doi:10.11887/j.cn.201502020 http://journal.nudt.edu.cn

二次检测立方攻击改进与实现*

王永娟[1],丁立人[1],任泉宇[1],杨 程[2]
(1. 解放军外国语学院 语言工程系,河南 洛阳 471003; 2. 国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:对二次检测立方攻击预处理阶段的提取二次表达式的算法进行了改进以优化攻击效率。将秘密变量的变化引入攻击中,使得攻击模型更加灵活;同时,利用时空折中的思想,通过存储常数项和一次项的计算结果,有效降低二次项的计算量。将改进的方法应用于简化版的PRESENT算法和Trivium算法上,攻击效率有显著提高。

关键词:立方攻击;二次检测;时空折中;改进
中图分类号:TP309.7 **文献标志码**:A **文章编号**:1001-2486(2015)02-106-06

Enhancement and application of cube attack with quadratic test

WANG Yongjuan[1], *DING Liren*[1], *REN Quanyu*[1], *YANG Cheng*[2]
(1. Department of Language Engineering, PLA University of Foreign Languages, Luoyang 410073, China;
2. College of Computer, National University of Defense Technology, Changsha 471003, China)

Abstract: The algorithm of extracting quadratic expressions in the pre-processing phase of cube attack with quadratic test was enhanced to optimize the attack efficiency. The variation of secret keys was introduced into cube attack, which makes the model much more flexible. At the same time, with the help of the trade-off between time and space, the complexity of extracting quadratic terms was reduced by storing the results of the constant and linear terms. The improved method was applied to the simplified PRESENT and Trivium algorithms and it turns out that the attack efficiency is enhanced obviously.

Key words: cube attack; quadratic test; trade-off between time and space; enhancement

2003年,Courtois和Meier[1]针对序列密码提出代数攻击,基本原理是将密钥恢复归约为超定多变元非线性方程组的求解。代数免疫成为了衡量布尔函数性质的重要密码学指标,具有最优代数免疫的各类布尔函数[2-3]相继提出。

2008年,Dinur和Shamir[4]提出了一种基于代数思想的新攻击方法,即立方攻击。它属于确定性选择明文攻击,通过对公开变量和秘密变量赋值,获得关于密钥的线性方程,从而将密钥恢复归约为线性方程组的求解。该攻击方法基于布尔函数的唯一代数标准型建立方程系统,彭昌勇等[5]在对高级加密标准(Advanced Encryption Standard, AES)进行分析时也利用了类似的思想。立方攻击可以被应用于序列密码、分组密码和哈希函数上,对Trivium[4,6-7]、Grain[8]、PRESENT[9-10]、AES[11]、KATAN[12]和MD6[13]等密码算法都有良好的应用效果,因此受到广泛关注。

然而,立方攻击在具体应用中也会受到一些因素的影响。如果密码体制主多项式的代数次数过高,预处理阶段就会消耗大量的时间和空间。从密码体制中提取的线性关系往往比较有限,这也限制了立方攻击的效率。

2010年,Mroczkowki等[6]首次将Dinur在文献[4]中提到的二次检测应用于对709轮Trivium算法的立方攻击中,可以恢复全部80比特密钥,其中39比特包含在二次方程中。2011年,Abdul-Latip等[9]针对提取低次非线性方程的算法提出了改进的立方攻击,将改进的立方攻击与旁道攻击相结合应用于PRESENT-80算法,恢复全部密钥的时间复杂度为2^{16}。2013年,Fouque等[7]利用莫比乌斯转换和二次检测对标准立方攻击进行改进,当Trivium算法的初始轮数为799时,利用关于密钥的二次方程将穷举攻击的复杂度由2^{68}降到2^{40}。

二次检测的引入虽然使立方攻击的模型更加灵活、高效,但是也增加了预处理阶段的复杂度。

* **收稿日期**:2014-05-26
基金项目:中国博士后科学基金面上资助项目(2014M552603)
作者简介:王永娟(1982—),女,河南开封人,博士,硕士生导师,E-mail: pinkywyj@163.com

