$s = \sum_{I,J} \prod_{i \in I} v_i \prod_{j \in J} k_j, \tag{2}$$

where the sum operation is over field $\mathbb{F}_2$. The $\prod_{i \in I} v_i \prod_{j \in J} k_j$ is also denoted as a state term of s and $\prod_{i \in I} v_i$ is denoted as its corresponding IV term. Let IV term $t_I = \prod_{i \in I} v_i$ be the multiplication of v_i whose indices are within I, the ANF of s can be rewritten as

$$s = \sum_I t_I g_I(k), \tag{3}$$

where $g_I(k)$ is the sum of the corresponding coefficient function of terms whose corresponding IV term is t_I. The $|I|$ is denoted as the degree of IV term t_I, $\deg(t_I)$. The degree of s is $\deg(s) = \mathtt{max}_I\{\deg(t_I)\}$.

2.4 Cube Attack and Cube Tester

Cube attack [4] is introduced by Dinur and Shamir at EUROCRYPT 2009. This method is also known as high-order differential attack introduced by Lai [15] in 1994. It assumes the output bit of a cipher is a d-degree polynomial $f(k_0..., k_{n-1}, v_0..., v_{m-1})$ over $GF(2)$. The polynomial can be written as a sum of two polynomials:

$$f(k_0..., k_{n-1}, v_0..., v_{m-1}) = t_I \cdot P + Q_{t_I}(k_0..., k_{n-1}, v_0..., v_{m-1})$$

t_I is called maxterm and is a product of certain public variables, for example $(v_0, ..., v_{s-1}), 1 \leq s \leq m$, which is called a cube C_{t_I}; P is called superpoly; $Q_{t_I}(k_0..., k_{n-1}, v_0..., v_{m-1})$ is the remainder polynomial and none of its terms is divisible by t_I. The major idea of the cube attack is that the sum of f over all values of the cube C_{t_I} (cube sum) is:

$$\sum_{x'=(v_0,...,v_{s-1}) \in C_{t_I}} f(k_0, ..., k_{n-1}, x', ...v_{m-1}) = P$$

whose degree is at most d-s, where the cube C_{t_I} contains all binary vectors of length s and the other public variables are fixed to constants. In cube attack, P is a linear function over key bits. The key is recovered by solving a system of linear equations derived by different cubes C_{t_I}.

Dynamic cube attack [5] is also introduced by Dinur and Shamir in FSE 2011. The basic idea is to find dynamic variables, which depend on some of the public cube variables and some private variables (the key bits), to nullify the complex function $P = P_1 \cdot P_2 + P_3$, where the degree of P_3 is relatively lower than the degree of P and $P_1 \cdot P_2$ is a complex function. Then guess the involved key bits and compute the dynamic cube variables to make P_1 to be zero and the function is simplified greatly. The right guess of key bits will lead the cube sum to be zero otherwise the cube sums will be random generally.

Cube testers [1] are used to detect non-random properties. Suppose in Eq. 3, an IV term t_I does not exist in the ANF of s, e.g. the coefficient $g_I(k) = 0$. Hence, the cube sum over cube C_{t_I} is definitely zero for different key guessing. However, if the IV term t_I exists, the value of cube sum $g_I(k)$ is dependent on the key guessing. This property was applied to break full-round Grain128 [5,9].

3 Basic Ideas

3.1 New Observation of Boolean Polynomial Reduction

In this paper, we propose a new nullification technique based on a lemma as follows.

Lemma 1. *Suppose z is the output polynomial of a cipher, and*

$$z = P_1 P_2 + P_3. \tag{4}$$

Then the polynomial can be reduced to a simpler one $(1 + P_1)z = (1 + P_1)P_3$ *by multiplying* $1 + P_1$ *in both sides of Eq. (4) if* $\deg(P_1 P_2) > \deg((1 + P_1)P_3)$.

Lemma 1 can be verified by $(P_1 + 1)z = (P_1 + 1)P_1 P_2 + (P_1 + 1)P_3 = (P_1 + 1)P_3$. In our cryptanalysis of Trivium, P_1 is a simple polynomial over several IV bits and key bits, while P_2 is much more complex than P_3. In our nullification technique, we multiply $P_1 + 1$ in both sides of Eq. (4) to nullify the most complex polynomial P_2 without changing P_3. The result $(1 + P_1)z = (1 + P_1)P_3$ could be analyzed by considering P_3 and $1 + P_1$ independently, and then multiply them together to get $(1 + P_1)z$.

3.2 Outline of Our Attack

Based on the novel observation in Sect. 3.1, our attack includes two phases, which are the preprocessing phase and on-line attack phase.

In the preprocessing phase,

1. We apply the new nullification technique by determining P_1, then multiply $1 + P_1$ in both sides of Eq. 4 and obtain the reduced polynomial $(1 + P_1)P_3$.
2. We study the polynomial $(1 + P_1)P_3$ and prove its upper bound degree to be d mathematically, then cubes of dimension d + 1 lead to distinguishers.

In the on-line phase, we guess the partial key bits in P_1, and compute the cube sums of $(P_1 + 1)z$ over $(d + 1)$-degree IV terms:

i For the right key guessing, $(P_1 + 1)z = (P_1 + 1)P_3$. Thus the cube sums must be zero.

ii For the wrong key guessing, the equation becomes $(P_1' + 1)z = (P_1' + 1)P_1 P_2 + (P_1' + 1)P_3$, which is more complex and dominated by P_2, thus the cube sums are not always zero.

We focus on constructing the distinguishers in the preprocessing phase and it costs most computing sources.

3.3 Constructing Distinguishers

After obtaining the reduced polynomial $(1 + P_1)P_3$, our major work is to study this polynomial and derive distinguishers. In our analysis, we demonstrate that the degree of the reduced polynomial is strictly lower than 70. As the degree is so high, such a result was hard to achieve in previous works. So we introduce various details of reducing polynomials in an iterative process.

We introduce several techniques to discard monomials in advance during the iterative computation of the ANF representation of the output bit $(1 + P_1)P_3$. Suppose we are proving the upper bound degree of $(1 + P_1)P_3$'s ANF to be d, then the following techniques are used to reduce the Boolean polynomial of $(1 + P_1)P_3$ by discarding monomials in advance. The whole process could be divided into the following three steps shown in Fig. 1.

- **Step 1.** We compute forward to express the ANF of some internal state bits over IV bits and key bits. In Trivium, the internal state bits s_i^j ($0 \leq i \leq 2$, $0 \leq j \leq 340$) are computed in a PC.
- **Step 2.** During the iterative computation of the ANF representation of $(1 + P_1)P_3$ in the backward direction (decryption), we introduce the ***fast discarding monomial technique*** in Sect. 3.4, which includes the following two algorithms:
  - First, we propose the ***degree evaluation*** algorithm to obtain the degree bounds of internal state bits. As the monomials of $(1 + P_1)P_3$'s ANF is a product of these internal state bits, the degree of a monomial is bounded by the sum of the degrees of the multiplied internal state bits, which is regarded as the degree estimation of the monomial. If the estimated degrees of monomials are lower than d, they are discarded directly.
  - Second, we exploit the iterative structure of Trivium, and find that the $(1 + P_1)P_3$'s ANF contains many products of consecutive internal state bits. Thus, we pre-compute the ***degree reduction***s of those products, which is $d_t = \sum_i \deg(x_i) - \deg(\prod_i x_i)$, where x_i is an internal state bit. Thus, the degree of a monomial is upper bounded by the difference value between the sum of the multiplied internal state bits and the corresponding degree reduction d_t. If it is smaller than d, the monomial is discarded.
- **Step 3.** For the left monomials of $(1 + P_1)P_3$'s ANF, we introduce ***IV representation technique*** in Sect. 3.5 to determine the upper bound degree of $(1+P_1)P_3$ or find the d-degree missing product of certain IV bits (missing IV term). In IV representation technique, the symbolic key bits in the internal state bits are removed and only IV bits are left. Combining with repeated IV term removing algorithm, we can simplify monomials of $(1 + P_1)P_3$'s ANF without losing the missing IV term information. If we find an IV term is not in the IV representation of $(1 + P_1)P_3$, we can conclude that it is also not in $(1 + P_1)P_3$.

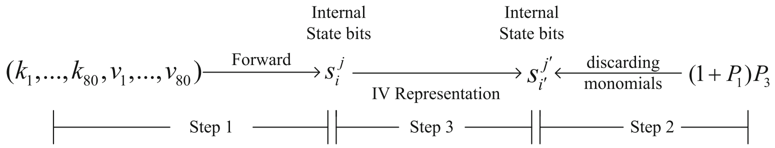

Fig. 1. Framework of constructing distinguishers

3.4 Fast Discarding Monomial Techniques

In Step 2 of Fig. 1, during the iterative computation of the ANF representation of $(1+P_1)P_3$ in the backward direction (decryption), there arise more and more state terms. We will give several techniques to simplify the polynomial by discarding monomials in advance. In this Step, repeated state terms arise according to the Trivium encryption scheme. The repeated state terms are removed using Algorithm 1. The complexity of Algorithm 1 is $O(n)$, supposing there are n state terms.

Algorithm 1. Repeated-(state)term Removing Algorithm

Input: The vector $\boldsymbol{T}$ with n terms, i.e., $T_1, T_2, \ldots, T_n$.
Output: Updated $\boldsymbol{T}$ with m terms, where $m \leq n$.

Initialize an empty Hash Set **H**.
for $i \leftarrow 1 : n$ **do**
    Compute the Hash value of T_i, i.e., $H(T_i)$
    if $H.contains(T_i)$ is **true then**
        $H.delete(T_i)$
    else
        $H.insert(T_i)$
    end if
end for

Degree evaluation technique. As we are proving the degree of the Boolean polynomial $(1+P_1)P_3$ to be d, thus many monomials with lower degree produced during the iterative computation backward (decryption) in Step 3 are deleted without consideration (we do not need to continue the iterative computation over those monomials). We estimate those monomials using degree information of the internal state bits in lower rounds. This section presents a degree evaluation algorithm for the internal state bits. For example, we are going to estimate the degree of $b_i = b_{i-3} + b_{i-1}b_{i-2}$.

$$\begin{aligned}\deg(b_i) &= \deg(b_{i-3} + b_{i-1}b_{i-2}) \\ &= \mathtt{max}\{\deg(b_{i-3}), \deg(b_{i-1}b_{i-2})\} \\ &\leq \mathtt{max}\{\deg(b_{i-3}), \deg(b_{i-1}) + \deg(b_{i-2})\}\end{aligned} \tag{5}$$

If we continue to decompose b_i, we find

$$\begin{aligned} b_{i-1}b_{i-2} &= (b_{i-4} + b_{i-2}b_{i-3})(b_{i-5} + b_{i-3}b_{i-4}) \\ &= b_{i-4}b_{i-5} + b_{i-3}b_{i-4} + b_{i-2}b_{i-3}b_{i-5} + b_{i-2}b_{i-3}b_{i-4}, \end{aligned} \tag{6}$$

If $\deg(b_{i-1}) = \deg(b_{i-2}b_{i-3})$ and $\deg(b_{i-2}) = \deg(b_{i-3}b_{i-4})$, then in Eq. (5), $\deg(b_{i-1}) + \deg(b_{i-2})$ may add $\deg(b_{i-3})$ twice. So in order to obtain a more accurate degree estimation, we are willing to decompose b_i for several rounds backwards.

