# Cube攻击原理与改进

孙 宇 王永娟

(洛阳外国语学院 洛阳 471003)

**摘 要** 介绍Cube攻击的原理、详细步骤及算法;通过比较Cube攻击和AIDA攻击,详细分析攻击原理,证明了Cube攻击在原理上等价于高阶差分;讨论线性化测试,得出Cube攻击以高概率成功的结论;给出关键点的详细算法,对简化Trivium进行攻击;对Cube攻击方法进行改进。

**关键词** Cube攻击,AIDA攻击,代数IV,高阶差分,Trivium

## Principles and Improvement of Cube Attack

SUN Yu WANG Yong-juan

(Luoyang University of Foreign Languages, Luoyang 471003, China)

**Abstract** This paper introduced the principles, detailed steps and algorithms of Cube attack. By comparing Cube attack and AIDA attack, we analyzed its principles in detail, and proved that Cube attack is equivalent to higher order difference in aspect of principle. And then, with discussing linear test, we concluded that Cube attack can be successful with high probability. In particular, this paper gave algorithms of the key steps, and employed them to attack the reduced Trivium. Finally, we improved the Cube attack.

**Keywords** Cube attack, AIDA attack, Algebraic IV, Higher order difference, Trivium

## 1 引言

2007年,针对序列密码Trivium算法,Michael Vielhaber提出了一种已知明文攻击——AIDA攻击[1]。给定一个初始变量子空间,攻击者用空间中所有的初始变量进行加密,求AIDA和,进行线性化测试,获得关于密钥的单变量方程,然后选择明密文对逐比特恢复密钥,完成攻击。针对One. Fivium,采用AIDA攻击,对于输出时刻不超过625的密文,通过选择47*64个初始密钥,可恢复47比特密钥。2009年,M. Vielhaber采用快速Reed-Muller变换提高搜索线性方程的效率,针对Bivium进行攻击,取得不错的效果[2]。

在AIDA攻击基础上,2008年Adi Shamir和Itai Dinur提出了Cube攻击[3]。对于拥有高强度抗攻击特征的密码体制,采用Cube攻击,选择合适的初始变量进行线性测试,建立关于密钥的线性方程,从而实施攻击。从原理上看,两者与高阶差分[4]有着密不可分的联系,采用差分得到密钥的线性方程;从攻击条件和对象等看,Cube攻击比AIDA攻击更普遍。由于Cube攻击适用较广,是近期密码学者关注的一个热点问题,出现了大量的文献和理论成果[5-9]。例如,Jean-Philippe Aumasson等使用Cube攻击对14轮的Hash函数MD6版本进行攻击,复杂度为$O(2^{22})$[4];Piotr Mroczkowski、Janusz Szmidt将Cube攻击应用于4轮的CTC分组密码,并能完全恢复出120bit密钥[5]。本文从攻击条件、原理和攻击对象等方面,对AIDA攻击和Cube攻击进行比较和分析,发现Cube攻击更为有效。由于次数$d$和初始变量选取是Cube攻击的关键问题,本文以简化的Trivium算法为例,通过半分查找和随机查找方法的对比,结合复杂度分析,给出快速确定次数$d$和搜索初始变量的方法。利用已恢复的密钥和满足Cube条件的集合,对Cube攻击进行改进,提高了Cube攻击的效率。

## 2 攻击的原理及步骤

### 2.1 Cube攻击原理及其步骤

Cube攻击是确定性选择明文攻击,对序列密码、分组密码和hash函数等都有不错的攻击效果。记初始向量$v=v_1\cdots v_m$,密钥$k=k_1\cdots k_n$,时刻$t$输出的密文为$z_t$,则加密过程可以表示为$z_t=p_t(v,k)$。它的主要思想就是选择不同的$v$产生多个方程$z_t=p_t(v,k)$,结合这些方程生成关于密钥的线性方程$f(k)=0$,如果得到足够多的线性方程,通过求解恢复密钥。

**定义1** $p(v,k)$在集合$I$上的Cube和记为$p_I=\sum_{v\in C_I}p(v,k)$,其中$I\subset\{1,2,3,\cdots\}$,$v=(v_1,\cdots,v_m)$,$k=(k_1,k_2,\cdots,k_n)$,$C_I=\{v|0\leqslant v_i\leqslant 1,i\in I\}$。