如何有效提升二次检测立方攻击的实施效率,是在立方攻击的发展中值得研究的问题。

利用时空折中的思想对提取二次表达式的算法进行改进,通过存储表达式中常数项和一次项的计算结果简化二次项的计算。利用空间换取时间,提高二次检测立方攻击在预处理阶段提取二次表达式的效率。

Mroczkowki 在文献[6]中只给出了二次检测的结果,并没有阐述提取表达式的具体方法。Abdul – Latip 在文献[9]中提出了提取低次表达式的方法,其思路是在表达式通过二次(或低次)检测后先确定其中存在的变量,再判断变量的存在形式。确定表达式中存在的变量涉及大量随机检测,检测次数过小会发生遗漏;过大则会影响算法效率。本文提出的算法,不需要进行随机测试就可以直接判断变量在二次方程中的存在形式,提高了计算的准确率;同时,通过存储线性检测的计算结果,计算二次项的时间复杂度是标准算法的 1/4。

将改进的算法应用于简化版的 PRESENT 算法和 Trivium 算法上,同时简化 Abdul – Latip 算法中的计算条件(将随机检测的次数由文献[9]中的 300 次降为 100 次),将其应用到相同算法上,对二者的计算效率进行比较。对于简化的 PRESENT 算法和简化的 Trivium 算法,改进算法提取二次表达式的效率为 Abdul – Latip 算法的 4 至 5 倍。

1　攻击方法简介

1.1　立方攻击原理

把目标密码体制看成有限域 F_2 上关于公共变量 $v=(v_1,\cdots,v_m)$ 和秘密变量 $k=(x_1,\cdots,x_n)$ 的多项式 $P(v,k)$。给定指标集 $I=\{i_1,\cdots,i_k\}\subseteq\{1,2,\cdots,m\}$,令 $t_I=v_{i_1}\cdots v_{i_k}$,目标多项式分解如下:

$$P(v_1,\cdots,v_m,x_1,\cdots,x_n)=$$
$$t_I\cdot P_{S(I)}+Q(v_1,\cdots,v_m,x_1,\cdots,x_n)$$

称形如

$$C_I=\{(v_{i_1},\cdots,v_{i_k})\in F_2^k\mid v_i\in F_2,i\in I\}$$

的集合为一个 Cube 集合,$|C_I|=2^k$。遍历 Cube 集合,对目标多项式进行求和,可得:

$$\sum\nolimits_{C_I}P(v,k)=\sum\nolimits_{C_I}t_I\cdot P_{S(I)}+\sum\nolimits_{C_I}Q(v_1,\cdots,v_m,x_1,\cdots,x_n)。$$

定理 1[4]　任意目标多项式 $P(v,k)$ 和公共变量都满足:

$$\sum\nolimits_{C_I}P(v,k)=P_{S(I)}(v,k)\bmod 2$$

通过立方求和所得的 $P_{S(I)}(v,k)$ 为超级多项式,如果该超级多项式为线性多项式,则称对应的 t_I 为极大项。

定理 2[4]　令 t_I 为极大项,则:

① 将所有集合 I 之外的变量赋值为 0,在 C_I 上进行立方求和可以得到表达式的常数项;

② 将所有集合 I 之外的公共变量赋值为 0,将除第 i 位秘密变量外其余秘密变量赋值为 0,令 $x_i=1$,对所有可能的输入进行模 2 求和,可以得到 x_i 的系数。

根据定理 2,在 $P_{S(I)}(v,k)$ 通过线性化检测之后可以计算出具体的表达式。

立方攻击分为预处理阶段和在线阶段两个部分,预处理阶段的目的是通过为公共变量和秘密变量赋值,获取关于密钥的线性表达式;在线阶段只为公共变量进行赋值,计算表达式右边的值,得到关于密钥的线性方程组,进而联立求解。

