广义生日攻击的改进

李梦东[1,2] 邵玉芳[2] 孙玉情[2] 蔡坤锦[2]

[1](北京电子科技学院 北京 100070)

[2](西安电子科技大学通信工程学院 陕西 西安 710071)

摘 要 广义生日攻击算法是密码分析的一个常用工具。2015 年亚密会,Ivica Nikolic 针对列表数目为 $2<k<n$的情况提出了多碰撞算法,该算法的复杂度优于经典的 k-树算法。针对多碰撞算法中列表数目是 $2<k\neq 2^a<n$情况,我们提出新的 newtree 算法,主要的改进有两点:首先在处理消极列表的最后加入 k-树算法的处理;其次,在处理积极列表的最后加入部分多碰撞算法。newtree 算法的时间渐进复杂度为 $\widetilde{O}(k2^{\frac{n-(2+kP)\log p}{\log k_A+3}})$,相比于多碰撞算法的渐进复杂度 $\widetilde{O}(k2^{\frac{n-kp\log p}{\log k_A+1}})$而言有所提高。

关键词 广义生日攻击 k-树算法 多碰撞算法 new tree 算法

中图分类号 TP309 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2018.06.054

GENERALIZED BIRTHDAY ATTACK IMPROVEMENT

Li Mengdong[1,2] Shao Yufang[2] Sun Yuqing[2] Cai Kunjin[2]

[1](*Beijing Electronic Science and Technology Institute*, *Beijing 100070*, *China*)

[2](*Institute of Communication Engineering*, *Xidian University*, *Xi'an 710071*, *Shaanxi*, *China*)

Abstract The generalized birthday attack algorithm is a common tool for cryptanalysis. In 2015 ASIACRYPT, Avici Ivica Nikolic proposed a multi-collision algorithm for the case where the number of lists was $2<k<n$. The complexity of the algorithm was better than that of the classical k-tree algorithm. For the multi-collision algorithm, the number of lists was $2<k\neq 2^a<n$. We proposed a new newtree algorithm. The main improvements were two points: First, the k-tree algorithm was added at the end of the negative list. Second, a partial multi-collision algorithm was added at the end of the active list. The time complexity of the newtree algorithm was $\widetilde{O}(k2^{\frac{n-(2+kP)\log p}{\log k_A+3}})$ and it was improved compared to the progressive complexity of the multi-collision algorithm$\widetilde{O}(k2^{\frac{n-kp\log p}{\log k_A+1}})$.

Keywords Generalized birthday attack K-tree algorithm Multi-collision algorithm Newtree algorithm

0 引 言

在密码分析技术中,碰撞搜索已经得到了广泛的研究,其中较为典型的是广义生日攻击,它是众所周知的生日攻击[1]的推广。广义生日攻击也被称为 k-列表问题,定义如下:已知 k 个列表和一个目标向量的异或结果为目标向量。Wagner 首次研究广义生日攻击,对任何范围的 k 值,他提出一个算法(k-树算法[2])解决 k-列表问题,2015 年亚密会,Ivica Nikolic 提出多碰撞算法[3]提高了算法的复杂度,该算法在处理消极列表方面与 k-树算法不同。

目前,基于格的密码算法[4-5]以及基于纠错码的密码算法[6-7]能够抵抗量子计算,它们是密码学研究的热点。而广义生日攻击或其改进算法[8-9]可以用来评价这些密码算法的安全性。2004 年,Croon 和 Joux 把广义生日攻击运用到译码问题上[10],使基于译码困难问题的加密方案受到了挑战,2007 年,Mathieu Fini-

