Linear Extension Cube Attack on Stream Ciphers

Liren Ding Yongjuan Wang Zhufeng Li
(Language Engineering Department, Luo yang University for Foreign Language, Luo yang city, He nan Province, 471003, P. R. China)

Abstract: Basing on the original Cube attack, this paper proposes an improved method of Cube attack on stream ciphers, which makes improvement on the pre-processing phase of the original attack. The new method can induce maxterms of higher-order from those of lower-order by the trade-off between time and space, thus recovering more key bits and reducing the search complexity on higher-dimension. In this paper, the improved attack is applied to Lili-128 algorithm and reduced variants of Trivium algorithm. We can recover 88 key bits of Lili-128 algorithm within time complexity of $O(2^{14})$ and 48 key bits of Trivium algorithm can be recovered by cubes with dimension no larger than 8 when the initialization round is 576, the results are much better than those of the original attacks.

Keywords: Cube Attack, Stream Cipher, Linear Extension, Pre-processing, Trivium, Lili-128

1. Introduction

In 2008, Dinur and Shamir[1] introduced a deterministic analysis method basing on algebraic theory, i.e. Cube Attack. Cube attack is a kind of chosen plaintext attack, it makes use of the fact that the output of a cryptosystem can be represented as a polynomial of public variables and key bits. Theoretically, Cube attack can be applied to all cryptosystems as long as they can be represented as tweakable polynomials. The algebraic degree of the master polynomial can be decreased by choosing arbitrary values for tweakable variables. By doing so, the attacker can obtain linear equations about the key bits. Hence, the security of a cryptosystem is reduced to the problem of solving a linear equation system. Although the specific representation of a cryptosystem is unknown, the attack can still be applied to a "black box". The performance of Cube attack against stream ciphers, block ciphers and hash functions is quite well[2,3,4].Therefore, Cube attack has achieved lots of attention since its announcement.

However, there being few limitations of Cube attack. First of all, the algebraic degree of a certain cryptosystem should not be too large, otherwise it would be difficult to search for linear expressions on the pre-processing phase. As a result, the performance of original Cube attack against NFSR based stream ciphers and block ciphers is limited. Second, choosing the right cubes is time-consuming. If the choice is not proper or the search has not been done thoroughly, the final result of Cube attack would be weakened.

With the research of Cube attack going deeper, cryptologists have come up with several extended versions of Cube attack against different algorithms. In 2010, Dinur and Shamir[5] proposed Dynamic Cube attack, which is based on the analysis of the internal structure of a certain cipher so that the proper choice for the values of such dynamic variables can lower the algebraic degree of the master polynomial. P. Mroczkowki[6] improved the linearity test of Cube attack by extracting quadratic equations. Yu Sun[7] made improvements on the search algorithm and linearity test so that more than one linear expressions can be extracted from only one cube with the help of the trade-off between time and space.

Our work is based on the phenomenon pointed out by P. Mroczkowki in [6] that although the cubes were chosen randomly, there was a way to obtain new cubes from another one with linear expression. In fact, for stream ciphers with shift register, whose transitivity enables the lower-round to influence the higher-round so that d and $d+1$ dimensional cubes will have some variables in common. Therefore, more cubes can be derived from one with linear expression, thus obtaining more linear expressions.

Motivated by the above observations, this paper proposes an improved attack on stream ciphers on the basis the original Cube attack, i.e. Linear Extension Cube Attack, which makes use of those common variables in two different dimensional cubes and the trade-off between time and space enables the attacker to induce maxterms of higher-order from those of lower-order, thus recovering more key bits and reducing the the search complexity on higher-dimension.

To demonstrate the application of the extended method, this correspondence provides Linear Extension Cube Attack against Lili-128 algorithm and two reduced variants of Trivium algorithm and the results are much better than those of the original attacks. For Lili-128 algorithm, only $O(2^9)$ key streams are needed to recover 88 key bits and the attack has time complexity less than $O(2^{14})$, which is better compared to $O(2^{16})$ of the original Cube attack. And it is the best attack against Lili-128 to the best of our knowledge. For Trivium algorithm, when the initialization round is 525, the original Cube attack on the 525th output bit with 4 dimensional cubes can only recover 6 key bits, while the improved attack can directly recover 31 key bits. When the initialization round is 576, applying the improved Cube attack to the 576th output bit, 48 key bits can be recovered by cubes with dimension no larger than 8, while the original Cube attack on several output bits in [14] can only recover 36 key bits by cubes with dimension no larger than 8.

The organization of this paper is as follows, in section 2, the preliminaries and basic steps of Cube attack are reviewed. Section 3 contains the main contribution of this paper, where the notion of Linear Extension Cube Attack and its details are provided. To testify the performance of the improved attack, it is applied to Lili-128 and two reduced variants of Trivium algorithm in section 4. At last, section 5 is a brief summary of this paper.