1.2　二次检测原理

二次检测立方攻击,即在预处理阶段扩展集合的搜索范围,增加搜索关于密钥的二次表达式的步骤。二次检测的原理可由线性化检测推广而得。

1993 年,Blum 等[14]针对布尔函数提出了多种代数性质的检测方案。其中,对于线性性质,他们提出了 Blum–Luby–Rubinfeld(BLR)检测法,具体原理详见文献[14]。

定理 3[14]　如果 n 元布尔函数 $f(x)$ 是 2 次的,那么对于任意 $x,y,z\in F_2^n$,都满足:

$$f(x)\oplus f(y)\oplus f(z)\oplus$$
$$f(x\oplus y)\oplus f(x\oplus z)\oplus f(y\oplus z)\oplus$$
$$f(x\oplus y\oplus z)\oplus f(0^{(n)})=0 \qquad (1)$$

根据定理 3,为了判断布尔函数 $f(x)$ 是否为二次,攻击者需要任意选取 ω 对 $x,y,z\in F_2^n$,若都满足式(1),则称该函数通过二次检测。注意,通过二次化检测的函数满足 $\deg(f(x))\leqslant 2$。

2　改进的二次检测立方攻击

2.1　改进算法的原理

在研究 Cube 集合时,不仅考虑了公共变量的变化,还将秘密变量的变化引入其中。定义 1 给出了 Cube 集合在秘密变量上的扩张,本文在 Cube 集合扩张的基础上对二次检测立方攻击的预处理阶段进行改进。

定义 1 给定指标集：

$$I=\{i_1,\cdots,i_k\}\subseteq\{1,2,\cdots,m\},$$

$$J=\{j_1,\cdots,j_s\}\subseteq\{1,2,\cdots,n\},$$

C_I 为一个 Cube 集合，定义与 1.1 节中相同，称形如

$$C_{I\cup J}=\{(v_{i_1},\cdots,v_{i_k},x_{j_1},\cdots,x_{j_s})\in F_2^{k+s}\mid v_i\in F_2,i\in I;x_j\in F_2,j\in J\}$$

的集合为 C_I 在秘密变量上的扩张。

依据定义 1，将 Dinur 和 Shamir 关于主多项式和公共变量的定理 1 扩展到秘密变量上，结论仍然成立，即：

$$\sum\nolimits_{C_{I\cup J}}P(v,k)=P_{S(I\cup J)}(v,k)\bmod 2$$

扩展立方集合的搜索范围，搜索关于秘密变量的线性和二次表达式。下文阐述如何利用 Cube 集合在秘密变量上的扩张提取关于秘密变量的二次表达式。

定义 2 给定集合 I，若 $\deg(P_{S(I)})=2$，则称对应的 t_I 为 2 阶极大项。

定理 4 令 t_I 为 2 阶极大项，a_i 表示 $P_{S(I)}$ 中 x_i 的系数，a_{ij} 表示 $P_{S(I)}$ 中 x_ix_j 的系数，a_0 表示 $P_{S(I)}$ 中的常数项，则

① 令所有 $x_q=0,v_l=0,l\notin I$，有 $a_0=\sum\nolimits_{C_I}P(v,k)$；

② 令所有 $x_q=0,x_q\in k\backslash\{x_i\},v_l=0,l\notin I$，有 $a_i=\sum\nolimits_{C_{I\cup\{i\}}}P(v,k)$；

③ 令所有 $x_q=0,x_q\in k\backslash\{x_i,x_j\},v_l=0,l\notin I$，有 $a_{ij}=\sum\nolimits_{C_{I\cup\{i,j\}}}P(v,k)$；

证明：由定理 2 易得定理 4 的①和②成立；用 $t_J=\prod_{b=1}^{s}x_{j_b}$ 表示 s 个秘密变量 x_j 的乘积，如果 t_J 是超级多项式 $P_{S(I)}$ 中的单项式，那么将 $P_{S(I)}(v,k)$ 中所有不属于 t_J 的变量赋值为 0 后，t_J 就是 $P_{S(I)}(v,k)$ 中除常数项外唯一的单项式，在集合 $C_{I\cup J}$ 上进行求和，经过 $2^{|I|+|J|}$ 次运算后常数项抵消，只有当 $j_1=\cdots=j_s=1$ 时可得 $P_{S(I\cup J)}(v,k)=1$，令 $t_J=x_ix_j$，则有定理 4 中的③成立。 □

