对 Bivium 流密码的变元猜测代数攻击

李 昕[1,2],林东岱[1]
(1.中国科学院软件研究所信息安全国家重点实验室,北京 100190;2.中国科学院研究生院,北京 100190)

摘 要: 非线性方程组的求解是代数攻击的关键一环.对于一个具体的密码系统,在转化为方程组后,由于其计算上的复杂性,一般采用先猜测部分变元,再进行求解分析的方法.本文首先给出了对于猜测部分变元后子系统平均求解时间的估计模型,提出了基于动态权值以及静态权值的猜测变元选则方法和面向寄存器的猜测方法.在计算 Gröbner 基的过程中,对变元序的定义采用了 AB,S,S-rev,SM,DM 等十种新的序.同时,提出了矛盾等式的概念,这对正确分析求解结果以及缩小猜测空间有重要作用.最后,我们对 Bivium 流密码算法的攻击时间进行了估计.结果表明,在最坏情况下,使用 DM-rev 序及 Evy3 的猜测位置,猜测 60 个变元有最优的攻击结果,约 2 exp(39.16)秒.

关键词: 方程组求解; Gröbner 基; Bivium 流密码算法; 猜测决策算法; 矛盾等式
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2011) 08-1727-06

Guessing Specific Variables in Algebraic Attacks on Bivium

LI Xin[1,2], LIN Dong-dai[1]
(1. *State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China*;
2. *Graduate School of the Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract: Solving an equation system is a very important step in algebraic attack. For a cryptosystem, after being transformed to equations, we often need to employ guess-and-determine algorithm to estimate computational complexity of this attack. In this paper, we introduce a model to estimate average time in solving subsystems more accurately, and propose some criteria on selecting specific guessed variables to speed up the solving efficiency, which based on static weight and dynamic weight etc. For comupting Gröbner bases, we use serveral varible order which are AB, S, S-rev etc. Meanwhile, we introduce the concept of conflicting equations, and show the importance for correct analysis and narrow guessing space. In the end, we estimate the time of attacking Bivium. Experiments showed that, in the worst cases, guessing 60 varibles in the Evy3 position and with DM-rev varible order will have the optimal result, that is about 2 exp(39.16) seconds.

Key words: equations solving; Gröbner bases; Bivium; guess-and-determine algorithm; conflicting equations

1 引言

近年来,代数攻击逐渐成为密码分析中的热点,出现了大量的对于分组密码,流密码,公钥密码的研究成果[1~4].代数攻击的发展也促进了解方程技术的进一步发展.如线性化算法,XL 算法[5],吴方法[6],Gröbner Basis 算法[7,8]等.其中,Gröbner Basis 算法最令人瞩目,其实现版本包括 Buchberger 算法, slimgb 算法, F_4, F_5 算法等.2003 年的美密会上,Faugère 公布了他对 HFE 公钥密码系统的分析结果[9],即使用 F_5 算法可以在 2 天内破解 HFE Challenge 1.然而,众所周知,求解多变元二次方程系统,也即 MQ 问题是一个 NP 难题.对于更大规模的方程系统,上述算法均很难直接求解.

先猜测部分变元,把原始的大的方程系统化为若干小的方程系统,逐个进行求解分析是一种变通的做法.如可以指定猜测 n 个变元,把整个方程系统化为 2^n 个子系统,一旦求出某个子系统中其余变元的值,便可从中恢复密钥.否则,若方程无解,说明猜测错误,继续进行下一组猜测.在最坏情况下的猜测数为 2^n 个子系统.本文针对 Bivium 流密码所产生的方程系统,采用 Gröbner 基算法进行了相关求解分析,如何选择最优的猜测变元,最优的 Gröbner 基算法的变元序以及最佳的猜测变元个数是本文主要要解决的问题.其相关实验均是在最坏情况下做的.

收稿日期:2010-08-12;修回日期:2011-01-19
基金项目:国家 863 高技术研究发展计划(No.2011CB302400);国家自然科学基金(No.60970152)

2 Bivium 流密码算法描述

Trivium 是由 C. De Canniere 和 B. Preneel 在 2005 年为欧洲流密码计划而设计的流密码算法. Bivium 是 Trivium 的一个简化版本.