**定义2** Cube分解是指多项式$p(v,k)$按$v^I$分解成$v^I p_{S(I)}+q(x)$,其中$v^I=\prod v_i,i\in I$,$q(x)$不含$v^I$作为因子的项。如果$p_{S(I)}$次数等于1,那么这样的$I$和$v^I$称为满足Cube攻击条件的,简称为满足条件的。

**定理1**[3] 在$GF(2)$上,对任意多项式$p(v,k)$和集合$I$

孙 宇(1988—),男,硕士,主要研究方向为密码学,E-mail:systudent@yahoo.com.cn;王永娟(1982—),女,博士,副教授,主要研究方向为密码学、信息安全,E-mail:pinkywyj@163.com。

做 Cube 分解，满足 $p_{S(I)}=p_I$。

Cube 攻击基于以上定理，它指出给定一个集合 $I$ 和密钥 $k$，取 $C_I$ 中所有初始向量 $v$ 进行加密，得到形如 $z_t=p_t(v,k)$ 的、方程个数为 $2^{|I|}$ 的方程组。将这些等式相加，得到 $v^J$ 的商式 $p_{S(I)}$，此时 $p(v,k)$ 的次数随即降低至少 $|I|$ 次。如果 $p_{S(I)}$ 的次数恰好为 1，那么 $I$ 上 Cube 和就产生一个关于密钥的线性方程。如果存在足够多的满足条件的 $I$，攻击者将得到关于密钥的线性方程组，通过求解线性方程组就可以恢复出一些密钥。

从以上讨论可知，Cube 攻击的步骤分为两步：

1）预处理阶段：搜索满足条件的集合 $I$，建立关于密钥的线性方程组；

2）计算阶段：通过求解线性方程组恢复出一些密钥。

### 2.2 原理分析

#### 2.2.1 与 AIDA 攻击的比较

**定义 3** $p(v,k)$ 在集合 $I$ 上的 AIDA 和记为 $t_I=\sum_{v\in A_I} p(v,k)$，其中

$$A_I=\{v\mid 0\leqslant v_i\leqslant 1, v_j=0, j\notin I, i\in I\}$$

**定理 2**[1] 在 $GF(2)$ 中，如果 $p_t(v,k)=v^J\cdot k_i+Z$，其中 $Z$ 中不含形如 $v^I\cdot(k_1\cdot k_2\cdots k_j)$ 的项，那么 $k_i=\sum_{v\in A_I} p_t(v,k)$。

定理 2 说明，如果找到某个 $v^J$，使得某时刻的输出关于初始向量和某比特密钥的方程满足定理 2 的条件，那么通过 $2^{|I|}$ 个选择明密文对，可以恢复某比特密钥。

由此可见，两种攻击都是通过寻找合适的集合 $I$，在 $I$ 求和，使得结果次数为 1，然后恢复出密钥。因此，两种攻击的关键都是搜索满足条件的 $I$，Shamir 采用随机搜索的改进方法寻找 $I$[3,10]。

由于 Cube 攻击在除 $v^I$ 外的公共变量的值没有限制，通过给其它公共变量赋值 0 或 1，可以增加 $p_{S(I)}$ 的项，因此更容易找到满足条件的 $I$。在实际攻击中这步是非常有效的，比如当找到正确的 $I$，使得

$$p_{S(I)}=\sum_{i=1}^{n}(a_i+\sum_{I^i\subset\{1,2,3,\cdots,m\}} b_{I^i}\cdot v^{I^i})\cdot k_i+a$$

这样通过给 $v$ 赋不同的值，能够获得更多的方程。虽然，那些“更多”的方程也能够通过某个 $I$ 得到，但是考虑到时间复杂度，通过 $I$ 得到“更多”的方程将会增加时间复杂度，甚至是在时间上不可能的。