例 1 目标多项式

$$P(v_1,v_2,v_3,x_1,x_2,x_3,x_4)=v_1v_2v_3x_1x_2+v_1v_2v_3x_4+v_1x_3+v_1v_2v_3$$

令 $I=\{1,2,3\}$，进行二次检测可得 $t_I=v_1v_2v_3$ 为 2 阶极大项，$P_{S(I)}=x_1x_2+x_4+1$。下面根据定理 4 判断该表达式的代数标准型：

$$a_0=P_{S(I)}(v,k)\mid_{(0,0,0,0)}=1$$

$$a_4=P_{S(I)}(v,k)\mid_{(0,0,0,0)}\oplus P_{S(I)}(v,k)\mid_{(0,0,0,1)}=1$$

$$a_1=P_{S(I)}(v,k)\mid_{(1,0,0,0)}\oplus P_{S(I)}(v,k)\mid_{(0,0,0,0)}=0$$

$$a_2=P_{S(I)}(v,k)\mid_{(0,1,0,0)}\oplus P_{S(I)}(v,k)\mid_{(0,0,0,0)}=0$$

$$a_{12}=P_{S(I)}(v,k)\mid_{(1,0,0,0)}\oplus P_{S(I)}(v,k)\mid_{(0,1,0,0)}\oplus P_{S(I)}(v,k)\mid_{(0,0,0,0)}\oplus P_{S(I)}(v,k)\mid_{(1,1,0,0)}=1$$

由例 1 可知，在计算二次项 a_{12} 的系数，包括对 a_1,a_2 和 a_0 的计算时，如果分别存储 a_1,a_2 和 a_0 的计算结果，那么计算 a_{12} 时只需考虑指标索引全 1 的情况，而不用遍历 2 维 Cube 集合，从而将 4 次运算简化为 1 次运算。

基于上述现象，利用时空折中的思想改进二次表达式的提取。通过存储常数项和一次项的系数，简化二次项系数的计算，从而提高二次检测立方攻击预处理阶段的效率。下文阐述改进算法的理论依据：

定理 5 给定目标多项式 $P(v,k)$，对任意指标集合 S，集合 $R=\{I\mid I\subseteq S\}$，有

$$\sum\nolimits_{C_S}P(v,k)=\sum\nolimits_R\sum\nolimits_{C_I}P(v,k)\oplus[P(v,k)\mid_{v_1=\cdots=v_{|S|=1}}]$$

证明：令 $d=|S|$，则集合 S 的真子集个数为 2^d-1，即 $R=\{I_1,\cdots,I_{2^d-1}\}$。$C_{I_1}\cup\cdots\cup C_{I_{2^d-1}}=C_S\backslash\{(\underbrace{1,1,\cdots,1,0}_{|S|},0,\cdots)\}$，又 $(0,0,\cdots,0)$ 在求和时出现 2^d-1 次；$(1,0,\cdots,0)$ 在求和时出现 $2^{d-1}-1$ 次；$(1,1,\cdots,0)$ 在求和时出现 $2^{d-2}-1$ 次，依次类推，每项都出现奇数次，因此在 F_2 上异或值不变，故有：

$$\sum\nolimits_{C_S}P(v,k)=\sum\nolimits_{C_{I_1}}P(v,k)\oplus\cdots\oplus\sum\nolimits_{C_{I_{2^d-1}}}P(v,k)\oplus P(v,k)\mid_{v_1=\cdots=v_{|S|=1}}$$

$$=[\sum\nolimits_R\sum\nolimits_{C_I}P(v,k)]\oplus[P(v,k)\mid_{v_1=\cdots=v_{|S|=1}}]$$