Bivium 的主要参数包括:80 比特的密钥,80 比特的初始值,177 比特的内部状态. Bivium 工作分两步,第一步由 80 比特的密钥和 80 比特的初始值经过初始化,生成 177 比特的初始内部状态. 第二步对内部状态进行非线性反馈移位,并由关于内部状态的线性函数输出密钥流比特. 密钥流由以下步骤生成:

算法 1 Bivium(m)

1: for $i = 1$ to m do
2: $t_1 \rightarrow s_{66} + s_{93}$
3: $t_2 \rightarrow s_{162} + s_{177}$
4: $z_i \rightarrow t_1 + t_2$
5: $t_1 \rightarrow t_1 + s_{91}s_{92} + s_{171}$
6: $t_2 \rightarrow t_2 + s_{175}s_{176} + s_{69}$
7: $(s_1, s_2, \cdots, s_{93}) \rightarrow (t_2, s_1, \cdots, s_{93})$
8: $(s_{94}, s_{95}, \cdots, s_{177}) \rightarrow (t_1, s_{94}, \cdots, s_{176})$
9: end for

这里 m 表示需要生成的密钥流比特数,$m \leqslant 2^{64}$. 初始状态的生成,是把 80 比特的密钥,80 比特的初始值和一些 0/1 比特装载入内部状态,然后经过 177×4 轮状态更新得到. 并且这个内部状态更新与第二步中的内部状态更新完全相同,只是不输出密钥流比特. 用变量 $s_1, s_2, \cdots s_{177}$ 表示 177 比特内部状态,用 z_i 表示每个时钟所产生的密钥流.

由算法 1,可以写出变元 s_i 与 z_i 之间的等式. 对于 Bivium 流密码系统,可以有两种方法写出其方程系统[10,11],一种是不添加任何新的变元,即保持 $s_1, s_2, \cdots s_{177}$ 作为变元. 它所产生的方程系统的次数随时钟而递增. 另一种是每个时钟增加两个新的变元和三个方程,这样做的优点是保持整个系统的稀疏性,并且每个方程的最大次数不超过 2. 对 Gröbner 基算法来说,前一种方法有更快的求解时间[10],其产生的方程系统如下:

$$\begin{aligned} &s_{66} + s_{93} + s_{162} + s_{177} + z_1 = 0; \quad \text{1st clock} \\ &s_{65} + s_{92} + s_{161} + s_{176} + z_2 = 0; \quad \text{2nd clock} \\ &s_{64} + s_{91} + s_{160} + s_{175} + z_3 = 0; \quad \text{3rd clock} \\ &\cdots\cdots \end{aligned} \tag{1}$$

3 变元猜测的攻击方法与统计模型

3.1 算法描述

由于在代数攻击中,对于大的方程系统,很难直接求解. 所以,一般采用先猜测 n 个变元,然后再进行求解分析的方法. 具体如算法 2 所描述:

算法 2 GDAlgorithm(ES, n)

输入:环上方程系统 ES;猜测的变元数,n.
输出:0/1,表示是否找到解.

1: 选择 n 个变元的所有猜测集合置为 $GSet$
2: for vc in $GSet$ do
3: 把 vc 中变元的值代入方程系统 ES,得到 $ES(vc)$
4: if $ES(vc)$有解
5: 记录 vc 的值
6: return l
7: end if
8: end for
9: return 0

在算法 2 中,对于猜测 n 个变元,有 2^n 种猜测组合,每种猜测值代入后,产生一个方程子系统. 假如原方程系统有唯一解,则在 2^n 种猜测组合中,只有一个是正确的. 设 $\bar{t}$ 表示求解一个错误猜测子系统的平均时间,t' 表示求解一个正确猜测子系统的时间. 在最坏情况下,找到正确解所要花费的时间是:

$$T = (2^n - 1)\bar{t} + t', \bar{t} = \frac{1}{2^n - 1}\sum_{i=1}^{2^n-1} t_i \tag{2}$$

然而,在现实中,当猜测的变元数 n 很大时,如 $n = 32$,则要分别计算 2^{32} 数量级的子系统方程. 当每个子方程系统的求解时间较长时,(如超过 5 分钟),则很难统计出每个子系统的求解时间 t_i,从而计算出所有子系统的平均求解时间 $\bar{t}$,选择合适的统计模型是我们下面要解决的问题.