#### 2.2.2 与高阶差分的关系

**定理 3**[4] 记 $L[\alpha_1,\alpha_2,\cdots,\alpha_i]$ 是 $\alpha_1,\alpha_2,\cdots,\alpha_i$ 生成的线性空间，那么

$$\Delta^{i}_{\alpha_1,\alpha_2,\cdots,\alpha_i} f(x)=\sum_{\beta\in L[\alpha_1,\alpha_2,\cdots,\alpha_i]} f(x\oplus\beta)$$

**定理 4** Cube 攻击在原理上等价于高阶差分。

证明：给定大小为 $s$ 的集合 $I$，记 $V_1=L_1[\alpha_1,\alpha_2,\cdots,\alpha_s]$，其中 $\alpha_i=(a_1,\cdots,a_m)$，$a_j=1, j\in I$。

记 $V_2=L_2[\alpha_1,\alpha_2,\cdots,\alpha_s]$，其中

$\alpha_i=(a_1,\cdots,a_m)$，$|\alpha_i|=1, a_j=1, j\in I$

当 $\beta$ 分别跑遍 $V_1$ 和 $V_2$ 时，$x\oplus\beta$ 分别跑遍 $C_I$ 和 $A_I$，因此 $I$ 上的 Cube 和可以用高阶差分表示：

$$\begin{aligned}p_{S(I)}&=\sum_{v\in C_I} p\mid_v=\sum_{\beta\in L_1[\alpha_1,\alpha_2,\cdots,\alpha_s]} p(v\oplus\beta)\\&=\Delta^{(s)}_{\alpha_1,\alpha_2,\cdots,\alpha_s} p(v,k)\end{aligned}$$

证明完毕。

同理，$I$ 上的 AIDA 和可以用高阶差分表示：

$$\begin{aligned}k_t&=\sum_{v\in A_I} p_t(v,k)=\sum_{\beta\in L_2[\alpha_1,\alpha_2,\cdots,\alpha_s]} p(v\oplus\beta)\\&=\Delta^{(s)}_{\alpha_1,\alpha_2,\cdots,\alpha_s} p(v,k)\end{aligned}$$

由此可见，在 $m$ 个点上做高阶差分等价于对多项式在某个集合 $I$ 上做 Cube 和，而这 $m$ 个点和 $I$ 之间存在双射关系。因此，Cube 攻击在原理上可用高阶差分表示，而 Cube 攻击侧重于实验搜索，而高阶差分是从理论上进行分析。做高阶差分时，AIDA 攻击要求除 $v^J$ 外的公共变量全为 0，而 Cube 攻击不加限制，所以 Cube 攻击可做差分点比 AIDA 攻击多 $2^{m-s}-1$ 个。两种攻击的原理对比如表 1 所列。

表 1

| 攻击类型 | 原理 | 差分点 | 恢复密钥 |
|---|---|---|---|
| Cube | 高阶差分 | 较多 | 线性方程 |
| AIDA | | 较少 | 逐比特 |

### 2.3 Cube 攻击的详细步骤及算法

分析定理 1，Cube 攻击的关键问题是寻找满足条件的 $I$。对于 $d$ 次随机多项式，任意一个 $d$ 次项都以很大概率满足条件；对于非随机多项式，攻击者在 1 和 $m$ 之间选取 $k$ 和大小为 $k$ 的集合 $I$，给不在 $I$ 中的公共变量赋予 0，然后在 $I$ 求 Cube 和。如果选择的 $k$ 太大，比如大于次数 $d$，那么得到的和为常量，这时需要降低 $k$ 重新进行搜索；如果选择的 $k$ 太小，那么得到的和将很可能不是线性多项式，这时需要增加 $k$ 重新进行搜索；满足条件的集合 $I$ 介于这两种情况之间；如果没有得到满足条件的集合 $I$，那么选择 $k$ 后就要选择不同的集合 $I$ 重新搜索。详细步骤如下：

1）在可选择范围内随机选择 $k$，试验中可选范围为 $[2,16]$；

2）随机选取大小为 $k$ 的 $I$，在 $I$ 上求 Cube 和。如果无论选什么密钥，Cube 和均为常数，则返回第一步，并且减少 $k$；如果 Cube 和是非线性的，返回第一步，并且增加 $k$；如果 Cube 和是线性的，但不是常数，进入第三步；如果得不到合适的 $I$，则返回第一步，重新搜索 $k$，并且选择不同的 $I$；