□

依据定理 5，存储常数项和一次项的计算结果，可使二次项系数的计算量减小为之前的 1/4，而改进算法只需要比标准算法多开辟 $n+1$ 个存储空间（其中 n 为秘密变量个数）。

2.2 改进算法的流程

改进算法在表达式通过二次检测后依次计算其中的常数项、一次项系数以及二次项系数，并将常数项和一次项的计算结果存储在新开辟的数组中，以便计算二次项系数时直接调用。具体过程

如算法 1 所示。

算法 1　改进算法伪代码

Alg. 1　Pseudo-code of the improved algorithm

输入:n;//秘密变量的个数

I;//2 阶极大项对应的指标集合

$R[n+1]$; //开辟数组,存储计算结果

输出:$P_{S(I)}$;//二次表达式标准型

1. $a_0 = \sum_{c_I} P(v,k)$; $R[0] = a_0$;
2. **for** $i = 1$ *to* n
3. $\quad a_i = \sum_{c_{I\cup\{i\}}} P(v,k)$; $\quad R[i] = a_i$;
4. **end for**
5. **for** $i = 0$ *to* n
6. $\quad$ **if** $R[i] = 1$ **then** output i;
7. $\quad$ **end if**
8. **end for**
9. **for** $i = 1$ *to* $n-1$
10. $\quad$ **for** $j = i+1$ *to* n
11. $\quad\quad x_i = 1, x_j = 1$;
12. $\quad\quad a_{ij} = R[0] \oplus R[i] \oplus R[j] \oplus \sum_{c_I} P(v,k)$;
13. $\quad\quad$ **if** $a_{ij} = 1$ **then** output i, j;
14. $\quad\quad$ **end if**
15. $\quad$ **end for**
16. **end for**

2.3　复杂度分析

改进的算法计算常数项需要 $2^{|I|}$次计算,计算一次项系数需要 $n \cdot 2^{|I|}$次计算,利用时空折中思想的二次项的计算量由$[n(n-1)/2] \cdot 2^{|I|+2}$降低为$[n(n-1)/2] \cdot 2^{|I|}$。综上,改进算法的时间复杂度为$[n(n+1)/2+1] \cdot 2^{|I|}$,空间复杂度为 $n+1$。

Abdul – Latip 在文献[9]中提出先确定表达式中存在的变量,再判断变量的存在形式。该算法提取二次表达式的时间复杂度为 $\mu n 2^{|I|-1} + \sum_{i=0}^{2} 2^{|I|-2+i}\binom{N}{i}$,其中$\mu$ 为随机检测的次数,目的是确定表达式中存在的变量。随机检测的次数是决定该算法时间复杂度的关键因素之一,检测次数过小会发生遗漏;过大则会增加复杂度影响算法效率。下文对 2.1 节提出的改进算法与 Abdul – Latip 的算法进行时间复杂度和空间复杂度比较(N 为二次表达式中包含变量的个数,算法所需空间复杂度为 N,用于存储上述变量)。

时间复杂度比较:

$$\frac{[\frac{n(n+1)}{2}+1] \cdot 2^{|I|}}{\mu n 2^{|I|-1} + \sum_{i=0}^{2} 2^{|I|-2+i}\binom{N}{i}} \approx \frac{n(n+1)}{N(N+1)+\mu n};$$

空间复杂度比较:$\frac{n+1}{N}$。

由上述比较可得时间复杂度优化的临界值,即当$\mu > n+1$ 时,在有限的存储空间内,改进算法能有效提高提取二次表达式的效率。为了确保准确性,随机检测的次数一般不小于 200 次,而 Abdul – Latip 在文献[9]中选择 $\mu = 300$,理论上新方法将大幅提升二次表达式的提取效率,本文将在第 3 节给出实验验证。表 1 和表 2 分别比较了标准算法、Abdul – Latip 的算法和提出的改进算法在时间复杂度和空间复杂度上的对比。