3.2 统计模型

当猜测的变元数 n 很大时,一般在 2^n 个子系统方程中选择 k 个,用这个 k 个子系统方程的平均求解时间来估计 2^n 个子系统的平均求解时间 $\bar{t}$[12]:

$$\bar{t} \approx \frac{1}{k}\sum_{i=1}^{k} t_i \tag{3}$$

然而,猜测变元的汉明重量(猜测变元的非零值数目)对于求解时间有重要的影响. 一般来说,汉明重量越小,则求解时间越快. 这是因为大部分的变元都被零值取代,从而使整个方程系统更为稀疏,平均而言,每个方程的次数也更小,从而更容易求解.

所以,随机选取 k 个子系统方程很难准确的估计 2^n 个子系统方程的平均求解时间. 我们采用根据不同汉明重量猜测值的比率,在整个猜测空间中选取. 假如 n 个猜测变元的汉明重量为 $d(0 \leqslant d \leqslant n)$,则其所占的比率为 $R_d = C_n^d/2^n$. 图 1 给出了猜测变元 $n = 42$ 时的不同汉明重量猜测值的比率比率分布.

由图 1 可见,汉明重量接近 $n/2$ 的猜测占了整个猜测空间的绝大部分. 因此,在实验中,我们按照如下步骤进行计算:

(1) 对所选取的 k 个子系统方程,汉明重量为 $d(0 \leqslant d \leqslant n)$的猜测应占到 $n_d = [kR_d]$个.

(2)对每个汉明重量 $d(d = 0,1,\cdots,n)$,计算求解平均值$\overline{t_d}$:

$$\overline{t_d} = \frac{1}{n_d}\sum_{i=1}^{n_d} t_{di} \tag{4}$$

其中 t_{di}是猜测汉明重量为d 的一个方程子系统的求解时间.

(3)整个 2^n 个子系统的平均求解时间$\bar{t}$:

$$\bar{t} \approx \sum_{d=0}^{n} R_d \overline{t_d} \tag{5}$$

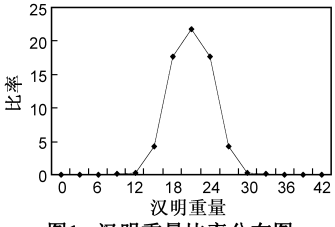


图1　汉明重量比率分布图

4　对猜测变元的选取

在实验中,对于特定的猜测变元数 n,选取不同的猜测变元,对于平均求解时间 $\bar{t}$ 有很大影响.如何选取使求解效率更高的变元去猜测,是我们下面要讨论的问题.这里,分两大类去讨论.其中基于权值的策略不仅对于 Bivium 流密码,对于任何布尔方程组均适合.

4.1　基于权值的选择策略

对于每个方程系统中的变元,我们可以以一定的策略统计所有变元的权值,然后从中选择权值最大的 n 个变元去猜测.根据权值计算方法的不同,分为静态权值与动态权值两种.

4.1.1　静态权值

静态权值即只是在变元替换前,对整个系统中的每个变元计算一次权值,然后选择权值最大的 n 个变元进行猜测.一种直接的方法是按照变元在整个方程系统中出现的次数作为该变元的权值,这样选出猜测的 n 个变元即为方程系统中出现变元次数最多的 n 个变元.

在实验中,我们采用了更为科学的权值计算方法.因为每个变元的重要性不仅与其出现次数有关,而且与其出现位置有关.从解方程得角度来说,出现在高次项中的变元更值得去猜测,因为这样可以更大限度的降低该多项式的次数,使整个方程系统更容易求解.鉴于此,我们把单项式的次数作为出现在单项式中每个变元的权值,然后再对整个方程系统中所有出现该变元的权值进行累加.最后选择权值最大的 n 个变元进行猜测.