收稿日期:2017-11-10。北京市支持中央高校共建项目——青年英才计划,中央高校基本科研业务费专项资金资助课题(2017CL06)。**李梦东**,教授,主研领域:密码算法及其应用。**邵玉芳**,硕士生。**孙玉情**,硕士生。**蔡坤锦**,硕士生。

asz 提出伴随式译码问题的变形[11](规则字译码问题),即已知 w 大小为 $r\times\frac{n}{w}$ 的二进制矩阵 $\boldsymbol{H}_i$ 和一个伴随式 $\mathcal{S}\in\{0,1\}^s$,在 W 个子矩阵 $\boldsymbol{H}_i$ 中各寻找一个向量,使得这些向量异或和与伴随式 s 相等。如果把每个二进制矩阵 $\boldsymbol{H}_i$ 视为列表,列表中的向量看作矩阵 $\boldsymbol{H}_i$ 中的向量,那么使用 Wagner 提出的 k-树算法可以求解规则字译码问题[2],但是复杂度较高。虽然对伴随式译码问题最好的攻击方法是信息集攻击,但是广义生日攻击也是有意义的。目前多碰撞算法是广义生日攻击的最好的攻击方式。

本文在分析已有广义生日攻击基础上,提出新的算法,newtree 算法的时间渐进复杂度为 $\widetilde{O}(k2^{\frac{n-(2+kp)\log p}{\log k_A+3}})$,相比较多碰撞算法的渐进时间复杂度 $\widetilde{O}(k2^{\frac{n-kp\log p}{\log k_A+1}})$有所提高。但是,newtree 算法只适用于 $2<k\neq 2^a<n$ 的情况,对于 $2<k=2^a<n$ 的情况还有待分析。

1　已有算法介绍与分析

1.1　k-树算法

Problem1 已知 k 个随机列表和一个随机向量 $x\in\{0,1\}^n$,试寻找 $x_1\in\mathscr{L}_1,\cdots,x_k\in\mathscr{L}_k$ 满足 $x_1\oplus x_2\oplus\cdots\oplus x_k=x$,一般情况下 $x=0$。

当 $|\mathscr{L}_1|\times|\mathscr{L}_2|\times\cdots\times|\mathscr{L}_k|\geqslant 2^n$ 时,虽然方案 $x_1\cdots x_k$ 存在的概率较高;但是,如何有效地寻找依然是困难的。对于不同范围的 k 使用不同的方法,例如,当 $k=2$ 时,使用标准的生日攻击算法,其渐进复杂度为 $\widetilde{O}(2^{n/2})$;当 $k\geqslant n$ 时,使用 Bellare 和 Micciancio 在 1997 年提出的算法[6],该算法通过高斯消元法完成,其渐进复杂度是 $\widetilde{O}(n^3+kn)$。对于 $2<k<n$ 时,却只能使用一个普通的算法,该算法首先创建两个较大的列表 $\overline{\mathscr{L}_1}$ 和 $\overline{\mathscr{L}_2}$,其中 $\overline{\mathscr{L}_1}=\{X|X=x_1\oplus\cdots\oplus x_{k/2},x_i\in\mathscr{L}_i\}$,$\overline{\mathscr{L}_2}=\{X|X=x_{k/2+1}\oplus\cdots\oplus x_k,x_i\in\overline{\mathscr{L}_i}\}$,最后在 $\overline{\mathscr{L}_1}$ 和 $\overline{\mathscr{L}_2}$ 中寻找碰撞。