For Trivium, the ANFs of s_i^j $(0 \leq i \leq 2, 0 \leq j \leq 340)$ are exactly obtained in a PC and their exact degrees can be obtained. For example, in the cryptanalysis of 855-round Trivium, we compute ANF of s_i^j $(0 \leq i \leq 2, 0 \leq j \leq 340)$ over 75 free IV variables[1], the degrees are shown in Table 2. To estimate the degree of s_i^r for $r > 340$, we decompose s_i^r until the state terms are the product of internal state bits s_i^j for $j < end = \lfloor \frac{r}{32} \rfloor \times 32 - 128$ considering the efficiency tradeoff of the computation.

Table 2. Degree $\deg(s_i^j)$ of s_i^j for $0 \leq i \leq 2$, $0 \leq j \leq 340$

$j+$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
$s_0^{j=0}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$s_1^{j=0}$	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
$s_2^{j=0}$	0	1	1	1	2	2	2	1	1	0	2	2	2	2	2	2	2	2	2	2	2	2	1	1	2	2	2	2	2	2	2	2	2	2	1
$s_0^{j=35}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
$s_1^{j=35}$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s_2^{j=35}$	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_0^{j=70}$	2	2	2	1	1	0	2	2	2	2	2	2	2	2	2	2	2	2	1	1	2	2	2	2	2	2	2	2	2	2	1	1	2	2	2
$s_1^{j=70}$	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
$s_2^{j=70}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_0^{j=105}$	2	2	2	2	2	2	2	2	3	3	3	3	2	1	1	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3
$s_1^{j=105}$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	2	2	2	1
$s_2^{j=105}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_0^{j=140}$	3	3	3	3	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3	3	3	3	3
$s_1^{j=140}$	1	1	2	2	2	2	2	2	2	2	2	2	2	2	1	1	2	2	2	2	2	3	3	3	3	2	1	2	3	3	3	3	3	3	3
$s_2^{j=140}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_0^{j=175}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	3	3	4	4	4	4
$s_1^{j=175}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	5	5	5	5	5	3
$s_2^{j=175}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_0^{j=210}$	3	4	4	4	4	4	4	4	4	3	4	4	3	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	3	4	4	4	4
$s_1^{j=210}$	2	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
$s_2^{j=210}$	2	2	2	2	2	2	2	2	3	3	3	3	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	5	5
$s_0^{j=245}$	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	4	4	4	4	4
$s_1^{j=245}$	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6	5	5
$s_2^{j=245}$	5	5	5	3	3	4	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6	5	5	6	6	6	6	6	6	6	6	6	6	6	6
$s_0^{j=280}$	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	4
$s_1^{j=280}$	6	6	6	6	6	6	6	5	5	6	6	7	7	7	7	7	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
$s_2^{j=280}$	6	6	6	6	7	7	8	8	8	8	8	8	5	6	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
$s_0^{j=315}$	4	4	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6	5	5	6	6	6	6	6	6	6									
$s_1^{j=315}$	7	7	7	7	7	7	7	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7									
$s_2^{j=315}$	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8									

[1] The other 5 IV bits are fixed as zero and their positions are given in Sect. 4.1.

For example, we estimate the degree upper bound of s_1^{341}, where $end = \lfloor \frac{r}{32} \rfloor \times 32 - 128 = 192$. We first express s_1^{341} using state bits in less rounds, and discard the state terms of degree lower than d.

- **Step 1.** First, we express $s_2^{341} = s_1^{272} + s_1^{259}s_1^{258} + s_1^{257} + s_2^{254}$ according to Eq. (1).
- **Step 2.** According to Table 2 highlighted in red, let $d = \mathtt{max}\{\deg(s_1^{272}), \deg(s_1^{259}) + \deg(s_1^{258}), \deg(s_1^{257}), \deg(s_2^{254}\} = \mathtt{max}\{5, 5+5, 5, 5\} = 10$.
- **Step 3.** Discarding the state terms of degree lower than 10, we get $s_2^{341*} = s_1^{259}s_1^{258}$. Iteratively compute s_2^{341*} and discard state terms with degree lower than 10, there is no state term surviving. We reset $d = d - 1$ and repeat the above decomposition and discarding process. We can get the result $s_2^{341**} = s_0^{166}s_0^{167}s_0^{193} + s_0^{167}s_0^{168}s_0^{192} + s_0^{166}s_0^{167}s_0^{168} + s_0^{165}s_0^{167}s_0^{168} + s_0^{167}s_0^{168}s_1^{180} + s_0^{166}s_0^{167}s_1^{181}$.
- **Step 4.** Note that there is still a state bit s_1^{193} in s_2^{341**} that is bigger than *end*=192. So we continue to iteratively compute and discard state terms with degree lower than 9, and we get:

$$
\begin{aligned}
&s_2^{341***} = s_2^{56}s_2^{57}s_2^{83}s_2^{84}s_2^{101} + s_2^{57}s_2^{58}s_2^{83}s_2^{84}s_2^{100} + s_2^{56}s_2^{57}s_2^{58}s_2^{83}s_2^{84} + \\
&s_0^{97}s_2^{57}s_2^{58}s_2^{83}s_2^{84} + s_0^{98}s_2^{56}s_2^{57}s_2^{83}s_2^{84} + s_0^{124}s_2^{56}s_2^{57}s_2^{101} + s_0^{124}s_2^{57}s_2^{58}s_2^{100} + \\
&s_0^{124}s_2^{56}s_2^{57}s_2^{58} + s_0^{124}s_2^{55}s_2^{57}s_2^{58} + s_2^{55}s_2^{57}s_2^{58}s_2^{83}s_2^{84} + s_0^{97}s_0^{124}s_2^{57}s_2^{58} \\
&+ s_0^{98}s_0^{124}s_2^{56}s_2^{57} + s_2^{57}s_2^{58}s_2^{82}s_2^{83}s_2^{102} + s_2^{58}s_2^{59}s_2^{82}s_2^{83}s_2^{101} + s_2^{57}s_2^{58}s_2^{59}s_2^{82}s_2^{83} + \\
&s_2^{56}s_2^{58}s_2^{59}s_2^{82}s_2^{83} + s_0^{98}s_2^{58}s_2^{59}s_2^{82}s_2^{83} + s_0^{99}s_2^{57}s_2^{58}s_2^{82}s_2^{83} + s_0^{123}s_2^{57}s_2^{58}s_2^{102} + \\
&s_0^{123}s_2^{58}s_2^{59}s_2^{101} + s_0^{123}s_2^{57}s_2^{58}s_2^{59} + s_0^{123}s_2^{56}s_2^{58}s_2^{59} + s_0^{98}s_0^{123}s_2^{58}s_2^{59} \\
&+ s_0^{99}s_0^{123}s_2^{57}s_2^{58} + s_2^{56}s_2^{57}s_2^{58}s_2^{59}s_2^{101} + s_0^{98}s_2^{56}s_2^{57}s_2^{58}s_2^{59} + s_2^{55}s_2^{56}s_2^{57}s_2^{58}s_2^{102} + \\
&s_2^{55}s_2^{56}s_2^{58}s_2^{59}s_2^{101} + s_2^{55}s_2^{56}s_2^{57}s_2^{58}s_2^{59} + s_0^{98}s_2^{55}s_2^{56}s_2^{58}s_2^{59} + s_0^{99}s_2^{55}s_2^{56}s_2^{57}s_2^{58} + \\
&s_0^{114}s_2^{57}s_2^{58}s_2^{102} + s_0^{114}s_2^{58}s_2^{59}s_2^{101} + s_0^{114}s_2^{57}s_2^{58}s_2^{59} + s_0^{114}s_2^{56}s_2^{58}s_2^{59} + s_0^{89}s_0^{90}s_2^{57}s_2^{58}s_2^{100} + \\
&s_0^{98}s_0^{114}s_2^{58}s_2^{59} + s_0^{99}s_0^{114}s_2^{57}s_2^{58} + s_0^{115}s_2^{56}s_2^{57}s_2^{101} + s_0^{115}s_2^{57}s_2^{58}s_2^{100} + s_0^{115}s_2^{56}s_2^{57}s_2^{58} + \\
&s_0^{115}s_2^{55}s_2^{57}s_2^{58} + s_0^{97}s_0^{115}s_2^{57}s_2^{58} + s_0^{98}s_0^{115}s_2^{56}s_2^{57} + s_0^{89}s_0^{90}s_2^{56}s_2^{57}s_2^{101} + \\
&s_0^{89}s_0^{90}s_2^{56}s_2^{57}s_2^{58} + s_0^{89}s_0^{90}s_2^{55}s_2^{57}s_2^{58} + s_0^{89}s_0^{90}s_0^{97}s_2^{57}s_2^{58} + s_0^{89}s_0^{90}s_0^{98}s_2^{56}s_2^{57}.
\end{aligned}
\tag{7}
$$

- **Step 5.** Here, there is no state bit in rounds more than *end* = 192, the expression ends and there are still state terms that survive. Then the current degree $d = 9$ is the estimated degree of s_2^{341}.
- **Step 6.** Note that, if there is no state item in s_2^{341***} surviving, which means the degree added twice or more shown in Eq. (6) happens to the iterative computation of s_2^{341}. So the degree must be less than 9. We reset $d = 8$ and continue the above steps 3–5 to get a more accurate degree bound.

We summarise the above 6 steps as Algorithm 2. We only estimate degree of s_i^r for $r \leq 665$ and list the results in Table 3.

Degree reduction technique. In this part, we formally consider the property in Eq. (6), that $\deg(b_{i-3})$ is added twice. We call it *degree reduction*. Define the degree reduction d_t as

$$d_t = \sum_{i \in I} \deg(x_i) - \deg(\prod_{i \in I} x_i), \tag{8}$$

where x_i is a state bit.