举例来说,对于多项式 $s_1 + s_1s_2 + s_1s_2s_3$,由于 s_1 分别出现在三个单项式里,所以其权值为 $1 + 2 + 3 = 6$,同理,s_2, s_3 的权值为 5,3.

4.1.2　动态权值

静态权值计算简单,但却没有反映出变元在猜测过程中的重要性变化.实际上,对于单项式 $s_1s_2\cdots s_m$ 来说,一旦其中某个变元 s_i 作为猜测变元,则 $s_j(j \neq i)$的重要性立刻降低了.举例来说,对于多项式 $s_1s_2s_3 + s_4s_5s_6$,若 s_1 作为猜测变元,则 s_2, s_3 或者不会再出现$(s_1 = 0)$,或者变为二次的单项式 s_2s_3,则此时,选择 s_4, s_5, s_6 中的任一变元作为下一个猜测变元无疑会更大限度的降低整个多项式的次数.因此,我们设计了动态权值的计算方法:第一个猜测变元依据静态权值的计算方法进行选取,即选择初始权值最大的那个变元.一旦该变元确定后,我们重新计算剩余各个变元的权值,更新权值表,然后选择下一个权值最大的变元.

权值的更新策略是依据概率来进行的.如对于单项式 $s_1s_2s_3$,若 s_1 作为猜测变元,则 s_1 为 0 或 1 各有 50% 的概率,这样 s_2 和 s_3 的权值就等于 $0 \times 0.5 + 2 \times 0.5$,即为 1.我们在算法 3 对动态权值进行更细致的描述.

算法 3　DynamicWeightAlgorithm(ES, n)
输入:环上方程系统 ES;猜测的变元数 n.
输出:$list_2$ 中选定的 n 个猜测变元.

```
1: while 1 do
2:   if | list2 | = n then// list2 中存放最终选择的 n 个变元
3:       return list2;
4:   end if
5:   对所有的变元权值置 0,结果存在 list1 中
6:   for ES 中的每个单项式 mi = s1s2…sj do
7:       for mi中的每个变元 v do
8:           if v 在 list1 中 then
9:               v 的权值增加 x × 0.5^y. 这里, x 是 mi 中属于 list1 中
         的变元个数, y 是 mi 中属于 list2的变元
10:          end if
11:      end for
12:  end for
13:  把当前 list1 中权值最高的变元移到 list2
14: end while
```

这里,第一次循环由于 $list_2$ 为空,所以选出的是按静态权值策略计算出的最大权值的变元.随后,一旦最大权值的变元被选出,与该变元在同一单项式的变元的权值都会降低,然后,在动态变化的基础上,选择权值次大的变元,直到选择 n 个变元为止.

4.2　面向寄存器的选择策略

对于 Bivium 流密码来说,我们还可以从其生成密钥流的寄存器结构来选择具体猜测变元的位置.我们

知道,Bivium 流密码的方程系统是随时钟产生的.由于加/解密效率的原因,在每一个时钟周期,并非所有的变元都参与了方程系统的生成.因此,原则上,我们选择最先参与生成方程系统的变元或对方程系统次数有重要影响的变元.

对于 Bivium 流密码,令两个寄存器中的从前到后的变元分别为 $s_1, s_2 \cdots, s_{93}$ 和 $s_{94}, s_{95}, \cdots s_{177}$.根据算法 1,可总结出 Bivium 流密码的以下特征:

(1)首先,存贮在两个寄存器后面部分的变元更早的参与了方程系统的生成,如 s_{93}, s_{66}, s_{177} 和 s_{162} 出现在第一个时钟周期所生成的等式;而变元 s_1, s_{94} 分别在第 66 和 69 个时钟周期才出现.

(2)非线性的单项式都来自于两个寄存器的倒数 2,3 个变元.