3）确定线性表达式。除了 $t_I$ 中的元素，所有初始向量和密钥比特设置为 0，在 $I$ 上求 Cube 和得到线性项中常数项；如下循环 $n$ 次，除了 $t_I$ 中的变量和某个密钥 $k_i$（赋值为 1），所有公共向量和密钥比特赋值 0，在 $I$ 上求 Cube 和再加常数项得到 $k_i$ 的系数；

4）按上述方法得到足够多的线性表达式，生成一个关于密钥的线性方程组，然后通过求解方程组恢复密钥。

关于如何判断 Cube 和是否为线性多项式，文献[10,11]中给出很多有效的测试方法，本文随机选取 20 对密钥 $(x,y)$ 采用 BLR 方法[10]进行测试，即判断

$$p_{S(I)}(x)\oplus p_{S(I)}(y)\oplus p_{S(I)}(x+y)\oplus a=0$$

是否成立。若对于 20 对都成立，并且存在某个 $x$ 和 $y$，使得 $p_{S(I)}(x)+p_{S(I)}(y)=1$，那么 $I$ 就是满足 Cube 条件的。

如何快速确定 $k$ 是影响攻击的效率。一方面，出于安全

性考虑，密码设计者设计的 $p(x)$ 的次数比较高，对于序列密码，当密文输出时增加尤为如此；另一方面，设计者总会把明文和密钥进行充分混淆和扩散。因此，当出现一个满足条件的 $I$ 时，无论此时 $k$ 是否为多项式次数 $d$，有理由相信还有更多大小为 $k$ 的 $I$。本文按顺序在可计算范围(min，max)内半分搜索 $k$，直到搜索出第一个满足条件的 $I$，然后穷举所有相同大小的 $I$。$k$ 初始为 4，步骤如下：

1)选取 1000 个大小为 $k$ 的 $I$，分别用 20 对密钥进行线性化测试，如果 $k>$max，则攻击失败，退出；

2)如果得到非线性项，立刻退出 1)中测试，且 min←$k$，$k$←min(3$k$/2，(max+min)/2)；如果选取的 1000 个 $I$ 用 20 密钥对测试均为 0，则 max=$k$，$k$←max(3$k$/4，max)；如果存在某个 $I$，使得选取 20 对密钥测试均为 1，则 $k$←$k$+1；如果通过线性化测试或者 max=min，则退出；

3)重复 1)、2)，直到找到正确的 $I$。

### 2.4 攻击的复杂度和成功概率

在计算阶段，Cube 攻击和求解线性方程组的时间复杂度相同。因此，Cube 攻击的复杂度由预处理阶段决定，即确定次数 $k$ 和搜索集合 $I$ 两部分。

设加密运算复杂度为 O($N$)，若确定次数 $k$ 的搜索结果为 $d$，则这部分复杂度约为 O($2^d N$)。而搜索集合 $I$ 的复杂度为($O\binom{m}{k}2^d N$)，所以整个复杂度约为 O($\binom{m}{k}2^d N$)。当 $k$ 和 $m$ 比较大时，求 Cube 和所需的时间复杂度极速增加。在 CPU 频率为 1.60GHz 的 PC 上，对 Trivium 算法进行攻击的次数 $k$ 不能超过 12。

通过线性化测试得到攻击所需的线性方程组，因此，线性化测试的有效性是攻击成功的重要因素。显然，线性化测试是一个概率事件，若

$$p_{S(D)}=k_1+k_2k_3k_4k_5$$

则 $p_{S(D)}$ 将会以约 $1-2^{-4}$ 的高概率通过线性化测试。

**定理 5** 单项式 $x_1x_2\cdots x_m$ 通过线性化测试的概率为

$$\Big(\sum_{\substack{s>0,t>0\\ s+t>m}}\binom{m}{s}\binom{m}{t}+\sum_{\substack{k=1\\ s+t\leqslant m}}^{\min(s,t)}\binom{m}{s}\binom{s}{k}\binom{m-s}{t-k}+\sum_{s=1}^{m-2}\sum_{t=1}^{m-s-1}\binom{m}{s}\binom{m-s}{t}\Big)/2^{2m}$$