Algorithm 2. Degree Evaluation Algorithm ($\mathcal{DEG}$) of State Bit

Input: The value t and r which indicates the state bit s_t^r.
Output: $\mathcal{DEG}(s_t^r)$=d.

Initialize the degree bound d similar to the above **Step 2.**, the end point *end*.
$len \leftarrow 0$
while $len = 0$ **do**
    Iteratively express s_t^r using state bits s_i^j, where $0 \le j \le 2$ and $0 \le j < end$. During each expression, discard the state terms of degree lower than d. Let len be the number of remaining state terms.
    if $len = 0$ **then**
        $d \leftarrow d - 1$
    end if
end while
Return d

We pay attention to the degree reduction of the state term $\prod_{j=l}^{l+t-1} s_i^j$ for a specific $i \in [0, 2]$. This state term results from the iteration structure of Trivium scheme, whose high degree state terms come from the multiplication of $s_i^j s_i^{j+1}$ shown in Eq. (1). After several rounds of iteration, the high degree state terms are in the form $\prod_{j=l}^{l+t-1} s_i^j$. Define the degree reduction $d_t = \sum_{j=l}^{l+t-1} \deg(s_i^j) - \deg(\prod_{j=l}^{l+t-1} s_i^j)$.

Table 3. The estimated upper bound degree $\mathcal{DEG}(s_i^j)$ of s_i^r for $r \le 689$

$j+$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
$s_0^{j=340}$	6	6	6	6	6	6	6	6	6	6	7	7	8	9	9	9	9	8	6	7	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
$s_1^{j=340}$	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	8	8	8	8	7	6	7	8	8	8	8	8	8	8	8
$s_2^{j=340}$	8	9	9	9	9	9	9	9	10	10	10	10	10	9	10	10	10	10	10	10	10	10	10	10	11	11	11	11	11	11	10	10	11	12	13
$s_0^{j=375}$	9	9	9	9	9	10	10	11	11	11	11	11	11	9	10	11	11	11	12	13	14	15	15	15	15	15	13	11	14	15	15	15	15	15	15
$s_1^{j=375}$	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	9	9	9	9	9	9	8	8	9	9
$s_2^{j=375}$	13	13	13	12	12	13	13	13	12	12	13	13	13	13	13	13	13	13	12	13	13	12	13	13	13	13	13	13	13	13	12	13	13	13	13
$s_0^{j=410}$	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
$s_1^{j=410}$	9	9	9	9	9	9	9	9	10	12	12	12	12	10	10	11	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	13	14	15	17
$s_2^{j=410}$	13	13	13	12	12	12	13	13	13	13	13	13	13	12	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	14	14	14
$s_0^{j=445}$	15	15	15	15	15	15	16	16	16	16	16	16	16	16	16	16	16	16	16	17	18	18	18	18	19	18	18	19	19	19	19	19	19	19	18
$s_1^{j=445}$	18	18	18	17	14	12	15	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	19	20	20	20	20	20	20	20	20
$s_2^{j=445}$	14	14	12	12	13	14	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
$s_0^{j=480}$	18	19	20	22	22	22	22	22	21	21	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	21
$s_1^{j=480}$	19	20	20	20	21	22	23	24	24	24	24	24	24	21	22	24	24	24	25	25	25	25	25	25	25	26	26	26	26	26	26	26	26	26	26
$s_2^{j=480}$	15	15	16	16	17	17	17	17	16	15	16	17	17	17	17	17	17	17	17	17	18	20	21	21	21	20	18	18	20	21	21	21	21	21	21
$s_0^{j=515}$	21	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	23	23	23	23	23	23	23	24	24	24	24	24	24
$s_1^{j=515}$	26	26	26	26	26	26	27	27	27	27	27	27	27	26	27	28	29	29	29	29	29	28	29	29	29	29	29	30	31	31	31	31	31	32	32
$s_2^{j=515}$	21	21	21	21	21	21	21	21	22	23	24	26	27	28	28	29	28	25	27	29	29	29	28	29	29	29	30	31	32	33	33	33	33	33	33
$s_0^{j=550}$	24	25	26	26	26	26	25	23	25	26	27	27	27	27	27	27	27	26	27	28	29	29	29	29	29	28	29	29	29	29	28	29	29	29	29
$s_1^{j=550}$	32	32	32	32	32	33	35	35	35	35	35	35	33	34	35	37	37	37	37	37	37	36	36	38	40	41	41	41	41	40	40	41	41	41	40
$s_2^{j=550}$	31	32	33	34	35	36	37	37	37	36	36	36	35	36	36	36	37	38	39	40	40	40	40	40	40	40	40	44	45	45	45	45	45	45	45
$s_0^{j=585}$	29	29	29	29	29	29	30	31	31	31	31	31	31	30	30	31	31	31	31	31	31	31	31	31	32	34	36	37	37	37	35	34	36	37	37
$s_1^{j=585}$	40	41	41	41	41	41	41	41	41	40	41	41	40	41	41	41	41	41	41	41	41	40	41	41	41	41	42	42	42	41	40	41	42	42	42
$s_2^{j=585}$	45	45	45	45	45	45	42	42	42	42	42	42	42	43	44	44	44	44	45	45	46	46	46	48	47	46	46	48	48	48	48	48	49	49	48
$s_0^{j=620}$	38	38	41	42	42	42	42	41	38	39	42	42	43	45	47	50	53	54	54	54	53	49	45	51	54	54	54	54	54	55	56	56	56	56	56
$s_1^{j=620}$	42	42	42	42	42	43	44	44	44	45	45	44	45	45	45	45	45	45	45	45	44	45	46	49	50	50	50	50	47	46	48	51	52	52	52
$s_2^{j=620}$	49	49	49	49	50	51	51	51	50	51	52	54	54	54	54	54	54	54	56	58	58	58	59	59	59	59	60	62	62	62	62	62	62	60	59
$s_0^{j=655}$	56	56	56	56	56	56	56	57	60	62	64	64	64	64	64	64	63	61	63	64	65	67	70	72	73	73	73	73	74	74	69	72	74	74	75
$s_1^{j=655}$	52	52	52	52	52	52	52	52	52	52	52	52	52	52	52	52	52	52	52	52	51	52	52	52	52	52	53	54	57	59	61	62	62	62	59
$s_2^{j=655}$	61	66	68	69	69	69	69	68	68	69	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	71	71	71	71	71	71	71	69	69

The degree reduction can help discard state terms of lower degree dramatically, as it can help predict the change of degree before expression operation[2]. We take the state term $s_1^{340}s_1^{341}$ as an example to illustrate the process to compute the degree reduction d_t. Algorithm 2 is first used to obtain the degree of state bits as shown in Tables 2 and 3.

Let *end* be $\lfloor \frac{r}{32} \rfloor \times 32 - 128 = 192$, too. The degree bound d is initialized as $d = \mathcal{DEG}(s_1^{340}) + \mathcal{DEG}(s_1^{341})$ and $d_t = 0$. Express the $s_1^{340}s_1^{341}$ by one iteration using Eq. (1). Discard the state terms of degree lower than $d - d_t = d$, there is no state term surviving. Increase the d_t by 1, such that $d_t = 1$. Express $s_1^{340}s_1^{341}$ again and discard the state terms of degree lower than $d - d_t = d - 1$, the result is $s_0^{249}s_0^{250}s_1^{262} + s_0^{248}s_0^{249}s_1^{263}$. Continue to compute iteratively, the remaining state terms are $s_0^{170}s_0^{171}s_0^{180}s_2^{140}s_2^{141} + s_0^{170}s_0^{171}s_0^{181}s_2^{139}s_2^{140} + s_0^{171}s_0^{172}s_0^{179}s_2^{139}s_2^{140} + s_0^{171}s_0^{172}s_0^{180}s_2^{138}s_2^{139}$. There is no state bits s_i^j with j bigger than $end = 192$ in all the state terms, hence the expression ends. Degree reduction $d_t = 1$ is returned. Thus the $\deg(s_1^{340}s_1^{341}) \leq \mathcal{DEG}(s_1^{340}) + \mathcal{DEG}(s_1^{341}) - d_t = 7 + 7 - 1 = 13$. The degree reduction algorithm is shown in Algorithm 3

Algorithm 3. Degree Reduction Algorithm of State Term

Input: The value i, r, t which indicates the state term degree reduction.
Output: The degree reduction $d_t = \sum_{j=l}^{l+t-1} \deg(s_i^j) - \deg(\prod_{j=l}^{l+t-1} s_i^j)$.

1: Initialize the degree bound $d = \sum_{i=l}^{l+t-1} \mathcal{DEG}(s_i^j)$, degree reduction $d_t = 0$, end point *end* and number of survived state terms *len*.
2: **while** $len = 0$ **do**
3: Express the state term $\prod_{j=l}^{l+t-1} s_i^j$ using state bits s_i^j, where $0 \leq i \leq 2$ and $0 \leq j < end$, discard the state terms of degree lower than $d - d_t$. Let *len* be the number of remaining state terms.
4: **if** $len = 0$ **then**
5: $d_t \leftarrow d_t + 1$
6: **end if**
7: **end while**
8: **Return** d_t

3.5 IV Representation Techniques

In the cryptanalysis of stream ciphers, the output is a boolean function over key and IV bits. But obtaining the exact expression is hard, thus we propose *IV representation* technique to reduce the computation complexity for obtaining the degree information.

Definition 1. *(IV representation) Given a state bit* $s = \sum_{I,J} \prod_{i \in I} v_i \prod_{j \in J} k_j$, *the IV representation of* s *is* $s_{IV} = \sum_I \prod_{i \in I} v_i$.

For example, if a boolean polynomial is $s = v_0k_1 + v_0k_0k_2 + v_1k_1k_2 + v_0v_1k_2$, then its corresponding IV representation is $s_{IV} = v_0 + v_0 + v_1 + v_0v_1$.

---

[2] The details are given in Sect. 4.2.

IV representation with repeated IV terms Removing Algorithm. Due to neglection of key bits, there are lots of repeated IV terms. Here we give an algorithm to remove the repeated IV terms of s_{IV}. The details of the algorithm are shown in Algorithm 4. This algorithm is based on a Hash function. First, an empty hash set is initialized. For each IV term T_i, compute the hash value as $H(T_i)$ (Line 3), then determine if T_i is already in H. If not, then insert T_i into H (Lines 4–5). Applying Algorithm 4 to the above example, the result is $v_0 + v_1 + v_0v_1$. Note that this algorithm is slightly different from Algorithm 1. If we apply Algorithm 1 to s_{IV}, the result is $v_1 + v_0v_1$.