(3)所有的非线性的单项式,其变元的下标都是连续的,即形如 $s_i s_{i+1} s_{i+2} \cdots\cdots$

依此三条特征,表 1 给出了一些最优猜测变元的位置.由特征 1,2,我们可以选择每个寄存器的最后变元进行猜测,具体包括 End1,End2,End0;由特征 1,2,3,可以每间隔 2、3 个变元进行猜测,也即 Ery2,Ery3.

表 1 猜测变元位置说明

Ery3	从最后交替的每隔 2 个变元选下一个作为猜测变元
Ery2	与 Ery3 类似,每隔 1 个变元选下一个作为猜测变元
End1	选择第一个寄存器后的若干变元作为猜测变元
End2	选择第二个寄存器后的若干变元作为猜测变元
End0	同时选择两个寄存器后的若干变元作为猜测变元
SW	静态权值策略
DW	动态权值策略

5 Gröbner 基算法中对变元序的选择

对一个多项式方程组 F 来说,其 Gröbner 基与该方程系统有同样的解.但是,由于 Gröbner 基算法的消去属性,其最终结果中会有一个多项式是单变元的.于是,该变元的解可以很方便得到;随后,我们把该变元的值代入到求出的 Gröbner 基中,于是,我们又能获得只含一个变元的多项式,依次类推,直至求出所有变元的解.实际上,Gröbner 基算法可以看作一般化的高斯消元算法.对于 Bivium 密码系统来说,由于其方程组只含有一个解,于是可以通过计算出既约 Gröbner 基来求出方程系统的解.既约 Gröbner 基形如 $G = \{x_1 + s_1, \cdots, x_n + s_n\}$,则 $(s_1, \cdots, s_n)$ 即为原方程系统的解.

Gröbner 基算法的最快的两个实现是 Faugère 的 F_4 与 F_5 算法,当前比较流行的代数分析软件中基本都实现了 F_4 算法.在实验中,我们选择了当前最流行的三款软件 Singular[13],MAGMA[14] 以及 PolyBoRi[15],其内嵌的 Gröbner 基算法算法都是基于 F_4 算法或 F_4 算法的变形.由于 Bivium 密码系统方程相对较稀疏,比起随机方程系统更容易求解.

对 Gröbner 基算法而言,有两个序非常重要.一个是变元序,即规定哪个变元优先级高;一个是单项式序,规定了整个多项式的单项式排列顺序.变元序一般分字典序,反字典序;而单项式序一般有字典序,分次字典序,分次反字典序等.分次反字典序被公认为求解 Gröbner 基的最快的序.对于 Bivium 密码系统,我们给出了 10 种变元序,如表 2 所示.前 6 种序我们参考了文献[10].实验表明(见第 7 节),应该选择那种出现不频繁或不太重要的变元先处理,这样可以保持整个方程系统在计算过程中不会膨胀的太快.

表 2 Bivium 密码系统变元序说明

AB	A,B 寄存器中最后一个变元的序最高,A 比 B 序高
S	对 AB 寄存器而言,最高序在后面,低序在前面
Freq	按变量的频率排序,这里颜色最深的部分序最高
SW	静态权值策略
DW	动态权值策略
* -rev	表示各种序的逆序

6 矛盾等式

在选择 n 个猜测变量后,首先应该检查猜测后的方程系统中是否存在矛盾等式.利用这些矛盾等式,我们可以缩小变元的猜测空间;而忽略了矛盾等式,有可能得到一个不正确的时间统计.

对于方程 $f(s_1, s_2, \cdots, s_t) = 0$ 来说,如果其中的所有变元均被猜测,且所到的等式为"1 = 0",则该等式称为**矛盾等式**.一旦方程系统中发现矛盾等式,可以立刻推断出,此次猜测是错误的,不必要再计算猜测后整个系统的 Gröbner 基,考虑下一组猜测即可.