当 $m$ 增加时，这种单项式以接近 1 的概率通过线性化测试，表 2 给出了 $m$ 与概率的关系。

表 2

| m | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 概率 | 0.625 | 0.719 | 0.836 | 0.912 | 0.955 | 0.977 |
| m | 8 | 9 | 10 | 11 | 12 | 13 |
| 概率 | 0.988 | 0.994 | 0.997 | 0.999 | 0.999 | 0.999 |

密码算法的扩散和混淆性质，使得 $m$ 次单项式不会单独出现，而会以一定概率出现 $n$ 元 $m$ 次齐次多项式。特别地，当 $m$=2 时，有以下定理。

**定理 6** $n$ 元 2 次齐次多项式以($1/2-1/2^n+1/2^{2n+1}$)通过线性化测试的概率出现。

当增加时，这种齐次多项式以接近 1/2 的概率通过线性化测试，表 3 给出了 $n$ 与概率的关系。

表 3

| n | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 概率 | 0.375 | 0.406 | 0.445 | 0.471 | 0.485 | 0.492 |

当选择 20 对随机密钥进行测试时，$n$ 元 2 次通过线性化测试的概率小于 $1/2^{20}$，所以 Cube 攻击成功的概率是非常大的。

## 3 Trivium 算法的 Cube 攻击

### 3.1 Cube 攻击的改进

通过实验数据分析，攻击中总能从部分线性无关的方程组中恢复部分密钥。如果攻击中找到正确的 $I$ 或恢复了某个密钥 $k_t$，攻击者可以做改进：

做线性化测试时，设置 20 对随机密钥中的 $k_t$ 恒为 0。如果 $p_{S(D)}$ 含有形如 $k_tk_{j_1}\cdots k_{j_t}$，$t\geqslant 1$ 的项，那么 $p_{S(D)}$ 也可以通过线性化测试，找出更多线性方程。也可以设置 20 对随机密钥中的 $k_t$ 恒为 1。如果 $p_{S(D)}$ 含有形如 $k_tk_j$ 项，那么 $p_{S(D)}$ 也可以通过线性化测试，找出更多线性方程。

找到正确的 $I$ 后，给其它公共变量赋不同的值，共 $2^{m-|I|}$ 种，从而能够获得更多的以下类型的方程：

$$p_{S(D)}=\sum_{i=1}^{n}(a_i+\sum_{I^i\subset\{1,2,3,\cdots,m\}}b_{I^i}\cdot v^{I^i})\cdot k_i+a$$

### 3.2 Trivium 算法的 Cube 攻击

在确定次数 $k$ 和搜索集合 $I$ 时，如何选取 $I$ 是攻击的关键，本文分别采取随机和半分方法选取 $I$。把 $I$ 可能选取的 $\binom{m}{k}$ 种情况依次排序，建立这 $\binom{m}{k}$ 种情况与$\{1,2,\cdots,\binom{m}{k}\}$的单射关系。随机方法就是从$\{1,2,\cdots,\binom{m}{k}\}$中随机选取一个数，然后分别进行线性化测试。半分选取方法就是从$\{1,2,\cdots,\binom{m}{k}\}$中对半选取进行线性化测试，得到线性方程。

简化 Trivium 为初始循环 100 次，密文输出时刻为 400，进行编程，得到 $d$ 为 5，其中密钥见附录 1。下面在确定次数 $d$ 中分别用随机和半分算法选取 $I$ 来进行比较，其中半分是指随机确定范围后进行半分，结果如表 4 所列。

表 4

| 指标 | 随机算法 | 半分算法 |
|---|---|---|
| 时间 | 慢 | 快 |
| 方式 | 通过测试 | Min=Max |
| I | {11,28,37,38,66} | 无 |

由于带有随机性，随机算法并不能取遍所有的满足条件的 $I$，若考虑不重复随机，那么复杂度将会增加，因此执行效率比半分算法低。但是，在确定 $d$ 的过程中，随机算法在实验中能够找到满足条件的 $I$，而且时间复杂度是平凡的。因此，在确定 $d$ 中采用随机选取 $I$ 的方法，而在搜索满足条件的 $I$ 时，采用半分方法穷举搜索。