2002 年,Wagner 基于假设列表中的元素较多,提出了 k-树算法来解决广义生日问题。k-树算法的优势体现在,k-树的每一层都使用同样的碰撞查找算法。定义列表 $S\Delta T=\{x|x=x_1\oplus x_2\in S\times T,x_1\in S,x_2\in T\}$,$low_l(x)$ 表示元素 x 的末端有 l 个连零。另外,$S\Delta_l T$ 集合包含集合 $S\times T$ 中所有 $low_l(x_1\oplus x_2)=0$ 的元素。假设 $\overline{\mathscr{L}_1},\overline{\mathscr{L}_2},\overline{\mathscr{L}_3},\overline{\mathscr{L}_4}$ 是四个列表,每个列表包含有 2^l 个元素,首先创建列表 $\overline{\mathscr{L}_{12}}=\overline{\mathscr{L}_1}\Delta_l\overline{\mathscr{L}_2}=x_1\oplus x_2$,其中 $x_1\in\overline{\mathscr{L}_1},x_2\in\overline{\mathscr{L}_2}$,满足 $low_l(x_1\oplus x_2)=0$;类似的,创建列表 $\overline{\mathscr{L}_{34}}=\overline{\mathscr{L}_3}\Delta\overline{\mathscr{L}_4}=x_3\oplus x_4$ 其中 $x_3\in\overline{\mathscr{L}_3},x_4\in\overline{\mathscr{L}_4}$,满足 $low_l(x_3\oplus x_4)=0$;此时,在列表 $\overline{\mathscr{L}_{12}}$ 和 $\overline{\mathscr{L}_{34}}$ 中每个元素的末端 l 比特皆为零。在列表 $\overline{\mathscr{L}_{12}}$ 和 $\overline{\mathscr{L}_{34}}$ 之间寻找碰撞,即满足剩余的 $n-l$ 比特相同,很显然 $x_1\oplus x_2\oplus x_3\oplus x_4=0$。为了获得匹配方案,$l$ 的值取 $n/3$,那么列表 $\overline{\mathscr{L}_{12}}$、$\overline{\mathscr{L}_{34}}$ 中元素的个数各为 $2^{n/3}\cdot 2^{n/3}/2^{n/3}=2^{n/3}$,所以在第二层中,两个列表将会产生 $2^{n/3}\cdot 2^{n/3}=2^{2n/3}$ 种可能的值,完成剩余的 $n-n/3=2n/3$ 比特匹配。l 的取值是为了保持两层之间复杂度一致。

基于上述的思想,Wagner 解决了任何 k-列表问题。当 k 是 2 的幂次,一般情况下,$\log_2^k$ 为树的层数,每一层(除最后一层)列表两两组合产生 l 比特连零串;在最后一层的两个列表将完成剩余 $2l$ 比特的匹配。此时,$l\cdot\log k+l=n$ 得到 $l=\frac{n}{\log k+1}$。例如,在 8-列表问题中,按照上述的规则分为 $3=\log_2^8$ 层,$l=n/4$。在第一层中,通过合并列表 $\mathscr{L}_1\cdots\mathscr{L}_8$ 建立列表 $\overline{\mathscr{L}_{12}}$,$\overline{\mathscr{L}_{34}}$,$\overline{\mathscr{L}_{56}}$,$\overline{\mathscr{L}_{78}}$,每个列表中有 $2^l=2^{n/4}$ 个元素且末端的 $n/4$ 比特串都为零;在第二层中,在次末端产生 $n/4$ 比特的连零串,建立列表 $\overline{\mathscr{L}_{1234}}$ 和 $\overline{\mathscr{L}_{5678}}$;在第三层中,完成剩余的 $n/2$ 的比特匹配。所以整个算法运行一次的复杂度为 $\widetilde{O}(8\cdot 2^{\frac{n}{\log k+1}})$。当 k 不是 2 的幂次,Wagner 把前 $2^{\lfloor\log_2^k\rfloor}$ 个列表称为积极列表,余下的列表称为消极列表。然后需要在各个消极列表中随机选取一个向量逐比特异或求和,并把结果置于列表 $\overline{\mathscr{L}}$ 中的每个元素。例如,在 $\overline{\mathscr{L}_1}\cdots\overline{\mathscr{L}_6}$ 的 6-列表问题中,需要从列表 $\mathscr{L}_5$ 和 $\mathscr{L}_6$ 中随机选取一个元素 $v_5\in\mathscr{L}_5$ 和 $v_6\in\mathscr{L}_6$,计算 $v_5\oplus v_6$,然后与列表 $\overline{\mathscr{L}_{56}}$ 中的每一个元素异或,最后利用 4-树算法解决 6-列表问题。Wagner 指出当 k 不是 2 的幂次时,该算法的复杂度和 $2^{\lfloor\log_2^k\rfloor}$ - 树算法的复杂度一样。