对于一个密码系统来说,如果猜测变元覆盖了该密码系统的部分线性方程,我们把该部分线性方程简记为 *GLE*(Guessing Linear Equations),则首先分析该 *GLE* 中有可能产生的冲突等式的比率.由线性代数中对线性方程组的解空间理论可知,对于一个含有 n' 个变元,秩为 m 的线性方程组来说,其解的个数或为 $2^{n'-m}$,或 0 个解.

假设总共猜测 n 个变元,其中覆盖到的线性方程中有 n' 个变元.对于包含这 n' 个变元的线性方程组 *GLE*,若其秩为 m,则共有 $2^{n'-m}$ 个解.而对于所有的 2^n 个猜测组合,理论上有 $2^{n-n'} \times 2^{n'-m} = 2^{n-m}$ 个猜测组合满足 *GLE*.

假设在平均情况下,需要时间 t' 去生成一个满足 *GLE* 的猜测,然后设求解满足 *GLE* 的子系统的平均求解时间 $\bar{t}$.则猜测 n 个变元的求解时间如式(6)表示:

$$T = 2^{n-m}\bar{t} + 2^{n-m}t' \tag{6}$$

由于 t' 远小于 $\bar{t}$,所以 T 近似表示为 $2^{n-m}\bar{t}$.也就是

说,对于猜测 n 个变元,其中覆盖到的线性方程中有线性无关的方程 m 个,则通过预先对矛盾等式的判断,排除了 $2^n - 2^{n-m}$种导致矛盾等式的猜测组合,才能正确的对 $\bar{i}$ 进行估计,也即整体求解时间 T 的估计.

7　实验结果

所有实验均是在一台双核(2.6GHz 每核)8G 内存的机器上完成的.时间统计模型采用的是 3.2 节的概率统计模型.对于每次猜测特定数目的变元,我们都进行了约 1000 次的猜测组合.

7.1　实验工具的对比

首先在当前最流行的三款软件 Singular,MAGMA 以及 PolyBoRi 中,选择最快的 Gröbner 基实现算法.这里,不失一般性,我们选择了 Ery3 位置进行猜测,猜测了 58 个变元,在排除了矛盾等式的情况下,共进行了 1000 次猜测组合,并对结果按概率模型进行了统计,结果如表 3.

表 3　猜测 58 个变元在各种变元序及三种计算工具下的平均计算时间

算法	十种变元序下的平均计算时间									
	AB	AB-rev	S	S-rev	Freq	Freq-rev	SW	SW-rev	DW	DW-rev
slimgb	35.23	32.18	36.57	34.98	35.17	33.68	37.39	35.36	35.57	34.33
PolyBoRi	3.225	2.958	3.197	2.891	3.302	3.235	3.278	3.225	3.354	2.764
magama	9.4	8.1	11.20	10.31	11.12	10.09	12.51	11.57	10.2	9.96

由于 PolyBoRi 是专用的求解布尔方程组的工具,所以效率最高,其底层的多项式采用二元决策图表示,相对于另外两款求解软件,有明显的求解优势.所以,下面的实验均采用 PolyBoRi 计算工具.

7.2　变元序与猜测位置的确定

我们使用 PolyBoRi 软件,在 10 种变元序下,选择了 7 种猜测策略进行分析.同样猜测了 58 个变元,在每种序与猜测位置下均生成了 1000 次猜测组合,并对结果按概率模型进行了统计,结果如表 4:

表 4　猜测 58 个变元在 10 种变元序及 7 种猜测位置下的平均计算时间

猜测位置	十种变元序下的平均计算时间									
	AB	AB-rev	S	S-rev	Freq	Freq-rev	SW	SW-rev	DW	DW-rev
Ery2	0.041	0.019	0.040	0.027	0.022	0.027	0.018	0.018	0.022	0.021
Ery3	3.225	2.958	3.197	2.891	3.302	3.235	3.278	3.225	3.354	2.764
End0	0.729	0.428	0.540	0.416	0.606	0.596	0.584	0.574	0.714	0.491
End1	2.333	1.875	5.298	1.407	5.781	5.755	5.788	5.786	4.525	1.378
End2	1.595	0.685	1.456	0.619	1.886	1.813	1.863	1.821	1.374	0.535
SW	2.737	1.864	5.583	1.689	5.732	5.414	5.350	5.281	4.198	1.942
DW	0.298	0.175	0.198	0.220	0.245	0.222	0.230	0.189	0.184	0.148

由表 4 数据,我们可以推出以下结论:

(1)对各种序而言,反序普遍优于正序,因为反序下,首先选择那种出现不频繁或不太重要的变元先处理,这样可以保持整个方程系统在计算过程中不至膨胀的太快.