2. Review on Cube Attack

Cube attack regards the investigated cryptosystem as a polynomial $P(v,k)$ about the public variables $v=(v_1,...,v_m)$ and secret key bits $k=(x_1,...,x_n)$. Let $I=\{i_1,...,i_k\}\subseteq\{1,2,...,n\}$ be the indexes of the public variables and $t_I=v_{i_1}\cdots v_{i_k}$ be the product of these public variables, through factoring the master polynomial by the monomial t_I, we have:

$$\begin{aligned}&P(v_1,...,v_m,x_1,...,x_n)\\&=t_I\cdot P_{S(I)}+Q(v_1,...,v_m,x_1,...,x_n).\end{aligned}$$

where $P_{S(I)}$, which is called the superpoly of t_I, does not have any common variables with t_I, and each monomial term in the residue polynomial $Q(v_1,...,v_m,x_1,...,x_n)$ misses at least one variable from t_I.

Denote $C_I=\{(v_1,...,v_m)\in F_2^m \mid v_i\in F_2, i\in I; v_i=0, i\notin I\}$ as a ***Cube***, apparently , we have $|C_I|=2^{|I|}$. Summing the polynomials when the public variables $v=\{v_1,...,v_m\}$ walking over the cube, we have:

$$\sum_{(v_1,...,v_m)\in C_I} P(v,k) = \sum_{(v_1,...,v_m)\in C_I} t_I \cdot P_{S(I)} + \sum_{(v_1,...,v_m)\in C_I} Q(v_1,...,v_m,x_1,...,x_n)\,.$$

The feasibility of Cube attack mainly depends on the following observations:

Theorom 1[1] For any polynomial $P(v,k)$ and public variable $v=(v_1,...,v_m)$, we have $\underset{v\in C_I}{\Sigma} P(v,k) = P_{S(I)}(v,k) \bmod 2$.

Proof: Each monomial term in the residue polynomial of $P(v,k)$,that is, $Q(v_1,...,v_m,x_1,...,x_n)$ misses at least one variable from t_I, the value of each term from $Q(v,k)$ after 2^k times computation is 0, thus $\underset{v\in C_I}{\Sigma} Q(v,k)=0$. Next, if and only if $v_{i_1},...,v_{i_k}$ are all set to 1, the coefficient of $P_{S(I)}(v,k)$ would be 1. #

A term t_I is called a ***maxterm*** if its superpoly is a linear polynomial while not a constant.

Example 1:

Let $P(v_1,v_2,v_3,x_1,x_2,x_3) = v_1v_2x_1 + v_1v_2x_3 + v_1v_3x_1 + v_1v_2 + x_2 + 1$, where (v_1,v_2,v_3) being the public variables and (x_1,x_2,x_3) being the key. And define *I*={1,2}, through factoring the master polynomial by the monomial t_I, we have:

$$P(v_1,v_2,v_3,x_1,x_2,x_3) = v_1v_2(x_1+x_3+1) + v_1v_3x_1 + x_2 + 1\,.$$

$C_I = \{(0,0,0),(1,0,0),(0,1,0),(1,0,0)\}$, by summing the polynomial over this cube, we have:

$$\sum_{v\in C_I} P(v_1,v_2,v_3,x_1,x_2,x_3) = P_{S(I)}(v_3,x_1,x_2,x_3) = x_1 + x_3 + 1\,.$$

Cube attack consists of two phases, the preprocessing phase and the on-line phase. During the pre-processing phase, the attacker can arbitrarily assign values to both the public variables and the key bits, and choose appropriate cubes to do the computation. Then the linearity test is applied to test whether the superpoly is linear or not. The main purpose of this phase is to extract linear expressions about the key bits as many as possible. During the on-line phase, the attacker can only control the public variables, through which he can conduct the cube sum over the same cube in order to obtain the value of the right side of a certain expression. The main purpose of this phase is to establish linear equations. Due to the selection of different cubes, he can establish a system of linear equations. At last, the key bits are recovered by solving the equation system.

3. Linear Extension Cube Attack

3.1 The Main Observation

When it comes to the shift register, a higher-position of a lower-round will become a lower-position of a higher-round with the extension of round itself. Therefore, the existence of common variables between the master polynomials before and after the extension is certain. Using these common variables to construct new cubes, the attacker can analysis the target cryptosystem of higher rounds on the basis of the attack results of a lower round. Motivated by the above observation, this paper proposes Linear Extension Cube Attack against stream ciphers. The main idea of this extended attack is to make improvement on the pre-processing phase of the original attack. Thanks to the transitivity of the shift register, by the trade-off between time and space, maxterms of degree $d+1$ can be derived from those maxterms of degree d, thus obtaining more linear expressions so that more key bits can be recovered.

To demonstrate the main contribution, definitions and examples are given as follows.

Definition 1 Let $N=\{1,2,...,n\}$, $I=\{i_1,i_2,\cdots,i_k\}$, $I\subset U$, $|I|=k$, define the maxterm as $t_I=v_{i_1}v_{i_2}\cdots v_{i_k}$, through factoring the master polynomial by t_I, we have: $P=t_IP_{S(I)}+Q$. Let $t_{I+1}=t_Iv_s$, where $s\in N\setminus I$, $P^{'}=t_{I+1}P_{S(I+1)}+Q^{'}$ denotes the master polynomial after extension, monomial t_{I+1} is called the **1-time extended maxterm** of t_I if $\deg(P_{S(I+1)})=1$.