所以,对于任意值 k,k-树算法运行一次的空间复杂度和时间复杂度都为 $\widetilde{O}(k\cdot 2^{\frac{n}{\lfloor\log k\rfloor+1}})$,但是这不意味着该算法的运行一定会产生碰撞。一般来说,k-树算法运行次数是由树结构第 $\log k-1$ 层列表中剩余未匹配向量的长度和每个列表中元素的数量所决定的。在 k-树算法的结构中,除最后一层所有的列表大小为 2^l,匹配的比特数为 l。那么每次运行 k-树算法攻击规则字译码问题或者 k-列表问题成功的概率为:

$$P_{\text{k-tree}}=1-\left(\frac{2^{n-(\log k-1)l}-1}{2^{n-(\log k-1)l}}\right)^{2^{2l}}$$

从而得到至少运行 $E=P_{\text{k-tree}}^{-1}$ 次才能找到 $x_1\in\mathscr{L}_1\cdots x_k\in\mathscr{L}_k$ 满足 $x_1\oplus x_2\oplus\cdots\oplus x_k=0$,当 $n-l\log k$ 的值较大时,E 的值大约为 $2^{n-l\log k}$。此时,通过 k-树算法攻

击成功的时间复杂度为 $\tilde{O}(k\cdot 2^{\frac{n}{\lfloor \log k\rfloor+1}}\cdot 2^{n-l\log k})$。

1.2 多碰撞算法

2015 年在亚密会上,Ivica Nikolic 对 Wagner 的 k-树算法进行了改进,此次改进主要是基于部分多碰撞的思想,使用部分多碰撞算法处理消极列表。换句话说,不再是简单地从消极列表中取出元素,而是从消极列表中找到固定比特位上具有相同值的向量集合。然后,迫使积极列表中的元素在对应的比特位上具有相同的比特。最后,把积极列表和消极列表合并满足剩余的比特位产生碰撞。Ivica Nikolic 把 $2<k<n$ 的值分为 $k=3$、$k>3$两种情况。接下来详细介绍改进后的 3-树算法。

定义 1 集合 $S=\{x_1,\cdots,x_p\}$,其中 $x_1,\cdots,x_p$ 是 n 比特的向量。如果 $low_s(x_1)=low_s(x_2)=\cdots=low_s(x_p)$,那么就形成向量末端 s 比特的部分多碰撞。

从定义可知,集合 S 中 p 个向量的最后 s 比特是相同的。对于 3-列表问题,每个列表的大小为 2^l,Ivica Nikolic 把列表$\overline{\mathscr{L}_1}$,$\overline{\mathscr{L}_2}$作为积极列表,列表$\overline{\mathscr{L}_3}$作为消极列表,并在消极列表$\overline{\mathscr{L}_3}$中完成部分多碰撞,并把碰撞的向量置于集合$\overline{\mathscr{L}_{3*}}=\{x_{31},\cdots,x_{3p}\mid low_l(x_{31})=low_l(x_{32})=\cdots=low_l(x_{3p})\}$,如图 1 所示。不失一般性,假设集合$\overline{\mathscr{L}_{3*}}$中的向量后 l 比特为零(如果不为零时,积极列表$\overline{\mathscr{L}_1}$,$\overline{\mathscr{L}_2}$的匹配值与部分多碰撞的值相同)。在第二层,创建列表$\overline{\mathscr{L}_{12}}=\{x_1\oplus x_2\mid x_1\in L_1,x_2\in L_2,low_l(x_1+x_2)=0\}$,列表$\overline{\mathscr{L}_{12}}$的大小约 2^l。在第三层,使用列表$\overline{\mathscr{L}_{12}}$,$\overline{\mathscr{L}_{3*}}$之间完成余下 $n-l$ 比特的匹配,只要 $p\mid\overline{\mathscr{L}_{12}}\mid\geqslant 2^{n-l}$,解决方案就会以较高的概率存在。