表 1　各算法时间复杂度对比

Tab. 1　Comparison of time complexity

算法	时间复杂度
标准算法	$(n+1)2^{\|I\|}+n(n-1)2^{\|I\|+1}$
Abdul – Latip[9]	$\mu n 2^{\|I\|-1}+\sum_{i=0}^{2}2^{\|I\|-2+i}\binom{N}{i}$
改进算法	$[n(n+1)/2+1] \cdot 2^{\|I\|}$

表 2　各算法空间复杂度对比

Tab. 2　Comparison of space complexity

算法	空间复杂度
标准算法	无
Abdul – Latip[9]	N
改进算法	$n+1$

3　对 PRESENT 和 Trivium 的实验结果

使用配备 i – 5 处理器、1.7GHz 主频、2GB 内存的笔记本电脑,对 3 轮的 PRESENT–80 算法和初始化轮数为 576 的 Trivium 算法进行二次检测立方攻击,利用改进算法提取二次表达式,验证其效率。

3.1　PRESENT 算法上的应用

3.1.1　PRESENT 算法简介

PRESENT[15]算法是基于硬件实现设计的轻量级分组密码,分组长度为 64 比特,按密钥长度及生成过程不同分为 PRESENT–80 和 PRESENT –128 两个版本。该算法基于 SPN 结构,迭代 31 轮,其中每轮都包括三层子函数。第一层为轮密钥函数,输入数据与该轮密钥进行异或;出于硬件实现的考虑,第二层平行使用 16 个相同的 4 比特 S 盒构成非线性代换层;第三层为线性置换层,基于比特置换实现扩散。

3.1.2　效率分析

对于 3 轮的 PRESENT–80 算法,选取 2 次和

3 次 2 阶极大项,分别利用 Abdul – Latip 的算法和改进算法提取二次表达式(其中在 Abdul – Latip 的算法中令 $\mu = 100$),比较时间以验证改进算法的效率。检测结果如表 3 所示,其中时间单位为 s,第 2 列为 Abdul – Latip 的算法所耗时间,第 3 列为本文改进算法所耗时间。

表 3 在 PRESENT-80 上的应用对比

Tab. 3 Comparison on PRESENT-80

集合	时间[9](s)	改进时间(s)
{7,24}	1.981	0.468
{7,43}	2.199	0.484
{17,33}	2.091	0.483
{36,59}	1.997	0.468
{7,41,42}	3.884	0.905
{8,21,22}	3.385	0.890
{25,43,53}	3.510	0.904
{25,41,52}	3.557	0.904

如表 3 所示,即使令文献[9]中随机检测的次数为原文的 1/3,利用改进算法提取二次表达式的效率仍为 Abdul – Latip 的 4 倍。而且,改进算法中不存在随机检测过程,准确性也高于 Abdul – Latip 的算法。

PRESENT 算法提出后受到了广泛关注,2009 年,YANG 等[16]利用旁道攻击对其进行分析,在空间复杂度 2^{15} 之内恢复 PRESENT-80 算法的 48 比特密钥。2011 年Abdul-Latip等[9]将二次检测立方攻击与旁道攻击结合,直接恢复PRESENT-80算法的 64 比特密钥。本文将改进的二次检测立方攻击与旁道攻击结合,通过搜索二次表达式,在文献[10]的基础上,能恢复原文无法恢复的 8 比特密钥,以时间复杂度 2^{15} 和空间复杂度 2^{15} 恢复全部 80 比特密钥。

表 4 比较了各分析方法对 PRESENT-80 算法的攻击结果。

表 4 对 PRESENT-80 的攻击

Tab. 4 Attack on PRESENT-80

攻击方法	密钥(bit)	明文空间	二次检测时间复杂度
旁道攻击[16]	48	2^{15}	无
旁道二次立方[9]	64	2^{13}	2^{17}
旁道立方[10]	72	2^{15}	无
本文	80[9-10]	2^{15}	2^{15}

3.2 Trivium 算法上的应用

3.2.1 Trivium 算法简介

Trivium[17]是 eSTREAM 计划的候选算法之一,初始向量长度为 80 比特,初始密钥长度为 80 比特,空跑 1152 轮后输出密钥。它由三个不同长度的非线性移位寄存器构成,内部状态共 288 比特。其中,每个移位寄存器的反馈都由其他寄存器比特的非线性组合与该寄存器比特进行异或而得。每轮的输出是 6 个比特的线性组合,每个寄存器选 2 个比特。