In the iterative computation process of the output bit of Trivium, it should be noted that if an IV term exists in s, it must also exist in s_{IV}, but not the opposite. For example, $x_1 = v_0(k_1k_2 + k_0k_2) + v_1 + v_0v_1k_2$, $x_2 = v_2k_0k_1 + v_1v_2k_1$ and $s = x_1x_2$. We use the IV representations of x_1 and x_2 to approximate the IV representation of s. Thus, $x_{1IV} = v_0 + v_1 + v_0v_1$, $x_{2IV} = v_2 + v_1v_2$, and $s_{IV} = x_{1IV}x_{2IV} = v_0v_2 + v_1v_2 + v_0v_1v_2$. However, $s = x_1x_2 = v_1v_2(k_0k_1 + k_1)$. ***So if we find an IV term is not in*** s_{IV}, ***we can conclude that it is not in*** s ***either***. We use this to determine the degree upper bound of the output ANF of Trivium.

Algorithm 4. Repeated-IV term Removing Algorithm

Input: The vector $\boldsymbol{T}$ with n IV terms, i.e., $T_1, T_2, \ldots, T_n$.
Output: Updated $\boldsymbol{T}$ with m IV terms, where $m \leq n$.

```
1: Initialize an empty Hash set H.
2: for i ← 1 : n do
3:    Compute the Hash value of T_i, i.e., H(T_i).
4:    if H.contains(T_i) is false then
5:       H.insert(T_i).
6:    end if
7: end for
```

After using IV representation combined with Algorithm 4, all the existent IV terms are left by ignoring their repetition. With collision-resistent hash function H, the time complexity of Algorithm 4 is $O(n)$ for processing n IV terms. It needs several minutes to apply Algorithm 4 on 1 billion IV terms on a single core.

4 Key Recovery Attack on 855-round Trivium

In the attack on 855-round Trivium, all the 80-bit IV are initiated with free variables: $IV_i = v_i$, $i \in [0, 79]$.

The output of 855-round Trivium can be described using the internal state bits:

$$z_{855} = s_0^{790} + s_0^{763} + s_1^{787} + s_1^{772} + s_2^{790} + s_2^{745}. \tag{9}$$

As a first step of the attack on 855-round Trivium, we need to determine P_1.

4.1 Determining the Nullification Scheme for the Output Polynomial of 855-round Trivium

For 855-round Trivium, the degree of output bit z is very high, as shown in [19]. So it is not easy to find the missing IV terms in the complex $z = P_1P_2 + P_3$. However, based on the new observation of Boolean polynomial introduced in Sect. 3.1. we can choose P_1 to reduce the Boolean polynomial $(1 + P_1)z = (1 + P_1)P_3$ such that the degree of $(1 + P_1)P_3$ is lower. The lower, the better. In fact, the lower the degree of a state term, the less high degree IV terms it can deduce.

Degrees of state bits are obtained first in order to determine the high degree state terms. The exact Boolean polynomial of s_i^j for $i \in [0, 2]$ and $j \in [0, 340]$ can be obtained. The other degree upper bounds can be obtained by executing Algorithm 2.

For a search of P_1, we use the decomposition of Trivium and preserve the high degree state terms (bigger than a given bound dependent on our computing ability in a PC), where the degree of state terms means the sum of degrees of each state bit in the earlier rounds involved. We decompose until all the state bits are within the range of $[0, 276]$. The key points to determine P_1 come from 3 criteria: (1) the frequency of P_1 is high; (2) the degree of P_1 is low; (3) the equivalent key guesses in P_1 are minimized. We calculate the frequency of state bits and find that s_1^{210} occurs in about $\frac{3}{4}$ of all the preserved high state terms. The degree of s_1^{210} is 5 and can be reduced to 2 after nullifying the 5 IV bits, and there are only 3 equivalent key bits to be guessed. So we choose $P_1 = s_1^{210}$.

The output polynomial can be rewritten as

$$z = s_1^{210}P_2 + P_3, \tag{10}$$

where P_2 and P_3 do not contain s_1^{210}. Polynomial P_2 is so complex that it is hard to compute its degree and density information while P_3 is relatively simple. Here $P_1 = s_1^{210} = v_{59}v_{60}v_{61} + v_{59}v_{60}v_{76} + v_{17}v_{59}v_{60} + v_{30}v_{31}v_{59}v_{60} + v_{32}v_{59}v_{60} + v_{59}v_{60}v_{62} + v_{59}v_{60}v_{77} + v_{59}v_{60}k_{20} + v_{59}v_{61}v_{73}v_{74} + v_{59}v_{73}v_{74}v_{76} + v_{17}v_{59}v_{73}v_{74} + v_{30}v_{31}v_{59}v_{73}v_{74} + v_{32}v_{59}v_{73}v_{74} + v_{59}v_{62}v_{73}v_{74} + v_{59}v_{73}v_{74}v_{77} + v_{59}v_{73}v_{74}k_{20} + v_{59}v_{60}v_{74}v_{75} + v_{59}v_{60}v_{75}v_{76} + v_{59}v_{73}v_{74}v_{75} + v_{59}v_{73}v_{74}v_{75}v_{76} + v_{59}v_{61}v_{75} + v_{59}v_{74}v_{75} + v_{17}v_{59}v_{75} + v_{30}v_{31}v_{59}v_{75} + v_{32}v_{59}v_{75} + v_{59}v_{62}v_{75} + v_{59}v_{75}v_{77} + v_{59}v_{75}k_{20} + v_{60}v_{61}v_{72}v_{73} + v_{60}v_{72}v_{73}v_{76} + v_{17}v_{60}v_{72}v_{73} + v_{30}v_{31}v_{60}v_{72}v_{73} + v_{32}v_{60}v_{72}v_{73} + v_{60}v_{62}v_{72}v_{73} + v_{60}v_{72}v_{73}v_{77} + v_{60}v_{72}v_{73}k_{20} + v_{61}v_{72}v_{73}v_{74} + v_{72}v_{73}v_{74}v_{76} + v_{17}v_{72}v_{73}v_{74} + v_{30}v_{31}v_{72}v_{73}v_{74} + v_{32}v_{72}v_{73}v_{74} + v_{62}v_{72}v_{73}v_{74} + v_{72}v_{73}v_{74}v_{77} + v_{72}v_{73}v_{74}k_{20} + v_{60}v_{72}v_{73}v_{74}v_{75} + v_{60}v_{72}v_{73}v_{75}v_{76} + v_{72}v_{73}v_{74}v_{75}v_{76} + v_{61}v_{72}v_{73}v_{75} + v_{17}v_{72}v_{73}v_{75} + v_{30}v_{31}v_{72}v_{73}v_{75} + v_{32}v_{72}v_{73}v_{75} + v_{62}v_{72}v_{73}v_{75} + v_{72}v_{73}v_{75}v_{77} + v_{72}v_{73}v_{75}k_{20} + v_{60}v_{61}v_{74} + v_{60}v_{74}v_{76} + v_{17}v_{60}v_{74} + v_{30}v_{31}v_{60}v_{74} + v_{32}v_{60}v_{74} + v_{60}v_{62}v_{74} + v_{60}v_{74}v_{77} + v_{60}v_{74}k_{20} + v_{17}v_{73}v_{74} + v_{30}v_{31}v_{73}v_{74} + v_{32}v_{73}v_{74} + v_{62}v_{73}v_{74} + v_{73}v_{74}v_{77} + v_{73}v_{74}k_{20} + v_{16}v_{60}v_{61} + v_{16}v_{60}v_{74}v_{75} + v_{16}v_{60}v_{76} + v_{16}v_{61}v_{73}v_{74} + v_{16}v_{73}v_{74}v_{75} + v_{16}v_{73}v_{74}v_{76} + v_{16}v_{61}v_{75} + v_{16}v_{74}v_{75} + v_{16}v_{17} + v_{16}v_{30}v_{31} + v_{16}v_{32} + v_{16}v_{62} + v_{16}v_{77} + v_{16}k_{20} + v_{29}v_{30}v_{60}v_{61} + v_{29}v_{30}v_{60}v_{74}v_{75} + v_{29}v_{30}v_{60}v_{76} + v_{29}v_{30}v_{61}v_{73}v_{74} + v_{29}v_{30}v_{73}v_{74}v_{75} + v_{29}v_{30}v_{73}v_{74}v_{76} + v_{29}v_{30}v_{61}v_{75} + v_{29}v_{30}v_{74}v_{75} + v_{17}v_{29}v_{30} + v_{29}v_{30}v_{31} + v_{29}v_{30}v_{32} + v_{29}v_{30}v_{62} + v_{29}v_{30}v_{77} + v_{29}v_{30}k_{20} + v_{31}v_{60}v_{61} + v_{31}v_{60}v_{74}v_{75} + v_{31}v_{60}v_{76} +$

$v_{31}v_{61}v_{73}v_{74} + v_{31}v_{73}v_{74}v_{75} + v_{31}v_{73}v_{74}v_{76} + v_{31}v_{61}v_{75} + v_{31}v_{74}v_{75} + v_{17}v_{31} + v_{30}v_{31} + v_{31}v_{62} + v_{31}v_{77} + v_{31}k_{20} + v_{60}v_{61} + v_{61}v_{75} + v_{61}v_{74}v_{75} + v_{17}v_{61} + v_{30}v_{31}v_{61} + v_{32}v_{61} + v_{61}k_{20} + v_{60}v_{74}v_{75}v_{76} + v_{60}v_{76} + v_{73}v_{74}v_{75}v_{76} + v_{17}v_{76} + v_{30}v_{31}v_{76} + v_{32}v_{76} + v_{76}v_{77} + v_{76}k_{20} + v_{60}v_{61}k_{19} + v_{60}v_{74}v_{75}k_{19} + v_{60}v_{76}k_{19} + v_{61}v_{73}v_{74}k_{19} + v_{73}v_{74}v_{75}k_{19} + v_{73}v_{74}v_{76}k_{19} + v_{61}v_{75}k_{19} + v_{74}v_{75}k_{19} + v_{17}k_{19} + v_{30}v_{31}k_{19} + v_{32}k_{19} + v_{62}k_{19} + v_{77}k_{19} + k_{19}k_{20} + v_{34}v_{35} + v_{34}v_{48}v_{49} + v_{34}v_{50} + v_{35}v_{47}v_{48} + v_{47}v_{48}v_{49} + v_{47}v_{48}v_{50} + v_{35}v_{49} + v_{48}v_{49} + k_{57} + v_{69} + v_4v_5 + v_6 + v_{36} + v_{51} + v_{60} + v_{73}v_{74} + v_{75} + k_{63} + v_{62}v_{74}v_{75} + v_{74}v_{75}v_{77} + v_{75}v_{76} + v_{18} + v_{33} + v_{63} + v_{78} + k_{21} + k_{28}k_{29} + k_3 + k_{30} + k_{12} + k_{37}k_{38} + k_{39} + v_{24}$.