Example 2:

$$\begin{aligned}&P(v_1,v_2,v_3,v_4,x_1,x_2,x_3,x_4)\\&=v_1v_2v_3x_4+v_1v_2v_3+v_2v_3x_1\\&+v_1v_3x_2+v_1v_2x_3+x_1x_2\\&=v_1v_2v_3(x_4+1)+v_2v_3x_1\\&+v_1v_3x_2+v_1v_2x_3+x_1x_2 \qquad ①\end{aligned}$$

$$\begin{aligned}&P^{'}(v_1,v_2,v_3,v_4,x_1,x_2,x_3,x_4)\\&=v_1v_2v_3v_4x_1+v_1v_2v_3+v_2v_3x_1\\&+v_1v_3x_2+v_1v_2x_3+x_1x_2 \qquad ②\end{aligned}$$

$P^{'}$ denotes the master polynomial after extension, let $I=\{1,2,3\}$, we have maxterm as $t_I=v_1v_2v_3$ according to ①, let $I^{'}=\{1,2,3,4\}$, then $t_{I^{'}}=t_{I+1}=v_1v_2v_3v_4$ is also a maxterm according to ②, so t_{I+1} is called the 1-time extended maxterm of t_I.

Note that, to conduct the Linear Extension Cube Attack, the target cryptosystem should be extended first so that the algebraic degree of master polynomial would increase. For extension here, it implies one round as well as several rounds, the existence of 1-time extended maxterm holds true for both conditions. The following of this section discusses the existence of 1-time extended maxterm under the extension of round.

Theorom 2 $P(x)$ denotes the master polynomial of a cryptosystem, through factoring $P(x)$ by maxterm t_I, we have: $P(x)=t_IP_{S(I)}+Q$, $P^{'}(x)$ denotes the polynomial after extension by one round, then the 1-time extended maxterm of t_I is existed by the possibility of 1 if t_I satisfies that $P^{'}(x)=t_IP^{'}{}_{S(I)}+Q^{'}$, where $\deg(P^{'}{}_{S(I)})=2$, and can be represented as $v_a\cdot l(x)+g(x)$, where $\deg(l(x))=1$ and $a\notin I$.

Proof: Since t_I is a maxterm of $P(x)$, we have: $P(x)=t_IP_{S(I)}+Q$. Assuming that $t_{I+1}=t_I\cdot v_a$ is a 1-time extended maxterm of t_I, where $a\notin I$ according to definition 1, then we have $P^{'}(x)=t_{I+1}P^{'}{}_{S(I)}+Q^{'}$, where $\deg(P^{'}{}_{S(I)})=1$. Obviously, the following equations hold true:

$$P^{'}(x)=t_I\cdot v_aP^{'}{}_{S(I)}+Q^{'} \qquad ③$$

$$P^{'}(x)=t_IP^{''}{}_{S(I)}+Q^{''} \qquad ④$$

Note that, $t_I\cdot v_a\nmid Q^{'}$, $t_I\nmid Q^{''}$ and $t_I\mid t_I\cdot v_a$, then we have:

$$Q^{'}=t_Ig(x)+Q^{''} \qquad ⑤$$

Subsitute ⑤ into ③, then combine it with ④, we have:

$$
\begin{aligned}
t_I P''_{S(I)} + Q'' &= t_I \cdot v_a P'_{S(I)} + Q' \\
&= t_I \cdot v_a P'_{S(I)} + t_I g(x) + Q'' \\
&= t_I (v_a P'_{S(I)} + g(x)) + Q''
\end{aligned}
$$

In conclusion, $P''_{S(I)} = v_a \cdot P'_{S(I)} + g(x)$ meets the criteria for the existence of t_{I+1}. #

Corollary 1 $P(x)$ denotes the master polynomial of a cryptosystem and t_I denotes one of its maxterms, let $P'(x)$ be the polynomial after the extension of $r(1<r<n)$ round, then the 1-time extended maxterm of t_I is existed by the possibility of 1 if t_I satisfies the criteria in Theorom 2.

Proof: According to the proof of Theorom 2, the times of extension would not make any differences on the existence of t_{I+1}. #

Note that, although the existence of 1-time extended maxterm is not strictly related to the times of extension, it is impossible to find maxterm through increasing the cubes by one when the algebraic degree of master polynomial goes too large after several rounds of extension.