(2)相对而言,最佳的序是 DW-rev,可以推断,对变元在整个方程系统中的重要性进行有效分析,并按照重要性逐渐降低得顺序去逐个消去,对求解效率有很大影响.

(3)仅从时间来看,最佳猜测位置是 Ery-2.但是在 Ery-2 位置猜测的 58 个变元并没有覆盖一个线性方程,从而无法减少猜测空间.以表 5 来看,尽管 Ery3 位置的计算时间最长,但是其猜测位置覆盖了 20 个线性方程.从而大大减少了猜测空间,即 2^{58}变为 2^{38},对于其他($2^{58} - 2^{38}$)种猜测必然是不满足 20 个线性方程的.

表 5　猜测 58 个变元在 7 种猜测位置下的最快平均计算时间及总体求解时间估计

猜测位置	Ery3	End2	End1	End0	Ery2	SW	DW
最快时间(秒)	2.764	0.535	1.378	0.428	0.018	1.689	0.148
GLE 个数	20	0	0	2	0	4	3
求解时间估计(秒)	2^38 * 2.764	2^58 * 0.535	2^58 * 1.378	2^56 * 0.428	2^58 * 0.018	2^54 * 1.689	2^55 * 0.148

(4)对于随机多变元方程组,只能采用 SW 或 DW 的猜测方式.而 DW 相对于 SW 更有优势,为随机多变元方程组求解中的变元猜测给出了确定的方法.然而,对于 SW 或 DW 的猜测方式中的权值的大小该如何定义,是我们后续要解决的问题.即要分析出对于特定的方程系统,变元的出现次数更重要还是其幂次越高越重要.

7.3　变元猜测个数的确定

在确定了算法,序,猜测位置后,需要对不同的变元猜测个数分别做出时间复杂度的估算,我们选择猜测 62,60,58,56,54 个变元.在 Ery3 猜测位置以及 DW-rev 序下,依概率模型进行了 1000 次猜测组合,并对结果进行了统计.

由表 6 可知,最佳猜测变元数是 60 个变元,其计算时间约为 $2^{39.16}$秒.相对于以前的结果,如使用 SAT 求解器的攻击,约 $2^{41.7}$秒[3];使用图论法的攻击,约 2^{56}秒[16];使用二元决策图(BDD)的攻击约 $2^{50.1}$秒[17],我们的结果有了较大的提高.

表 6 猜测不同个数变元的平均计算时间及总体求解时间估计

猜变元数	62	60	58	56	54	52
GLE 个数	22	21	20	19	18	17
时间 t	0.91	1.11	2.76	6.41	16.12	35.14
时间 T	$2^{39.80}$	$2^{39.16}$	$2^{39.38}$	$2^{39.68}$	$2^{40.01}$	$2^{40.14}$

8 结束语

在代数攻击中,对多变元方程组的求解是一个 NP 问题.通过先猜测部分变元,把整个方程系统进行分割,然后选择一个子集进行求解分析,可以有效的估计整个方程系统的求解复杂度.本文使用了概率分布的方法去估计对子系统求解的时间平均值.对于猜测变元的选择,本文提出了静态权值与动态权值的方法,其中动态权值的方法比较有竞争力,为对于一个随机方程组系统的猜测变元选择提供了理论准则.同时,提出了矛盾等式的概念,这对正确分析求解结果以及缩小猜测空间有一定的价值.最后,我们通过实验给出了新的对 Bviuim 攻击的结果约为 $2^{39.16}$秒,研究更优的猜测策略以及并行求解是我们下一步工作的重点.