IV Nullification. The degree of s_1^{210} is 5 and the IV bits involved in s_1^{210} are shown in Table 4.

Table 4. Count of IV bits in s_1^{210} before IV nullification.

IV	v_4	v_5	v_6	v_{16}	v_{17}	v_{18}	v_{24}	v_{29}	v_{30}	v_{31}	v_{32}	v_{33}	v_{34}	v_{35}	v_{36}	v_{47}	v_{48}
Count	1	1	1	14	14	1	1	14	27	26	13	1	3	3	1	3	5
IV	v_{49}	v_{50}	v_{51}	v_{59}	v_{60}	v_{61}	v_{62}	v_{63}	v_{69}	v_{72}	v_{73}	v_{74}	v_{75}	v_{76}	v_{77}	v_{78}	
Count	4	2	1	28	44	26	13	1	1	26	56	62	46	26	14	1	

In order to simplify s_1^{210} so that it is easier to obtain the degree bound of $(1 + s_1^{210})P_3$, we nullify v_{74}, v_{60}, v_{75}, v_{30} and v_{48}.

After nullifying the 5 IV bits, we obtain the simplified boolean function:

$$\begin{aligned}s_1^{210} = {} & v_{16}v_{17} + v_{16}v_{32} + v_{16}v_{62} + v_{16}v_{77} + v_{16}k_{20} + v_{17}v_{31} + v_{31}v_{62} + \\ & v_{31}v_{77} + v_{31}k_{20} + v_{17}v_{61} + v_{32}v_{61} + v_{61}k_{20} + v_{17}v_{76} + v_{32}v_{76} + v_{76}v_{77} + \\ & v_{76}k_{20} + v_{17}k_{19} + v_{32}k_{19} + v_{62}k_{19} + v_{77}k_{19} + k_{19}k_{20} + v_{34}v_{35} + v_{34}v_{50} + \\ & v_{35}v_{49} + k_{57} + v_{69} + v_4v_5 + v_6 + v_{36} + v_{51} + k_{63} + v_{18} + v_{33} + v_{63} + \\ & v_{78} + k_{21} + k_{28}k_{29} + k_3 + k_{30} + k_{12} + k_{37}k_{38} + k_{39} + v_{24}.\end{aligned} \tag{11}$$

Here, the degree of s_1^{210} is 2 and key information equivalent to 3 bits in s_1^{210} are k_{19}, k_{20} and $k_{57} + k_{63} + k_{21} + k_{28}k_{29} + k_3 + k_{30} + k_{12} + k_{37}k_{38} + k_{39}$. The IV bits involved in s_1^{210} are shown in Table 5.

After determining $P_1 = s_1^{210}$, we multiply $1 + s_1^{210}$ in both sides of Eq. (10), then $(1 + s_1^{210})z = (1 + s_1^{210})P_3$. Finding the non-randomness in $(1 + s_1^{210})P_3$ will help us to construct the cube tester of 855-round Trivium. More specifically, we will determine the nonexistent IV terms of degree 70 in $(1+s_1^{210})P_3$. First, we will reduce the polynomial, then IV presentation technique is applied to determine the nonexistent IV terms. The framework is presented in Fig. 2 and details are shown in the following Sect. 4.2.

Table 5. Frequency of IV bits in s_1^{210} after IV nullification.

IV	v_4	v_5	v_6	v_{16}	v_{17}	v_{18}	v_{24}	v_{31}	v_{32}	v_{33}	v_{34}	v_{35}
Count	1	1	1	5	5	1	1	4	4	1	2	2
IV	v_{36}	v_{49}	v_{50}	v_{51}	v_{61}	v_{62}	v_{63}	v_{69}	v_{76}	v_{77}	v_{78}	
Count	1	1	1	1	3	3	1	1	4	4	1	

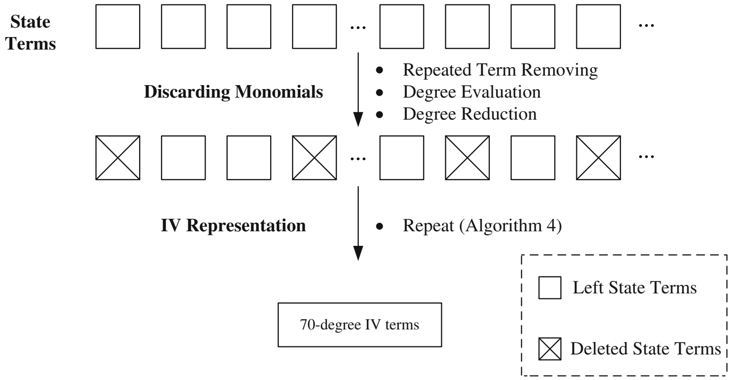

Fig. 2. Framework of determining the missing IV terms

4.2 Determining the Degree Bound of Reduced Polynomial

We are going to iteratively compute $(1 + s_1^{210})P_3$. In each iteration, many state terms of $(1 + s_1^{210})P_3$ are produced. Based on our computing ability, we can compute the IV terms of degree around 70. In computing the 70-degree IV terms, we use a cluster of 600–2400 cores. Since we are finding the 70-degree missing IV terms, state terms with degree less than 70 are removed without consideration, because they do not contain those 70-degree IV terms certainly. The removing process could be divided into 2 steps:

1. Deleting state terms according to degree evaluation;
2. Deleting state terms according to degree reduction.

Degree evaluation phase. After nullifying the 5 IV bits in Sect. 4.1, the exact boolean functions and degrees of state bits s_i^j for $0 \le i \le 2$ and $0 \le j \le 340$ can be updated. Then we execute Algorithm 2 to obtain the degrees of the other state bits, partially in Tables 2 and 3. For example, given a state term $b_1 b_2$, we first find $\mathcal{DEG}(b_1)$ and $\mathcal{DEG}(b_2)$ in Tables 2 and 3, if $\mathcal{DEG}(b_1) + \mathcal{DEG}(b_2) < 70$, then $\deg(b_1 b_2) \le \mathcal{DEG}(b_1) + \mathcal{DEG}(b_2) < 70$, delete $b_1 b_2$.

Degree reduction phase. In the structure of stream ciphers based on NFSR, degree reduction arises often due to the iterative structure. We use Algorithm 3 to obtain the degree reduction, which is shown in Tables 6, 7 and 8 for products of 2 consecutive state bits $s_i^j s_i^{j+1}$ ($t = 2$), 3 consecutive state bits $s_i^j s_i^{j+1} s_i^{j+2}$ ($t = 3$) and 4 consecutive state bits $s_i^j s_i^{j+1} s_i^{j+2} s_i^{j+3}$ ($t = 4$), respectively. Note that we only list the degree reduction when $j \geq 340$. The degree reduction for $j < 340$ is much easier to obtain in a PC.

Table 6. Degree reductions $d_t(s_i^j s_i^{j+1})$ of $s_i^j s_i^{j+1}$ with $t = 2$

$j+$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$j = 340, i = 0$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$j = 340, i = 1$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	2	2	2	2	2	2	2	2	1	1	1	1	1	1
$j = 340, i = 2$	1	3	3	3	3	3	3	2	4	4	4	4	4	4	4	4	4	4	4	3	2	2	2	1	3	3	3	3	3	3	3	3	2	2
$j = 374, i = 0$	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	1	2	3	4	6	6	6	6	6	6	6	6	6	6	5	5
$j = 374, i = 1$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
$j = 374, i = 2$	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
$j = 408, i = 0$	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	1	1	1
$j = 408, i = 1$	1	1	1	1	1	1	1	1	1	0	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3
$j = 408, i = 2$	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2
$j = 442, i = 0$	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	2	1	1	2	2	2	1	3	3	3	3	3	3	3
$j = 442, i = 1$	4	4	7	8	8	7	7	7	7	7	7	7	7	7	7	7	6	5	4	3	3	3	3	3	3	3	3	3	2	3	4	4	4	4
$j = 442, i = 2$	3	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$j = 476, i = 0$	3	3	1	1	1	0	0	3	3	3	3	3	3	3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
$j = 476, i = 1$	4	4	4	4	4	4	4	3	4	5	6	8	8	8	8	8	8	8	8	8	8	7	9	9	9	9	9	9	8	10	10	10	10	10
$j = 476, i = 2$	1	1	1	1	1	0	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	5	7	7	7	7	7	7	7	7
$j = 510, i = 0$	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	3	3	3	3	2	1	1
$j = 510, i = 1$	10	10	10	10	10	9	8	8	8	8	7	9	8	8	8	8	8	8	8	7	8	10	10	10	9	9	9	9	9	9	9	10	11	11
$j = 510, i = 2$	7	6	4	3	3	3	3	3	3	3	3	3	2	3	4	4	7	8	10	9	11	11	11	11	11	11	11	11	11	11	10	9	9	9
$j = 544, i = 0$	3	3	3	3	3	3	2	1	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	4	6	6	6	6	6	6	6
$j = 544, i = 1$	11	11	11	10	10	10	10	10	10	10	9	9	12	12	12	11	10	10	10	9	10	12	12	12	12	12	12	12	11	10	12	13	13	13
$j = 544, i = 2$	11	11	11	11	11	11	11	11	10	9	8	8	10	10	10	8	8	8	8	8	8	7	7	6	6	7	7	7	7	6	6	6	16	16
$j = 578, i = 0$	6	6	6	6	6	6	6	6	6	6	6	5	4	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$j = 578, i = 1$	13	13	13	13	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	11	11	11	11	11	11	11	11	11	11	11	11	10	12
$j = 578, i = 2$	16	15	15	15	15	15	15	15	15	15	15	15	15	9	9	9	9	9	9	9	10	12	12	12	12	13	12	14	14	14	18	18	17	16
$j = 612, i = 0$	0	0	0	0	0	0	0	0	0	0	3	5	5	6	6	6	6	6	6	5	4	5	5	7	11	12	11	10	10	5	5	5	5	5
$j = 612, i = 1$	12	12	12	12	12	12	12	11	10	10	10	9	11	12	12	12	11	13	13	13	12	10	9	9	9	8	7	7	7	6	5	10	12	12
$j = 612, i = 2$	16	13	12	12	11	13	13	13	13	13	13	12	13	15	15	15	14	13	12	16	16	16	16	16	15	15	18	20	19	18	17	16	16	15
$j = 646, i = 0$	5	5	5	6	6	4	2	2	4	4	4	4	4	4	4	3	3	7	8	12	12	12	11	11	10	10	10	8	7	7	9	13	16	18
$j = 646, i = 1$	12	12	12	12	11	9	9	9	9	9	9	9	9	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
$j = 646, i = 2$	16	18	18	18	18	18	18	18	17	15	17	19	21	21	21	21	21	21	20	22	21	20	20	20	20	20	20	19	18	18	18	18	18	18

In the cryptanalysis of Trivium, the degree reduction may be more complicated. Further degree reduction for $t > 4$ is hard to be obtained using PC for loop executing Algorithm 3. Some man-made work should be involved to obtain further degree reduction. The degree reduction can help discard state terms of lower degree dramatically. For example, if the state term $b_1 b_2$ goes through degree evaluation phase, that means $\mathcal{DEG}(b_1) + \mathcal{DEG}(b_2) \geq 70$, then we check if $\mathcal{DEG}(b_1) + \mathcal{DEG}(b_2) - d_t(b_1 b_2) < 70$. If yes, $\deg(b_1 b_2) < 70$ and delete it.