Example 3：

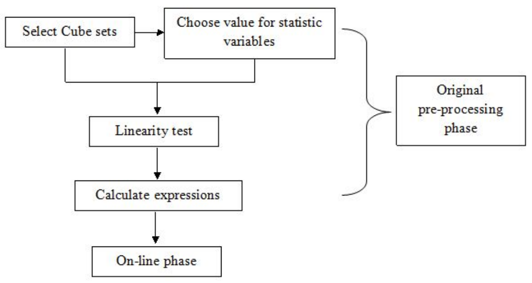
$$
\begin{aligned}
P(x_1,\ldots,x_5) &= x_1x_2x_3 + x_2x_3x_4 + x_1x_4x_5 \\
&= x_2x_3(x_1 + x_4) + x_1x_4x_5
\end{aligned}
$$

$$
\begin{aligned}
P' = P(x_2,\ldots,x_5,x_6) &= x_3x_4(x_2+x_5)+x_2x_5x_6 \\
&= x_3x_4(x_2+x_5)+x_2x_5(x_1x_2x_3+x_2x_3x_4+x_1x_4x_5) \\
&= x_3x_4(x_2+x_5)+x_1x_2x_3x_5+x_2x_3x_4x_5+x_1x_2x_4x_5 \\
&= x_2x_3x_4(1+x_5)+x_3x_4x_5+x_1x_2x_3x_5+x_1x_2x_4x_5 \\
&= x_2x_3[x_4(x_1+x_5)+x_1x_5]+x_3x_4x_5+x_1x_2x_4x_5
\end{aligned}
$$

P' denotes the master polynomial after extension, note that $t_I = x_2x_3$ is a maxterm of P, there exists $t_{I+1} = t_I \cdot x_4$, which satisfies the criteria in Theorom 2, being the 1-time extended maxterm of t_I.

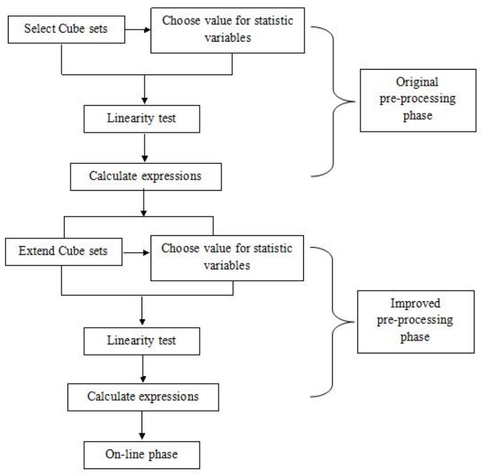
3.2 Attack Procedure

Compared to the original Cube attack, the Linear Extension Cube Attack makes improvement on the pre-processing phase. Cube attack consists of two phases, pre-processing phase and on-line phase. The improved attack, which makes use of the cubes with linear expressions, consists of the original pre-processing phase, the improved pre-processing phase and on-line phase. The following figures reveal the difference between the two versions of Cube attack:



Select Cube sets
Choose value for statistic variables
Original pre-processing phase
Linearity test
Calculate expressions
On-line phase

Figure 1 Procedures of Original Cube Attack



Select Cube sets
Choose value for statistic variables
Original pre-processing phase
Linearity test
Calculate expressions
Extend Cube sets
Choose value for statistic variables
Improved pre-processing phase
Linearity test
Calculate expressions
On-line phase

Figure 2 Procedures of Linear Extension Cube Attack

As is shown above, the details of the improved attack display as follows:

Phase 1, the original pre-processing phase. In this phase, the attacker can arbitrarily choose values for both public variables and key bits in order to find proper cubes.

(1) Choose the initial dimension as d and walk over all of the d dimensional cubes;

(2) Sum over each cubes and use linearity test to find maxterms;

(3) Calculate algebraic normal forms for each linear expression.

Phase 2, the improved pre-processing phase. The attack can search $d+1$ dimensional cubes on the basis of the results of the original pre-processing phase.

(1) Store all of the d dimensional cubes obtained via phase 1;

(2) Extend the investigated cryptosystem by extension of round or choosing the next output bit;

(3) Extend each maxterm of degree d respectively. Firstly, modify each variable of the d dimensional cube according to the regulation of the target algorithm. The modification here means increase by one (or r), decrease by one (or r) or stay the same for each variable. Secondly, add a new variable from the public variables to set I to form a new cube;

(4) Sum over the new $d+1$ dimensional cube and use linearity test to find maxterms;

(5) Calculate algebraic normal forms for each linear expression;

(6) Testify the linear dependence of linear expressions by Gaussian elimination.

Phase 3, the on-line phase. According to the cubes obtained by phase 1 and phase 2, the attacker can choose values for the public variables to obtain the value of the right side of each expression. At last, the key bits are recovered by solving the linear equations system.

Special attention should be paid to the following steps:

(1) The extension of the targeted algorithm is necessary after walking over all of the d dimensional cubes since there is no 1-time extended maxterms under the same polynomial;

(2) There is no need for Gaussian elimination after walking over all of the d dimensional cubes since linearly independent expressions can be derived from linearly dependent ones. Therefore, Gaussian elimination is introduced after the extension in the improved attack.