3.2.2 效率分析

对于初始轮数为 576 的 Trivium 算法,选取 6 次 2 阶极大项,分别利用改进算法和 Abdul – Latip 的算法提取二次表达式(其中在 Abdul – Latip 的算法中令 $\mu = 100$),比较时间以验证改进算法的效率。检测结果如表 5 所示,时间单位为 s,第 2 列为 Abdul – Latip 的算法所耗时间,第 3 列为本文改进算法所耗时间。

表 5 在 Trivium 上的应用对比

Tab. 5 Comparison on Trivium

集合	时间[9](s)	改进时间(s)
{7,20,43,57,60,77}	525.549	117.109
{4,8,19,30,35,59}	533.427	116.844
{2,14,36,45,60,63}	520.152	114.676
{12,20,48,71,72,77}	515.255	113.443
{18,30,32,60,67,75}	553.130	113.319
{8,22,26,57,63,65}	535.129	113.006

如表 5 所示,即使简化 Abdul – Latip 的算法,改进算法在 Trivium 上的应用效率仍接近原来的 5 倍。

立方攻击和二次检测立方攻击提出伊始都被应用于 Trivium 算法上,且具有良好的攻击结果。2008 年,Dinur 和 Shamir[4]利用立方攻击直接恢复了 672 轮 Trivium 算法 63 比特密钥。2012 年,Mroczkowski[6]首次将二次检测的思想应用于立方攻击中,恢复 709 轮 Trivium 算法全部密钥,其中二次检测所耗时间复杂度为 2^{37}。2013 年,Fouque[7]将二次检测立方攻击应用于 799 轮的 Trivium 算法上,直接恢复 40 比特密钥,其中二次检测的时间复杂度约为 2^{51}。利用本文提出的改进算法,搜索 37 次 2 阶极大项的时间复杂度约为 2^{48}。

表 6 比较了各分析方法对 Trivium 算法的攻

击结果。

表 6　对 Trivium 的攻击

Tab. 6　Attack on Trivium

攻击方法	轮数	密钥(bit)	二次检测时间复杂度
立方攻击[4]	672	63	无
二次检测立方[6]	709	80	2^{37}
二次检测立方[7]	799	40	2^{51}
本文	709	80[6]	2^{34}
	799	40[7]	2^{48}

综上所述,改进的算法在 PRESENT 算法和 Trivium 算法上的实验结果符合对于算法复杂度的理论估计。在立方攻击时利用二次检测可以恢复更多密钥比特,改进的算法能够优化提取关于密钥的二次表达式的效率。

4　结论

利用低次关系(不单是线性关系)恢复密钥已经成为立方攻击发展的一大趋势,低次表达式的提取因此也成为该领域的研究热点。本文将秘密变量的变化引入立方攻击中,基于时空折中的思想提出了提取二次表达式的改进算法,主要思想是对常数项和一次项的计算结果进行存储,以减少二次项系数的计算量,从而降低二次检测立方攻击预处理阶段的时间复杂度。

理论上,改进算法与 Abdul-Latip 在文献[9]中提出的算法相比,效率至少为后者的$\mu/n+1$倍。当文献[9]中的随机检测次数$\mu=100$时,改进算法在简化的 PRESENT-80 算法和简化的 Trivium 算法上的应用效率为其 4 至 5 倍。

改进的二次检测立方攻击有较为广阔的应用前景,在今后的工作中也可参照该算法的改进模式,利用时空折中的思想优化对其他低次表达式的提取。

参考文献(References)

[1] Courtois N T, Meier W. Algebraic attacks on stream ciphers with linear feedback [C]//Advances in Cryptology-EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Springer Berlin Heidelberg, 2003, 2656: 345 - 359.