For example, the Eq. (9) can be expressed furthermore using state bits: $z_{855} = s_2^{724} + s_2^{680} s_2^{681} + s_2^{679} + s_0^{721} + s_2^{697} + s_2^{653} s_2^{654} + s_2^{652} + s_0^{694} + s_0^{721} + s_0^{695} s_0^{696} + s_0^{694} + s_1^{709} + s_0^{706} + s_0^{680} s_0^{681} + s_0^{679} + s_1^{694} + s_1^{721} + s_1^{707} s_1^{708} + s_1^{706} + s_2^{703} + s_1^{676} + s_1^{662} s_1^{663} + s_1^{661} + s_2^{658}$. Then s_2^{652}, s_1^{676}, s_1^{661}, can be discarded because their degree are lower than 68, shown in Table 3 highlighted in red, and the total degree of the multiplication of each one with $(1 + s_{210})$ is lower than 70. In addition, the state terms highlighted in blue can be discarded by removing the repeated state terms. Furthermore, the output can be expressed using state bits in lower rounds and more state terms can be discarded.

Table 7. Degree reductions $d_t(s_i^j s_i^{j+1} s_i^{j+2})$ of $s_i^j s_i^{j+1} s_i^{j+2}$ with $t = 3$

$j+$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$j = 340, i = 0$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$j = 340, i = 1$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	2	2	2	2	2	2	2	2	1	1	1	1	1	1
$j = 340, i = 2$	0	3	3	3	3	3	2	2	4	4	4	4	4	4	4	4	4	4	4	3	2	2	1	1	3	3	3	3	3	3	3	2	0	1
$j = 374, i = 0$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	3	5	5	5	5	5	5	5	5	5	5	5	5
$j = 374, i = 1$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0
$j = 374, i = 2$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$j = 408, i = 0$	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3	2	1	1	1
$j = 408, i = 1$	0	0	0	0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	1	2
$j = 408, i = 2$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	1	1	1	1	1	1	1	0
$j = 442, i = 0$	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	2	2	1	1	3	3	3	3	3	3	3
$j = 442, i = 1$	2	3	7	8	8	7	7	7	7	7	7	7	7	7	7	7	6	5	4	3	3	3	3	3	3	3	3	2	1	2	4	4	4	4
$j = 442, i = 2$	2	2	2	2	2	2	2	1	1	3	3	3	3	3	3	3	3	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$j = 476, i = 0$	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	0
$j = 476, i = 1$	4	4	4	4	4	4	3	2	3	4	6	8	8	8	8	8	8	8	8	8	7	7	9	9	9	9	9	8	8	10	10	10	10	10
$j = 476, i = 2$	1	1	1	1	0	0	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	0	1	5	7	7	7	7	7	7	7	7
$j = 510, i = 0$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	1	1	3	3	3	3	0	0	0
$j = 510, i = 1$	10	10	10	10	10	9	8	8	8	7	7	9	8	7	7	7	7	7	6	5	7	9	9	9	9	9	9	9	8	8	7	9	11	10
$j = 510, i = 2$	7	6	4	3	3	3	3	3	3	3	3	2	1	2	2	3	7	8	9	9	11	11	11	11	11	11	11	11	11	9	7	7	7	9
$j = 544, i = 0$	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	2	5	5	5	5	5	5	5	5
$j = 544, i = 1$	10	10	10	9	10	10	10	10	10	9	7	8	12	11	10	9	9	9	8	6	8	11	11	11	11	11	11	10	7	7	10	11	11	11
$j = 544, i = 2$	11	11	11	11	11	11	11	10	8	7	6	6	7	7	7	4	4	4	4	4	3	1	1	0	0	1	0	0	0	0	0	0	15	15
$j = 578, i = 0$	5	5	5	5	5	5	5	5	5	5	5	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$j = 578, i = 1$	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	10	10	12
$j = 578, i = 2$	15	15	15	15	15	15	15	15	15	15	15	15	15	6	6	6	6	6	6	5	8	10	10	10	9	11	10	12	12	12	18	18	17	16

After the above 2 steps to reduce $(1 + s_1^{210})P_3$, the degrees of the left state terms are possibly higher or equal to 70. As the dimension is high, a cube tester over such a big dimension is far beyond our computing ability. For the left state terms, we use IV representation for each left state terms and remove the repeated IV terms using Algorithm 4 in order to determine the missing 70-degree IV terms. After the above steps, there is no 70-degree IV term in $(1 + s_1^{210})P_3$. So the degree of $(1 + s_1^{210})P_3$ is strictly lower than 70, which is summarized as the following Lemma 2.

Lemma 2. *Set the v_{74}, v_{60}, v_{75}, v_{30} and v_{48} to zeros, then the degree of $(1 + s_1^{210})z_{855}$ is bounded by 70, where z_{855} is the output after 855-round initializations.*

According to Lemma 2, we strictly prove that the degree of the reduced polynomial is lower than 70, so the sum over any selected cube of dimension 70 is zero, such that the distinguishers can be constructed.

Table 8. Degree reductions $d_t(s_i^j s_i^{j+1} s_i^{j+2} s_i^{j+3})$ of $s_i^j s_i^{j+1} s_i^{j+2} s_i^{j+3}$ with $t = 4$

$j+$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$j = 340, i = 0$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$j = 340, i = 1$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	0	4	4	4	4	4	4	3	3	2	2	2	2	2	2
$j = 340, i = 2$	2	6	6	6	6	5	5	4	8	8	8	8	8	8	8	8	8	7	6	5	4	3	3	2	6	6	6	6	6	6	5	3	2	4
$j = 374, i = 0$	0	0	0	0	0	0	0	0	4	4	4	4	4	4	4	4	3	2	0	2	5	8	12	12	12	12	12	12	12	12	11	11	10	10
$j = 374, i = 1$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	0	0	2	2	2	2	2	2	2	2
$j = 374, i = 2$	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
$j = 408, i = 0$	10	10	10	10	9	8	8	8	8	8	8	8	8	8	8	8	8	8	8	7	7	6	6	6	6	6	6	6	5	4	3	2	2	2
$j = 408, i = 1$	2	2	2	2	2	2	2	1	0	0	0	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	5	4	2	3
$j = 408, i = 2$	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	7	7	6	6	6	6	6	5	5	4	4	4	4	4	4	3	2
$j = 442, i = 0$	2	2	2	2	2	1	0	0	0	2	2	2	1	0	0	0	0	0	0	0	1	0	1	4	3	3	2	6	6	6	6	6	6	6
$j = 442, i = 1$	5	7	13	15	15	14	14	14	14	14	14	14	14	14	13	12	10	8	7	6	6	6	6	6	6	6	5	4	2	4	8	8	8	8
$j = 442, i = 2$	5	4	4	4	4	4	4	3	3	6	6	6	6	6	6	5	4	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

4.3 Online Phase and Complexity Analysis

We first guess the 3 key bits in s_1^{210}, i.e. k_{19}, k_{20} and $k_{57} + k_{63} + k_{21} + k_{28}k_{29} + k_3 + k_{30} + k_{12} + k_{37}k_{38} + k_{39}$ as shown in Eq. (11), for the right guess the result is 0 while for wrong guesses, the result is 1 with probability $\frac{1}{2}$. If the sum over cubes of dimension 70 is 1, then the key guess is wrong and dropped (Line 7). After the first cube sum, about half key bits remain, and sum over another cube again. The remaining guess is the key. The on-line phase is shown in Algorithm 5.

Algorithm 5. On-line Attack

```
1: Initialize the possible key space KEY with size of 2^3.
2: for i ← 1 : 3 do
3:    for Each possible key in KEY do
4:       Compute the value s_1^210, so that obtain the value of (1 + s_1^210)z,
5:       Compute cube sums z_sum of (1 + s_1^210)z,
6:       if z_sum = 1 then
7:          Delete key from KEY.
8:       end if
9:    end for
10: end for
```

For each guess, we need to sum over a cube of dimension 70, so that the complexity is $2^3 \cdot 2^{70} + 2^2 \cdot 2^{70} + 2^1 \cdot 2^{70} \approx 2^{74}$.

After the above process, the bits k_{19}, k_{20} and $k_{57} + k_{63} + k_{21} + k_{28}k_{29} + k_3 + k_{30} + k_{12} + k_{37}k_{38} + k_{39}$ can be determined. k_{19} and k_{20} are single master key bits. Let $c = k_{57} + k_{63} + k_{21} + k_{28}k_{29} + k_3 + k_{30} + k_{12} + k_{37}k_{38} + k_{39}$ (c is 0 or 1), then it can be rewritten as $k_{57} = k_{63} + k_{21} + k_{28}k_{29} + k_3 + k_{30} + k_{12} + k_{37}k_{38} + k_{39} + c$. We guess the other 77 key bits excluding k_{19}, k_{20} and k_{57}, the value k_{57} can be obtained directly. So the other 77 key bits excluding k_{19}, k_{20} and k_{57} can be recovered by brute force. Thus the complexity to recover all the key bits is 2^{77}.

4.4 Experimental Verification

We apply a powerful nullification technique to reduce the output polynomial, and prove the degree bound of the reduced polynomial theoretically and recover key bits. To make the attack more clear, we give an attack instance. We give two attacks on 721-round Trivium: a distinguishing attack and a key-recovery attack.