The pseudo-code of the improved pre-processing phase is as follows:

Algorithm 1：The Improved Pre-processing Phase

```
Input：$V$ ; // set of public variables, $v_i$ denotes each variable
       $T$ ; // set of maxterms of degree $d$ , $I_i$ denotes each maxterm
       $R$ ; // number of round extension
       $r=0$ ; // initialization of round extension
Output：linear expressions
repeat
    $r++$ ;
    repeat
        Choose an $I_i$ which has not been chosen from $T$ ;
        repeat
            Modify each variable in $I_i$ ;
            Choose a $v_i$ which has not been chosen from $V$ , i.e. $I_i^{'} = I_i \cup \{v_i\}$ ;
            Sum over $I_i^{'}$ ;
            Introduce linearity test;
        until
            Walk over $V$ ;
    until
        Walk over $T$ ;
until
```

```
    $r = R$;
```

Figure 3 The Pseudo-code of the Improved Pre-processing Phase

4. The Application of Linearity Extension Cube Attack

To demonstrate the application of the extended method, this correspondence provides Linear Extension Cube Attack against Lili-128 algorithm and two reduced variants of Trivium algorithm when the initialization round is 525 and 576 by using the personal computer equipped with i-5 CPU, 1.7GHz dominant frequency and 2GB RAM.

4.1 Attack 1: Lili-128

4.1.1 Brief Introduction of Lili-128 Algorithm

Lili-128[10] is one of the candidate algorithms of NESSIE. The clock-controlled algorithm is consists of two linear feedback shift registers, a linear function and a non-linear function, where one LFSR is used to clock control and the other to generate key stream. The following figure displays it in detail:

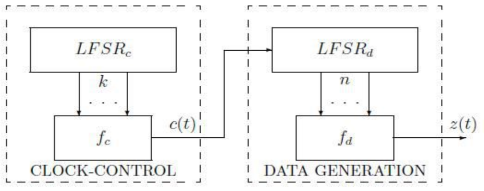


Figure 4 The Structure of Lili-128 Algorithm

$LFSR_c$ contains 39 bits for clock controlling, $k=2$, $c(t)$ ($1 \le c(t) \le 4$) is generated by f_c, where $f_c(x_{12}, x_{20}) = 2x_{12} + x_{20} + 1$. $LFSR_d$ contains 89 bits for key stream generation, $n=10$, and the 10 input positions of f_d, with the algebraic degree of 6, the nonlinearity of 480 and correlation immunity of 3, are (0，1，3，7，12，20，30，44，65，80).

4.1.2 Attack Procedure

Step 1, initialization. Since $LFSR_c$ does not directly control the bit generation, attacking against $LFSR_d$ is a common choice. To make the extended Cube attack possible, an initialization process is introduced to Lili-128 to make sure that the initial vector and key bits are blended thoroughly.

Step 2, choose annihilator. According to the result of algebraic attack in [8], $g(x) = x_{44}x_{80}$ is one of the annihilators of f_d which render $\deg(f_d \cdot g) = 4$ instead of 8.

Step 3, choose initial dimension of cubes and the output bit. In this paper, the 20th output bit is attacked and we search cubes with the dimension of 2 to obtain linear expressions thanks to the annihilator which reduces the algebraic degree of master polynomial to 4.

Step 4, 1-time extension of 2-degree maxterms.

(1) Bound the extension of round as 2;

(2) choose one 2 dimensional cube a time until they are walked over;

(3) modify each variable in the cube according to the algorithm;

(4) choose one number a time from 0 to 87 to form a new 3 dimensional cube ;

(5) sum over the new cube and extract linear expression by linearity test;

(6) extend the algorithm by one round until it meets the bound.

4.1.3 Attack Result

We found 161 cubes with the dimension of 2 within seconds when the initialization round is bounded from 176 to 178. What was more, on the improved pre-processing phase, only 19 2-degree maxterms are needed to extend enough 3 dimensional maxterms, with which 88 linearly dependent expressions can be obtained. The cubes used for extension are as follows:

Round	Cube	Output bit	Round	Cube	Output bit
178	{23，45}	20	177	{22，44}	20
176	{1，50}	20	176	{1，65}	20
176	{2，50}	20	176	{3，50}	20
176	{4，38}	20	176	{5，56}	20
176	{8，56}	20	176	{11，12}	20
176	{12，52}	20	176	{18，57}	20
176	{38，62}	20	176	{31，47}	20
176	{31，58}	20	176	{2，65}	20
176	{3，65}	20	176	{6，65}	20
176	{8，65}	20			

Table 1 2-degree maxterms used for extension

Linear Extension Cube Attack searches 3 dimensional cubes on the basis of 2 dimensional cubes, which dramatically reduces the search scale on 3 dimensional cubes. Only 19×87 times computation are needed to extract enough expressions so that the time complexity of the improved attack on pre-processing phase is $O(2^{13})$ and the total time complexity of our attack is less than $O(2^{14})$. And its data complexity is $O(2^{9})$ since the 88 linear expressions are obtained from 5 cubes with the dimension of 2 and 83 cubes with the dimension of 3.