采用半分穷举大小为 5 的集合 $I$，在 CPU 频率为 1.60GHz的 PC 上，用不超过 10 分钟时间得到 9 个线性无关的线性方程，如表 5 所列。

表 5

| 集合 I | 线性方程 | 集合 I | 线性方程 |
|---|---|---|---|
| 2,38,55,67,71 | 54,1 | 0,28,42,45,68 | 55,1 |
| 1,28,53,65,76 | 65,0 | 11,41,42,68,79 | 13,1 |
| 55,60,63,73,78 | 63,1 | 10,26,43,56,72 | 2,65,0 |
| 41,47,55,71,72 | 1,64,0 | 29,30,47,54,55 | 1,13,15,64,0 |
| 11,28,42,45,68 | 66,1 | | |

穷举所有可能的集合，用不超过 12 小时的时间得到 34 个线性无关的方程，恢复出至少 29bit 密钥，详见附录 2。

另外，采用半分穷举大小为 4 的集合 $I$，在 CPU 频率为 1.60GHz 的 PC 上，3 小时内穷举得到含 44 组线性无关的线性方程，详见附录 3。

**结束语** Cube 攻击是一种对序列密码、分组密码和 hash 函数都有效的攻击方法，能够以极高的概率成功攻破复杂的密码算法。针对 Cube 攻击的关键——如何高效准确地确定次数 $d$ 和搜索满足条件的 $I$，本文对简化的 Trivium 进行测试。通过对复杂度和试验结果的结合分析，得到先用随机方法确定次数 $d$、然后用半分穷举方法搜索 $I$ 的方法。利用这个方法，对初始循环 100 次、密文输出时刻为 400 的 Trivium 算法，能够很快地得到 $d=5$，并且能够恢复出全部密钥比特。但是，若初始循环次数增大，次数 $d$ 相应增加，Cube 和的时间复杂度就会快速增加。因此，如何有效地选择 $I$ 依然是以后研究的重点。由于能够应用到不同的密码体制，Cube 攻击存在巨大研究价值，如何提高线性化测试方法，如何利用低次的 $p_{S(I)}$，如何结合其他攻击方法等，这些都是 Cube 攻击亟需解决的问题。

## 参 考 文 献


[1] Vielhaber M. Breaking One. Fivium by AIDA an Algebraic IV Differential Attack[Z]. IACR Cryptology ePrint Archive, 2007

[2] Vielhaber M. AIDA Breaks BIVIUM(A and B) in 1 Minute Dual Core CPU Time[C]//IACR Cryptology ePrint Archive, 2009

[3] Dinur I, Shamir A. Cube attack on Tweakable black box polynomials[C]// AJoux, ed. EUROCRYPT 2009. LNCS, vol 5479, Springer, 2009: 278-299

[4] Lai X J. Higher Order Derivative and Differential Cryptanalysis [J]. private communication, September 1993, 30

[5] Dinur I, Shamir A. Side Channel Cube Attacks on Block Ciphers [J]. IACR Cryptology ePrint Archive, 2009, 127

[6] Bedi S S, Pillai R. Cube attacks on Trivium[J]. IACR Cryptology ePrint Archive, 2009, 15

[7] Aumasson J P, Meier W, Dinur I, et al. Cube Testers and Key Recovery Attacks on Reduced Round MD6 and Trivium[J]. Fast Software Encryption, Springer-Verlag, 2009

[8] Mroczkowski P, Szmidt J. The Cube Attack on Courtois Toy Cipher[C]//IACR Cryptology ePrint Archive Proceedings of WEWoRC 2009. LNCS. Springer, 2009

[9] Mroczkowski P, Szmidt J. The Cube Attack on Stream Cipher Trivium and Quadraticity Tests[C]//Rump Session. CRYPTO. 2010

[10] Fischer S, Khazaei S, Meier W. Chosen IV statistical analysis for key recovery attacks on stream ciphers[C]// AFRICACRYPT. 2008: 236-245