参考文献

[1] Nicolas T Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of toyocrypt[A]. ICISI 2002[C]. Berlin: Springer-Verlag, 2002. 182 – 199.

[2] Nicolas T Courtois, Willi Meier. Algebraic attacks on stream ciphers with linear feedback[A]. EUROCRYPT 2003[C]. Berlin: Springer-Verlag, 2003. 345 – 359.

[3] C McDonald, C Charnes, J Pieprzyk. Attacking bivium with minisat[A]. SAT′08 Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing[C]. Berlin: Springer-Verlag, 2008. 63 – 76.

[4] 谢佳,王天择. 寻找布尔函数的零化子[J]. 电子学报, 2010, 38(11): 2686 – 2690.
Xie Jia, Wang Tianze. Finding the annihilators of a boolean function[J]. Acta Electronica Sinica, 2010, 38(11): 2686 – 2690. (in Chinese)

[5] Nicolas Courtois, Adi Shamir, et al. Efficient algorithms for solving overdefined systems of multivariate polynomial equations[A]. EUROCRYPT′00 Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques[C]. Berlin: Springer-Verlag, 2000. 392 – 407.

[6] F Chai, X S Gao, et al. A characteristic set method for solving boolean equations and applications in cryptanalysis of stream ciphers[J]. Journal of Systems Science and Complexity, 2008, 21(2): 191 – 208.

[7] Faugère J C. A new effcient algorithm for computing Gröbner bases (F4)[J]. Pure Appl Algebra, 1999, 139: 61 – 88.

[8] Faugère J-C. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)[A]. ISSAC'02: Proceedings from the International Symposium on Symbolic and Algebraic Computation[C]. Berlin: Springer-Verlag, 2002. 75 – 83.

[9] Jean-Charles Faugère, Antoine Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using groebner bases[J]. Advances in Cryptology, 2003, 2729: 44 – 60.

[10] Tobias Eibach, Gunnar Volkel, Enrico Pilz. Optimising Gröbner bases on bivium[J]. Mathematics in Computer Science, 2010, 3: 159 – 172.

[11] Baiqiang Chen. Strategies on algebraic attacks using sat solvers[A]. ICYCS 2008[C]. Zhang Jia Jie, China, 2008. 2204 – 2209.

[12] Tobias Eibach, Enrico Pilz. Attacking bivium with sat solvers[A]. SAT2008[C]. Springer-Verlag Berlin, 2008. 63 – 76.

[13] Greuel, G-M, Pfister, et al. Singular 3.0.4. A computer algebra system for polynomial computations[OL]. http://www.singular.uni-kl.de, 2010-01-16/2010-02-04.

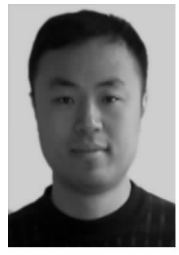
[14] Bosma, Cannon, et al. MAGMA computational algebra system. Vers. 2.14 10[OL]. http://magma.maths.usyd.edu.au/magma/, 2010-01-16/2010-02-04.

[15] Brickenstein, M, Dreyer, A. PolyBoRi: A framework for grobner basis computations with Boolean polynomials[OL]. www.ricam.oeaw.ac.at/mega2007/electronic/26.pdf, 2010-01-16/2010-02-04.

[16] H Raddum. Cryptanalytic results on trivium[OL]. www.ecrypt.eu.org/stream/papersdir/2006/039.ps, 2010-01-16/2010-02-04.

[17] A Maximov, A Biryukov. Two trivial attacks on trivium[A]. Selected Areas in Cryptography[C]. Berlin: Springer-Verlag, 2007. 36 – 55.

作者简介



李 昕 男,1978 年生于江苏常州.中国科学院软件研究所信息安全国家重点实验室博士生.研究方向为信息安全与密码学.
E-mail: groebner@126.com

林东岱 男,1964 年生于山东聊城.中国科学院软件研究所,信息安全国家重点实验室研究员,博士生导师.研究方向为信息安全、密码理论、安全协议、分布式密码计算等.