Compared to the previous attacks on Lili-128, our attack is the best as the following table displays:

	Time and Space Attack[11]	Algebra Attack[8]	Fast Algebra Attack[12]	Cube Attack[13]	Our Attack
Key stream	2^{46}	2^{18}	2^{60}	2^{12}	2^{9}
Pre-processing				2^{16}	2^{13}
On-line	2^{48} DES	2^{57}	2^{34}	2^{12}	2^{10}
Total complexity	2^{48} DES	2^{57}	2^{34}	2^{16}	2^{14}

Table 2 Complexity of various attacks on Lili-128

4.2 Attack 2: Reduced Variants of Trivium

4.2.1 Brief Introduction of Trivium Algorithm

Trivium[1]is one of the candidate algorithms of eSTREAM. The length of IV is 80 bits and the initial key is also 80 bits. During the initialization, the internal state is updated for 1152 rounds. The 288 bits of its internal state store in 3 nonlinear feedback shift register with different length. The following figure displays it in detail:

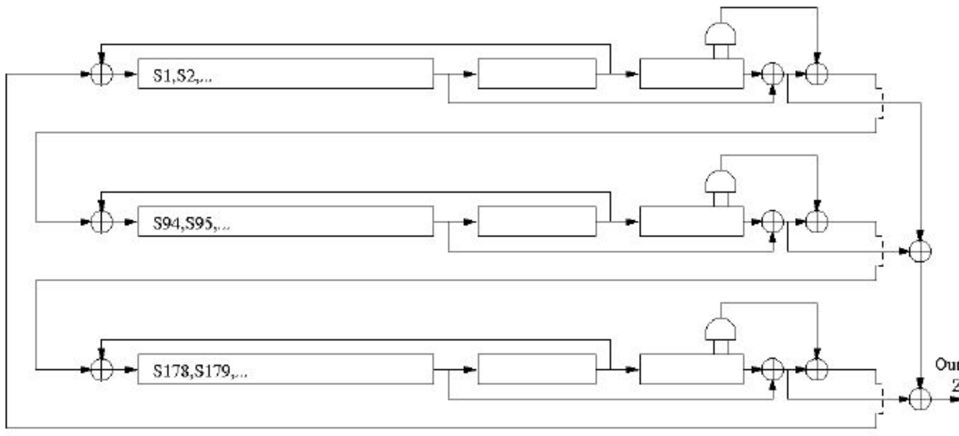


Figure 5 The Structure of Trivium Algorithm

The feedback to each register consists of a nonlinear combination of bits from different registers. The output bit of each round is a linear combination of six state bits, two taken from each register.

4.2.2 Attack Procedure

Step 1, choose initial dimension of cubes. When the initialization round is 525, we randomly search 4 dimensional cubes for maxterms and linear expressions. When the initialization round is 576, according to paper [14], we search cubes with the dimension no larger than 7.

Step 2, choose the output bit.

When the initialization round is 525, we apply the improved attack to the 525th output bit. On the original pre-processing phase, 8 cubes with the dimension of 4 can be obtained ***without*** Gaussian elimination and they are extended on the improved pre-processing phase. The cubes are as follows:

Cube	Expression	Cube	Expression
{0,13,14,70}	x_{57}	{11,21,38,74}	x_{63}
{13,23,72,75}	x_{61}	{34,62,67,75}	x_{54}
{25,26,31,57}	x_{27}	{4,50,72,75}	x_{61}
{22,40,42,68}	x_{27}	{2,6,35,45}	x_{47}

Table 3 Original Results on the 525th Output Bit

When the initialization round is 576, we apply the improved attack to the 576th output bit. On the original pre-processing phase, 20 cubes with the dimension of 6 or 7 can be obtained ***without*** Gaussian elimination and they are extended on the improved pre-processing phase. The cubes are as follows:

Cube	Expression	Cube	Expression
{22,32,35,47,68,77}	x_{55}	{5,7,17,54,58,74}	x_{19}
{9,15,21,40,67,69}	x_{56}	{24,33,42,46,68,76}	x_{20}
{2,6,32,36,38,57}	x_{6}	{11,30,34,45,51,66}	x_{19}
{18,20,53,56,76,78}	$x_{22}+1$	{5,15,26,42,49,62}	x_{4}
{8,10,18,26,30,73}	x_{62}	{22,23,37,56,60,67,77}	x_{19}
{7,9,23,25,50,55,71}	x_{27}	{1,9,11,12,22,41,69}	$x_{61}+x_{67}+1$
{10,11,13,24,29,55,62}	x_{59}	{9,34,44,57,65,68,78}	x_{57}
{1,8,10,15,35,51,70}	x_{57}	{8,18,26,31,37,63,64}	x_{20}
{4,20,23,58,64,68,71}	x_{24}	{20,23,32,34,53,58,74}	x_{19}

{1,13,27,30,35,45,65}	$x_{61}+x_{67}+1$	{7,8,26,29,57,61,77}	x_{64}

Table 4 Original Results on the 576th Output Bit

Step 3, 1-time extension of 12-degree maxterms.