[11] Mroczkowski PSzmidt J. The Cube Attack on Stream Cipher Trivium and Quadraticity Tests [C]// Rump Session. CRYPTO. 2010


**附录 1**

密钥：

1110 1111 0010 0101 0001 1101 1010 0101 0000
1111 1011 1111 1100 0111 1111 0001 1010 1101
0010 0100

## 附录 2

| 集合 I | 线性方程 | 集合 I | 线性方程 |
|---|---|---|---|
| 2,38,55,67,71 | 54,1 | 27,65,71,74,75 | 60,0 |
| 1,28,53,65,76 | 65,0 | 20,25,63,64,67 | 52,0 |
| 55,60,63,74,76 | 63,1 | 23,66,67,68,72 | 25,0 |
| 41,47,55,71,71 | 1,64,0 | 9,10,28,67,69 | 56,1 |
| 11,28,42,69,79 | 66,1 | 24,64,67,68,78 | 22,0 |
| 0,28,42,45,68 | 55,1 | 9,57,63,66,67 | 0,12,26,27,54,63,0 |
| 18,24,25,59,76 | 15,1 | 16,19,32,53,54 | 14,0 |
| 11,41,42,68,79 | 13,1 | 50,57,70,71,74 | 61,0 |
| 4,31,61,69,75 | 64,1 | 2,23,24,52,64 | 53,1 |
| 10,26,43,56,72 | 2,65,1 | 9,35,43,68,70 | 37,1 |
| 21,57,67,73,75 | 62,0 | 8,36,55,67,73 | 40,1 |
| 11,25,69,70,79 | 16,0 | 11,42,54,66,68 | 68,1 |
| 5,41,48,72,75 | 59,1 | 9,39,68,69,79 | 58,1 |
| 17,18,34,41,70 | 38,1 | 16,17,45,46,78 | 67,0 |
| 13,19,26,27,72 | 0,63,0 | 24,32,33,54,68 | 23,1 |
| 19,50,54,66,70 | 39,1 | 19,65,68,71,79 | 57,1 |
| 10,24,25,72,73 | 15,16,17,1 | 9,13,22,68,69 | 24,1 |

**附录 3**

| 集合 I | 线性方程 | 集合 I | 线性方程 |
|---|---|---|---|
| 0,1,2,24 | 65,0 | 0,11,19,62 | 21,1 |
| 0,1,5,26 | 15,1 | 0,11,33,62 | 22,0 |
| 0,1,1,12,25 | 14,0 | 0,13,30,44 | 2,15,17,0 |
| 0,1,14,38 | 54,1 | 0,14,15,37 | 41,0 |
| 0,1,18,43 | 58,1 | 0,14,15,38 | 40,1 |
| 0,1,19,26 | 13,1 | 0,19,42,77 | 57,59,0 |
| 0,1,19,42 | 1,58,64,1 | 0,22,63,77 | 26,1 |
| 0,1,25,66 | 55,1 | 0,23,63,77 | 25,0 |
| 0,1,26,51 | 53,1 | 0,38,39,50 | 52,0 |
| 0,1,42,65 | 56,1 | 1,4,9,67 | 0,1 |
| 0,3,5,76 | 63,1 | 1,6,29,75 | 62,0 |
| 0,3,6,75 | 64,1 | 1,10,35,74 | 37,1 |
| 0,4,31,64 | 51,54,1 | 2,3,8,72 | 61,0 |
| 0,4,46,79 | 54,66,0 | 2,3,14,73 | 54,60,1 |
| 0,4,61,78 | 67,0 | 4,9,22,70 | 24,1 |
| 0,5,6,16 | 59,1 | 6,7,44,68 | 3,0 |
| 0,5,27,77 | 16,0 | 6,55,58,64 | 0,68,69,1 |
| 0,5,64,66 | 68,1 | 8,31,44,46 | 33,60,0 |
| 0,9,19,62 | 23,1 | 10,12,26,46 | 16,50,0 |
| 0,10,25,37 | 39,1 | 10,12,26,47 | 49,1 |
| 0,10,26,36 | 38,1 | 11,12,14,61 | 48,1 |
| 0,11,12,43 | 2,65,1 | 13,32,65,79 | 34,0 |