Obtain the Degree Upper Bound of Output of 721-round Trivium. Initial $IV_i = v_i$ with $i \in [0, 79]$. In the example attack on 721-round Trivium, we only use 40 freedom variables, i.e. set $v_{2 \cdot j+1} = 0$ for $j \in [0, 39]$ and the other 40 IV bits are freedom variables.

The exact boolean functions of the first 340 state bits s_i^j for $i \in [0,2]$ and $j \in [0,340]$ can be obtained directly on PC. Hence, the degrees of them can be obtained directly. Degrees upper bounds of other state bits can be evaluated using Algorithm 2 and are shown in Table 9. Note that in Table 9, the estimated degrees of some state bits are larger than 40, e.g. $\mathcal{DEG}(s_2^{665}) = 41$, which is because the accuracy of Algorithm 2 decreases for state bits with large rounds. Thus we only apply this algorithm to s_i^j for $j \le 665$.

The output of 721-round Trivium is $z_{721} = s_0^{656} + s_0^{629} + s_1^{653} + s_1^{638} + s_2^{656} + s_2^{611}$. According to Table 9, the 6 state terms (bits) highlighted in red are of degree lower than 40, so the degree of z_{721} is lower than 40, which can serve as distinguishers. This result can be obtained easily by rough computing.

Next, we give a more accurate bound of z_{721}. In the following, we will determine whether z_{721}'s degree is bigger than 37. The 6 state bits are expressed using state bits in lower rounds again and substituted into z_{721}, which is called the substitution or expression process in [9]. Then $z_{721} = s_2^{590} + s_2^{546}s_2^{547} + s_2^{545} + s_0^{587} + s_2^{563} + s_2^{519}s_2^{520} + s_2^{518} + s_0^{560} + s_0^{587} + s_0^{561}s_0^{562} + s_0^{560} + s_1^{575} + s_0^{572} + s_0^{546}s_0^{547} + s_0^{545} + s_1^{560} + s_1^{587} + s_1^{573}s_1^{574} + s_1^{572} + s_2^{569} + s_1^{542} + s_1^{528}s_1^{529} + s_1^{527} + s_2^{524}$. According to degree upper bounds Table 9, $\deg(s_2^{590}) = 27 < 37$ highlighted in blue, so s_2^{590} is removed. Then $\deg(s_2^{546}s_2^{547}) \le \mathcal{DEG}(s_2^{546}) + \mathcal{DEG}(s_2^{547}) = 20 + 21 = 41$ and $41 \ge 37$, so the degree of $s_2^{546}s_2^{547}$ is possibly bigger than 36 and left. After discarding all the state terms whose degrees are lower than 36, $z_{721}|_{\deg>36} = s_2^{546}s_2^{547} + s_1^{573}s_1^{574}$. Continue substitution and expression process for $z_{721}|_{\deg>36}$ and finally, there remain no state terms with degree bigger than 36, so that the degree bound of z_{721} is 36. The details of the above step are shown in Appendix A.

A Key-Recovery Attack on 721-round Trivium. Similar to the IV setting above for distinguishing 721-round Trivium, we set $v_{2\cdot j+1} = 0$ for $j \in [0,39]$ and the other 40 IV bits are freedom variables.

According to our attack outline introduced in Sect. 3.2, we need to determine the nullification scheme first. We express the output of 721-round Trivium iteratively and calculate the frequency of state bits in the polynomial. Then we choose s_1^{290} as P_1, the output can be rewritten as $z_{721} = s_1^{290}P_2 + P_3$. Multiply $1 + s_1^{290}$ with z_{721} such that the result is $(1 + s_1^{290})z_{721} = (1 + s_1^{290})P_3$. We study the reduced polynomial $(1 + s_1^{290})P_3$. In order to decrease the number of key bits in s_1^{290}, we choose to nullify v_{58}, v_{64} and v_{72}, so that there are 37 freedom variables. Set the degree bound to 32, we express $(1+s_1^{290})P_3$ using internal state bits furthermore and discard state terms whose degree are lower than $32 + d_t$, where d_t is the corresponding degree reduction. We use IV presentation, combined with Algorithm 4 in order to obtain the IV terms of degree higher than 32. Finally, there is no IV term. Hence, we prove that the degree of $(1 + s_1^{290})z_{721}$ is lower than 32. Then the sum of $(1+s_1^{290})z_{721}$ over any selected cube of dimension 32 is zero. This process can be executed in an hour in a PC.

Table 9. Degree upper bounds $\mathcal{DEG}(s_i^j)$ of the state bits s_i^j for $j \leq 665$

$j+$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
$s_0^{j=0}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$s_1^{j=0}$	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$s_2^{j=0}$	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s_0^{j=35}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
$s_1^{j=35}$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
$s_2^{j=35}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s_0^{j=70}$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s_1^{j=70}$	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$s_2^{j=70}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s_0^{j=105}$	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_1^{j=105}$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1
$s_2^{j=105}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s_0^{j=140}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_1^{j=140}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2
$s_2^{j=140}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s_0^{j=175}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_1^{j=175}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3
$s_2^{j=175}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s_0^{j=210}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_1^{j=210}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$s_2^{j=210}$	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_0^{j=245}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
$s_1^{j=245}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4
$s_2^{j=245}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
$s_0^{j=280}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3
$s_1^{j=280}$	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
$s_2^{j=280}$	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
$s_0^{j=315}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
$s_1^{j=315}$	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
$s_2^{j=315}$	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
$s_0^{j=350}$	4	4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7
$s_1^{j=350}$	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
$s_2^{j=350}$	5	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
$s_0^{j=385}$	7	7	7	7	7	7	7	7	7	7	7	7	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
$s_1^{j=385}$	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
$s_2^{j=385}$	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
$s_0^{j=420}$	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	10	10
$s_1^{j=420}$	7	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	9	11	11	11	11	11	11	11	11	11
$s_2^{j=420}$	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
$s_0^{j=455}$	10	10	10	10	10	10	10	10	10	10	10	11	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	13	13	13	13	13	13
$s_1^{j=455}$	11	11	11	11	11	11	11	11	12	12	12	12	12	12	12	12	12	12	12	12	13	13	13	13	13	13	13	13	13	13	13	13	13	14	15
$s_2^{j=455}$	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	9	10	10	10	10	10
$s_0^{j=490}$	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
$s_1^{j=490}$	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	16	16	16	16	16	16	16	16	16	16	16	16	16	16	17	17
$s_2^{j=490}$	10	10	10	10	10	10	10	10	10	10	10	10	11	13	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
$s_0^{j=525}$	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	14	14	14	14	14	14	14	14	14	14	14	14
$s_1^{j=525}$	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	18	19	19	19	19	20	21	21
$s_2^{j=525}$	14	14	15	17	18	18	18	19	19	19	19	19	19	19	19	19	19	19	19	19	**20**	**21**	21	21	21	21	21	21	21	21	21	21	22	23	23
$s_0^{j=560}$	14	14	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	16	18
$s_1^{j=560}$	21	21	21	21	21	21	22	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23
$s_2^{j=560}$	23	23	23	23	23	23	23	23	23	23	24	25	25	25	25	25	27	27	27	27	27	27	27	27	27	27	27	27	27	**27**	27	26	26	26	26
$s_0^{j=595}$	19	19	19	19	20	20	20	20	20	20	20	20	20	20	20	20	21	23	24	24	24	24	24	24	24	24	24	24	25	27	27	27	27	**27**	27
$s_1^{j=595}$	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	24	24	24	24	24	24	24	24	24	24	24	24	24	25	26	26	26
$s_2^{j=595}$	26	26	26	26	26	26	26	26	26	26	27	28	29	29	29	**29**	29	29	29	29	30	30	30	30	30	30	30	30	30	30	30	31	31	31	31
$s_0^{j=630}$	27	27	27	27	27	27	28	31	33	33	33	34	34	34	34	34	34	34	34	34	34	34	34	34	36	**37**	37	37	37	37	37	37	37	37	37
$s_1^{j=630}$	26	26	26	26	26	26	26	**26**	26	27	28	28	28	28	28	28	28	28	28	28	28	28	**28**	29	29	29	29	29	29	29	29	30	30	30	30
$s_2^{j=630}$	31	31	31	31	32	33	33	33	33	34	36	37	37	37	37	37	37	37	38	39	39	39	39	39	39	**39**	39	40	41	41	41	41	41	41	41

Guess the key bit involved in s_1^{290}. For right guess, sum over a cube of dimension 32 is zero while for wrong guesses, the result is 1 with probability $\frac{1}{2}$. The key bits involved in s_1^{290} are shown in Table 10. After 19 summations over cubes of dimension 32, the 19 key bits can be recovered. The complexity is about $2 \times 2^{19} \times 2^{32} = 2^{52}$. The other key bits can be recovered using brute force with a complexity of 2^{61}. Hence, the total complexity of recovering all key bits of 721-round Trivium is 2^{61}.

Table 10. The key bits involved in s_1^{290}.

Equivalent key bits
$k_{18}, k_{17}, k_{63}, k_{61}, k_{59}, k_{60} + k_{16}k_{17}, k_{35} + k_{60}k_{61} + k_{62}, k_{33} + k_{58}k_{59} + k_{60}, k_{15} + k_{40}k_{41} + k_{42}, k_{42}k_{43} + k_{44}, k_{48} + k_{73}k_{74} + k_{75} + k_{61}k_{62}, k_{47} + k_{72}k_{73} + k_{74} + k_{60}k_{61} + k_{62}, k_{46} + k_{71}k_{72} + k_{73} + k_{59}k_{60},\ k_{45} + k_{70}k_{71} + k_{72} + k_{58}k_{59} + k_{60},\ k_{34}k_{35} + k_{34}k_{60}k_{61} + k_{34}k_{62} + k_{35}k_{59}k_{60} + k_{59}k_{60}k_{61} + k_{59}k_{60}k_{62} + k_{35}k_{61} + k_{60}k_{61} + k_{21} + k_{46}k_{47} + k_{48} + k_{36},\ k_{33}k_{34} + k_{33}k_{59}k_{60} + k_{33}k_{61} + k_{34}k_{58}k_{59} + k_{58}k_{59}k_{60} + k_{58}k_{59}k_{61} + k_{34}k_{60} + k_{59}k_{60} + k_{20} + k_{45}k_{46} + k_{47} + k_{35} + k_{62}, k_{16}k_{17} + k_{16}k_{42}k_{43} + k_{16}k_{44} + k_{17}k_{41}k_{42} + k_{41}k_{42}k_{43} + k_{41}k_{42}k_{44} + k_{17}k_{43} + k_{42}k_{43} + k_3 + k_{28}k_{29} + k_{30} + k_{45} + k_{48} + k_{73}k_{74} + k_{75} + k_{61}k_{62} + k_9, k_{15}k_{16} + k_{15}k_{41}k_{42} + k_{15}k_{43} + k_{16}k_{40}k_{41} + k_{40}k_{41}k_{42} + k_{40}k_{41}k_{43} + k_{16}k_{42} + k_{41}k_{42} + k_2 + k_{27}k_{28} + k_{29} + k_{44} + k_{47} + k_{72}k_{73} + k_{74} + k_{60}k_{61} + k_{62}, k^*(\textit{A complex expression of key bits})$.