(2) choose one dimensional cube a time from table 2 (table 3) until they are walked over;

(3) modify each variable in the cube according to the algorithm;

(4) choose one number a time from 0 to 79 to form a new 13 dimensional cube ;

(5) sum over the new cube and conduct linearity;

(1) Bound the extension of round as 5 and extend the algorithm by 1 round;

(6) extend the algorithm by one round until it meets the bound.

4.2.3 Attack Result

When the initialization round is 525, 28 5-dimensional cubes can be obtained ***after*** Gaussian elimination by extending the 8 4-dimensional cubes in table 3. And ***31*** key bits can be directly recovered altogether. The extended cubes are as follows:

Cube	Expression	Bit	Cube	Expression	Bit
{2,15,16,72,27}	x_{54}	526	{3,16,17,73,4}	x_{62}	527
{5,18,19,75,7}	x_{7}	529	{5,18,19,75,36}	$x_{38}+x_{62}$	529
{13,23,40,76,39}	x_{65}	527	{14,24,41,77,43}	$x_{47}+1$	528
{14,24,41,77,44}	x_{46}	528	{14,24,41,77,67}	$x_{56}+1$	528
{14,24,41,77,68}	x_{55}	528	{15,25,42,78,56}	x_{22}	529
{14,24,73,76,3}	x_{60}	526	{14,24,73,76,41}	x_{26}	526
{15,25,74,77,1}	x_{64}	527	{15,25,74,77,1}	$x_{64}+x_{63}$	527
{36,64,69,77,10}	x_{58}	527	{36,64,69,77,12}	$x_{66}+1$	527
{38,66,71,79,10}	$x_{53}+x_{68}+1$	529	{38,66,71,79,13}	x_{23}	529
{35,63,68,76,44}	x_{48}	526	{29,30,35,61,17}	x_{37}	527
{30,31,36,62,38}	$x_{55}+x_{66}+1$	528	{30,31,36,62,39}	x_{51}	528
{8,54,76,79,78}	x_{0}	529	{23,41,43,69,14}	x_{10}	526
{24,42,44,70,56}	x_{11}	527	{25,43,45,71,13}	x_{65}	528
{6,10,39,49,51}	x_{53}	529	{7,11,40,50,8}	x_{52}	530

Table 5 The Extended Cubes on the 525th Output Bit

When the initialization round is 576, ***26*** key bits can be directly recovered and a linear equation about another 2 key bits can be obtained by extending those cubes in table 3, while the original attack on this output bit can only recover ***13*** bits and obtain a linear equation about another 2 key bits. The extended cubes are as follows:

Cube	Expression	Bit
{23,33,36,48,69,78,62}	x_{56}	577
{24,34,37,49,70,79,29}	x_{57}	578
{6,8,18,55,59,75,16}	$x_{56}+1$	577
{14,20,26,45,72,74,43}	x_{16}	581
{26,35,44,48,70,78,27}	$x_{39}+x_{57}$	578
{26,35,44,48,70,78,37}	x_{22}	578
{26,35,44,48,70,78,57}	x_{39}	578
{11,13,21,29,33,76,65}	x_{54}	579
{12,14,24,30,34,77,6}	x_{64}	580
{7,17,28,44,51,64,59}	x_{48}	578

{10,20,31,47,54,67,1}	x_2	581
{10,20,31,47,54,67,6}	x_{65}	581
{9,11,25,27,52,57,73,10}	x_{39}	578
{9,11,25,27,52,57,73,28}	$x_{39}+x_{41}+1$	578
{10,12,26,28,53,58,74,0}	x_{55}	579
{11,13,27,29,54,59,75,17}	$x_{15}+1$	580
{24,25,39,58,62,69,79,40}	x_{68}	578
{24,25,39,58,62,69,79,74}	x_{58}	578
{11,12,14,25,30,56,63,29}	x_{60}	577
{5,12,14,19,39,55,74,54}	x_1	580
{8,24,27,62,68,72,75,4}	x_{62}	580
{4,16,30,33,38,48,68,39}	x_{17}	579

Table 6 The Extended Cubes on the 576th Output Bit

Combined with Bedi's result in [14], we can improved the final result by applying the Linear Extension Cube Attack to only one output bit. 48 key bits can be directly recovered by cubes with dimension no larger than 8, while the original Cube attack can only recover 36 key bits on the same condition. With the help of 10-dimensional cubes, the original Cube attack can only recover 45 key bits. The following table displays the results in detail:

	Bedi's	Ours'	Bedi'	Ours'
Dimension	≤ 10	≤ 8	≤ 8	≤ 10
Key Bits	45bits	48bits	36bits	≥ 55 bits

Table 7 Comparison of Results

Note that, this paper conduct the new method of Cube attack on the 576th output bit with boundary of cube dimension, more output bits and larger dimensional cubes can be implemented in future work and the more than 55 key bits should be recovered.