[2] 李超, 薛朝红, 付绍静. 代数免疫度最优的旋转对称布尔函数的构造[J]. 国防科技大学学报, 2012, 34(2): 34 - 38.
LI Chao, XUE Chaohong, FU Shaojing. Construction of rotation symmetric Boolean function with maximum algebraic immunity [J]. Journal of National University of Defense Technology, 2012, 34(2): 34 - 38. (in Chinese)

[3] 董德帅, 李超, 屈龙江, 等. 偶变元 MAI 旋转对称布尔函数[J]. 国防科技大学学报, 2012, 34(4): 85 - 89.
DONG Deshuai, LI Chao, QU Longjiang, et al. Rotation symmetric Boolean functions in even-variable with maximum algebraic immunity [J]. Journal of National University of Defense Technology, 2012, 34(4): 85 - 89. (in Chinese)

[4] Dinur I, Shamir A. Cube attack on tweakable black box polynomials [C]//Advances in Cryptology-EUROCRYPT 2009, Springer Berlin Heidelberg, 2009: 278 - 299.

[5] 彭昌勇, 祝跃飞, 康绯, 等. AES 轮变换的代数正规型及其应用[J]. 国防科技大学学报, 2012, 34(2): 14 - 17.
PENG Changyong, ZHU Yuefei, KANG Fei, et al. The ANFs of the component functions of AES round transformation and its application [J]. Journal of National University of Defense Technology, 2012, 34(2):14 - 17. (in Chinese)

[6] Mroczkowski P, Szmidt J. The cube attack on stream cipher Trivium and quadraticity tests [J]. Fundamenta Informaticae, 2012, 114(3): 309 - 318.

[7] Fouque P A, Vannet T. Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks [EB/OL]. (2013 - 12 - 14) [2014 - 05 - 06]. http://fse2013.spms.ntu.edu.sg:80.

[8] Dinur I, Shamir A. Breaking Grain-128 with dynamic cube attacks [C]//Fast Software Encryption, Springer Berlin Heidelberg, 2011: 167 - 187.

[9] Abdul-Latip S F, Reyhanitabar M R, Susilo W, et al. Extended cubes: enhancing the cube attack by extracting low-degree non-linear equations[C]//Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, New York: Association for Computing Machinery, 2011: 296 - 305.

[10] Zhao X J, Wang T, Guo S Z. Improved side channel cube attacks on PRESENT[EB/OL]. (2011 - 04 - 10) [2013 - 06 - 09]. http://eprint.iacr.org/2011/165.

[11] Dinur I, Shamir A. Side channel cube attacks on block ciphers[EB/OL]. (2009 - 03 - 20) [2013 - 04 - 16]. http://eprint.iacr.org/2009/127.

[12] Abdul-Latip S F, Reyhanitabar M R, Susilo W, et al. Fault analysis of the KATAN family of block ciphers [C]// Information Security Practice and Experience, Springer Berlin Heidelberg, 2012: 319 - 336.

[13] Aumasson J P, Dinur I, Meier W, et al. Cube testers and key recovery attacks on reduced-round MD6 and Trivium [C]// Fast Software Encryption, Springer Berlin Heidelberg, 2009: 1 - 22.

[14] Blum M, Luby M, Rubinfeld R. Self-testing/correcting with applications to numerical problems [J]. Journal of Computer and System Sciences, 1993, 47(3): 549 - 595.

[15] Bogdanov A, Knudsen L R, Leaner G, et al. PRESENT: an ultra-lightweight block cipher [C]//Cryptographic Hardware and Embedded System-CHES 2007, Springer Berlin Heidelberg, 2007: 450 - 466.

[16] Yang L, Wang M Q, Qiao S Y. Side channel cube attack on PRESENT [C]//Cryptology and Network Security, Springer Berlin Heidelberg, 2009: 379 - 391.

[17] Canniere C, Preneel B. Trivium-a stream cipher construction inspired by block cipher design principles [EB/OL]. (2005 - 05 - 30) [2013 - 07 - 01]. http://www.ecrypt.eu.org/stream.