5 Conclusions

In this paper, we propose the Boolean polynomial reduction techniques and IV representation, which can be applicable to cryptanalysis of stream ciphers based on NFSRs. These techniques can help obtain more accurate degree bounds. We apply these techniques to the cryptanalysis of reduced round Trivium. For recovering the key bits of Trivium, we propose a new nullification technique. Combined with the distinguishers, we propose a key-recovery attack on 855 round Trivium, where 3 equivalent key bits can be recovered with complexity of 2^{74}. The other key bits can be recovered by brute force with a complexity of 2^{77}.

Furthermore, our flexible methods can be applied to attack more round of Trivium by adjustment of P_1, which is our future work. In addition, the degree evaluation and degree reduction techniques can be applicable to other encryption primitives such as Grain family.

Acknowledgement. The authors would like to thank anonymous reviewers for their helpful comments. We also thank National Supercomputing Center in Wuxi for their support of Sunway TaihuLight, which is the most powerful supercomputer. This work was supported by the National Key Research and Development Program of China (Grant No. 2017YFA0303903), and National Cryptography Development Fund (No. MMJJ20170121), and Zhejiang Province Key R&D Project (No. 2017C01062).

References

1. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03317-9_1
2. De Cannière, C., Preneel, B.: Trivium. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_18
3. Dinur, I., Güneysu, T., Paar, C., Shamir, A., Zimmermann, R.: An experimentally verified attack on full grain-128 using dedicated reconfigurable hardware. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 327–343. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_18
4. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_16
5. Dinur, I., Shamir, A.: Breaking grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_10
6. Englund, H., Johansson, T., Sönmez Turan, M.: A framework for chosen IV statistical analysis of stream ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77026-8_20
7. Fischer, S., Khazaei, S., Meier, W.: Chosen IV statistical analysis for key recovery attacks on stream ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68164-9_16
8. Fouque, P.-A., Vannet, T.: Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 502–517. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_26
9. Fu, X., Wang, X., Chen, J.: Determining the nonexistent terms of non-linear multivariate polynomials: how to break grain-128 more efficiently. IACR Cryptology ePrint Archive 2017, 412 (2017). http://eprint.iacr.org/2017/412
10. International Organization for Standardization (ISO): ISO/IEC 29192–3:2012, Information technology - Security techniques - Lightweight cryptography - Part 3: Stream ciphers (2012)
11. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of NLFSR-based cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_8
12. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of Trivium and KATAN. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 200–212. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28496-0_12
13. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60590-8_16
14. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45661-9_9

15. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography, pp. 227–233. Springer, Boston (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
16. Liu, M.: Degree evaluation of NFSR-based cryptosystems. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 227–249. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_8
17. Liu, M., Yang, J., Wang, W., Lin, D.: Correlation cube attacks: from weak-key distinguisher to key recovery. Cryptology ePrint Archive, Report 2018/158 (2018). https://eprint.iacr.org/2018/158
18. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_12
19. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_9
20. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_2
21. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_2

A The Details of Determining the Degree Upper Bound of Output for 721-round Trivium

For $z_{721}|_{\deg>36} = s_2^{546}s_2^{547} + s_1^{573}s_1^{574}$, the 4 state bits s_2^{546}, s_2^{547}, s_1^{573}, s_1^{574} can be expressed using state bits furthermore. Substitute the 4 state bits using the expression and discard the state terms whose degree is lower than 37, then the resulted $z_{721}|_{\deg>36} = s_1^{463}s_1^{464}s_1^{478} + s_1^{464}s_1^{465}s_1^{477} + s_0^{481}s_0^{482}s_0^{508} + s_0^{482}s_0^{483}s_0^{507} + s_0^{482}s_0^{483}s_1^{495} + s_0^{481}s_0^{482}s_1^{496}$. Then the state bits involved in the polynomial can be expressed using state bits, so that we can obtain $z_{721}|_{\deg>36} = s_0^{412}s_2^{372}s_2^{373}s_2^{398}s_2^{399} + s_0^{413}s_2^{371}s_2^{372}s_2^{398}s_2^{399} + s_0^{413}s_2^{373}s_2^{374}s_2^{397}s_2^{398} + s_0^{414}s_2^{372}s_2^{373}s_2^{397}s_2^{398} + s_0^{403}s_0^{404}s_2^{372}s_2^{373}s_2^{417} + s_0^{403}s_0^{404}s_2^{373}s_2^{374}s_2^{416} + s_0^{403}s_0^{404}s_2^{372}s_2^{373}s_2^{374} + s_0^{403}s_0^{404}s_2^{371}s_2^{373}s_2^{374} + s_0^{403}s_0^{404}s_0^{413}s_2^{373}s_2^{374} + s_0^{403}s_0^{404}s_0^{414}s_2^{372}s_2^{373} + s_0^{404}s_0^{405}s_2^{371}s_2^{372}s_2^{416} + s_0^{404}s_0^{405}s_2^{372}s_2^{373}s_2^{415} + s_0^{404}s_0^{405}s_2^{371}s_2^{372}s_2^{373} + s_0^{404}s_0^{405}s_2^{370}s_2^{372}s_2^{373} + s_0^{404}s_0^{405}s_0^{412}s_2^{372}s_2^{373} + s_0^{404}s_0^{405}s_0^{413}s_2^{371}s_2^{372}$.

Repeat the process above and we can obtain $z_{721}|_{\deg>36} = s_1^{290}s_1^{291}s_1^{305}s_2^{293}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{291}s_1^{292}s_1^{304}s_2^{293}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{290}s_1^{291}s_1^{292}s_2^{293}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{289}s_1^{291}s_1^{292}s_2^{293}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{291}s_1^{292}s_2^{286}s_2^{293}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{290}s_1^{291}s_2^{287}s_2^{293}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{290}s_1^{291}s_1^{305}s_2^{292}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{291}s_1^{292}s_1^{304}s_2^{292}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{290}s_1^{291}s_1^{292}s_2^{292}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{289}s_1^{291}s_1^{292}s_2^{292}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{291}s_1^{292}s_2^{286}s_2^{292}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{290}s_1^{291}s_2^{287}s_2^{292}s_2^{294}s_2^{295}s_2^{303}s_2^{304} + s_1^{289}s_1^{290}s_1^{304}s_2^{293}s_2^{294}s_2^{295}s_2^{304}s_2^{305} + s_1^{290}s_1^{291}s_1^{303}s_2^{293}s_2^{294}s_2^{295}s_2^{304}s_2^{305} + s_1^{289}s_1^{290}s_1^{291}s_2^{293}s_2^{294}s_2^{295}s_2^{304}s_2^{305} + s_1^{288}s_1^{290}s_1^{291}s_2^{293}s_2^{294}s_2^{295}s_2^{304}s_2^{305} + s_1^{290}s_1^{291}s_2^{285}s_2^{293}s_2^{294}s_2^{295}s_2^{304}s_2^{305} + s_1^{289}s_1^{290}s_2^{286}s_2^{293}s_2^{294}s_2^{295}s_2^{304}s_2^{305} + s_1^{289}s_1^{290}s_1^{304}s_2^{292}s_2^{294}s_2^{295}s_2^{304}s_2^{305} +$

$s_1^{290} s_1^{291} s_1^{303} s_2^{292} s_2^{294} s_2^{295} s_2^{304} s_2^{305} + s_1^{289} s_1^{290} s_1^{291} s_2^{292} s_2^{294} s_2^{295} s_2^{304} s_2^{305} + s_1^{288} s_1^{290} s_1^{291} s_2^{292} s_2^{294} s_2^{295} s_2^{304} s_2^{305} + s_1^{290} s_1^{291} s_2^{285} s_2^{292} s_2^{294} s_2^{295} s_2^{304} s_2^{305} + s_1^{289} s_1^{290} s_2^{286} s_2^{292} s_2^{294} s_2^{295} s_2^{304} s_2^{305} + s_1^{289} s_1^{290} s_1^{304} s_2^{294} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{290} s_1^{291} s_1^{303} s_2^{294} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{289} s_1^{290} s_1^{291} s_2^{294} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{288} s_1^{290} s_1^{291} s_2^{294} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{290} s_1^{291} s_2^{285} s_2^{294} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{289} s_1^{290} s_2^{286} s_2^{294} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{289} s_1^{290} s_1^{304} s_2^{293} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{290} s_1^{291} s_1^{303} s_2^{293} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{289} s_1^{290} s_1^{291} s_2^{293} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{288} s_1^{290} s_1^{291} s_2^{293} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{290} s_1^{291} s_2^{285} s_2^{293} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{289} s_1^{290} s_2^{286} s_2^{293} s_2^{295} s_2^{296} s_2^{302} s_2^{303} + s_1^{288} s_1^{289} s_1^{303} s_2^{294} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{289} s_1^{290} s_1^{302} s_2^{294} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{288} s_1^{289} s_1^{290} s_2^{294} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{287} s_1^{289} s_1^{290} s_2^{294} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{289} s_1^{290} s_2^{284} s_2^{294} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{288} s_1^{289} s_2^{285} s_2^{294} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{288} s_1^{289} s_1^{303} s_2^{293} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{289} s_1^{290} s_1^{302} s_2^{293} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{288} s_1^{289} s_1^{290} s_2^{293} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{287} s_1^{289} s_1^{290} s_2^{293} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{289} s_1^{290} s_2^{284} s_2^{293} s_2^{295} s_2^{296} s_2^{303} s_2^{304} + s_1^{288} s_1^{289} s_2^{285} s_2^{293} s_2^{295} s_2^{296} s_2^{303} s_2^{304}.$

Substitute once again and there remains no state term, so that the degree of z_{721} is lower than 37, which can be derived as distinguishers with lower complexity.