4.3 Analysis

Linear Extension Cube Attack can improve the complexity in two ways, one is that more key bits can be recovered so that the complexity of brute force attack is improved. The other is that the search scale on higher rounds is narrowed. Instead of walking over all of the $d+1$ dimensional cubes, the attacker only need to search on a relatively smaller scale. Assuming there are m public variables altogether, t maxterms of degree d can be obtained by the original attack, then the search scale on $d+1$ dimension is $O(t \times (m-d))$ instead of $O(C_m^{d+1})$.

5. Conclusion

Motivated by the observation in [6], this paper proposes an improved attack on stream ciphers basing on the original Cube attack, i.e. the Linear Extension Cube Attack, which makes improvement on the pre-processing phase of the original attack and the trade-off between time and space enables the attacker to induce maxterms of higher-order from those of lower-order, thus recovering more key bits and improving the search complexity on higher-dimension. This paper provides Linear Extension Cube Attack against Lili-128 algorithm and two reduced variants of Trivium algorithm. For Lili-128 algorithm, only $O(2^9)$ key streams are needed to recover 88 key bits and the attack has time complexity less than $O(2^{14})$. It is the best attack on Lili-128 to the best of our knowledge. For Trivium algorithm, 48 key bits can be recovered by cubes with dimension no larger than 8 when the initialization round is 576, the results are much better than those of the original attacks.

We also find two interesting phenomena during our experiments. First, the improved Cube attack is applied to Trivium algorithm when the initialization round is 672, 9 key bits can be directly recovered by the improved attack on the 672th output bit, which is 2 more than Dinur and Shamir's attack on the same output bit in [1]. However, the performance is not sparkle compared to variants of lower round. Here we propose an open problem about enhancing the performance of the Linear Extension Cube Attack against ciphers with complex initialization.

Second, the algebraic degree of a certain monomial may increase by $a(a \geq 2)$ after 1-round extension. Hence, the extension from d to $d+1$ can also be improved to $d+a$, which extends the attack by adding a new indexes into a cube and searching for linear expressions. Therefore, the improvement of Linear Extension Cube Attack is also considerable in future research.

At last, according to our experiments and results, we should say that our new method of Cube attack is efficient and of certain importance, especially with the application of lightweight cryptography.

References

[1]. Dinur I, Shamir A. Cube Attack on Tweakable Black box polynomials[A]. Advances in Cryptology-EUROCRYPT 2009[C]. Springer Berlin Heidelberg, 2009: 278-299.

[2]. Pierre Alain Fouque, Thomas Vannet. Improving Key Recovery to 784 and 799 rounds of Trivium using Optimized Cube Attacks[EB/OL]. In Fast Software Encryption 2013. http://fse2013.spms.ntu.edu.sg:80.

[3]. Xinjie Zhao, Tao Wang, Shize Guo.Improved Side Channel Cube Attacks on PRESENT [EB/OL]. Cryptology ePrint Archive, 2011. http://eprint.iacr.org/2011/165.

[4]. Aumasson J P, Meier W, Dinur I. Cube Testers and Key Recovery Attacks on Reduced Round MD6 and Trivium[A]. Fast Software Encryption[C]. Springer Berlin Heidelberg, 2009: 1-22.

[5]. Dinur I, Shamir A. Breaking Grain-128 with Dynamic Cube Attacks[A]. Fast Software Encryption[C]. Springer Berlin Heidelberg, 2011: 167-187.

[6]. Mroczkowski P, Szmidt J. The Cube Attack on Stream Cipher Trivium and Quadraticity Tests[J]. Fundamenta Informaticae, 2012, 114(3): 309-318.

[7]. Yu Sun, Yongjuan Wang. Cube Attack and Its Improvement[J]. Computer Science, 2012, 39(100): 77-80.

[8]. Nicolas T. Courtois, Willi Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback[A]. Advances in Cryptology-EUROCRYPT 2003[C]. Springer Berlin Heidelberg, 2003: 345-359.

[9]. Guo Li, Wei Wang, Yongjuan Wang. Cube Attack on Lili-128 Algorithm[J]. Advanced Cipher Study, 2012, 2.

[10]. E. Dawson, A. Clark, J. Golic et al. The Lili-128 Keystream Generator[C]. Proceedings of First NESSIE Workshop, 2000.

[11]. Markku-Juhani Olavi Saarinen: A Time-Memory Trade-off Attack Against LILI-128[A]. FSE 2002[C], LNCS 2365, Springer, 2002: 231-236.

[12]. Nicolas T. Courtois. Fast Algebraic Attacks on Stream Ciphers with Linear Feedback[A]. Advances in Cryptology-CRYPTO 2003[C]. Springer Berlin Heidelberg, 2003: 176-194.

[13]. Guo Li, Wei Wang, Yongjuan Wang. Cube Attack on Lili-128 Algorithm[J]. Research of Enhanced Ciphers, 2010(2).

[14]. Bedi, S.S., Pillai, N.R. Cube Attacks on Trivium[EB/OL]. Cryptology ePrint Archive, 2009. http://eprint.iacr.org/2009/015.