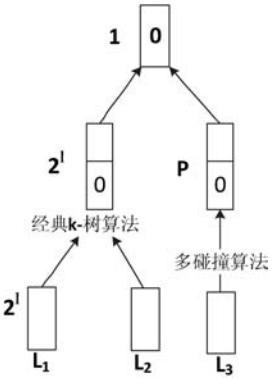

图 1 改进 3-树算法

该算法的复杂度主要取决于两个部分,一个是创建列表$\overline{\mathscr{L}_{12}}$的复杂度,另一个是产生部分多碰撞列表$\overline{\mathscr{L}_{3*}}$的复杂度。每个积极列表中的复杂度为 $\tilde{O}(2^l)$,大小为 2^l 的消极列表中产生部分多碰撞的时间复杂度为 $\tilde{O}(2^l)$。因此该算法解决 3-列表问题的时间复杂度为 $\tilde{O}(3\times 2^l)$。对于 l,把不等式 $p2^{2l-n}\geqslant 1$ 变换为 $p2^{2l-n}=1$,得到 $l=\frac{n}{2}-\frac{1}{2}\log(p)$;该算法的时间复杂度为 $\tilde{O}(3\times 2^{n/2}/\sqrt{p})$,其中 p 是多碰撞集合的大小,当 p 的值变大时,复杂度随之减小。当使用经典的 Wagner 算法时,只从列表$\overline{\mathscr{L}_3}$中获得一个元素,因此在 $p=1$ 时,Wagner 算法的时间复杂度为 $\tilde{O}(k\times 2^{n/2})$,可知多碰撞算法是 Wagner 算法的推广。

Kazuhiro Suzuki 指出[12]一个集合具有高概率产生多碰撞元素,可以得到:

$$(p!)^{1/p}2^{\frac{p-1}{p}l}=2^l$$

当然也可以通过一种简单的方式,寻找 p 的最大值。Kazuhiro Suzuki 提出球箱问题,即 m 个球随机扔进 m 个箱子里,试寻找那个箱子中的球最多。这个问题的解决方案是公知的,并且最大期望渐近值为:

$$\Theta\left(\frac{\ln m}{\ln\ln m}\right)$$

而部分多碰撞列表$\overline{\mathscr{L}_3}$是球箱问题的一个实例,2^l 表示为球和箱的个数,p 表示箱子中球数的最大值。因此,$p(l)$ 的渐进值可以认为是 $\Theta\left(\frac{\ln m}{\ln\ln m}\right)=\Theta\left(\frac{l}{\ln l}\right)$。最后,当 $l\approx\frac{n}{2}$时,$p=\sqrt{\frac{n/2}{\ln n/2}}$,该算法运行一次的时间复杂度为 $\tilde{O}=\left(k2^{n/2}/\sqrt{\frac{n/2}{\ln n/2}}\right)$,每次运行该算法成功的概率为 $P_{\text{multicollisions}}=1-\left(\frac{2^{n-l(\log k_A)}-1}{2^{n-l(\log k_A)}}\right)^{p\cdot 2^l}$,那么当 $k=3$时,改进的树算法(多碰撞算法)需要运行 $E=P^{-1}_{\text{multicollisions}}$次方能成功。整个 3-多碰撞算法的时间复杂度为 $\tilde{O}=\left(E\cdot 2^{n/2}/\sqrt{\frac{n/2}{\ln n/2}}\right)$。

Ivica Nikolic 的多碰撞算法在 k 值较大(不是 2 的幂)的情况也是有效的。具体的操作如图 2 所示,把 k 个列表分为 k_A 个积极列表和 k_P 个消极列表。

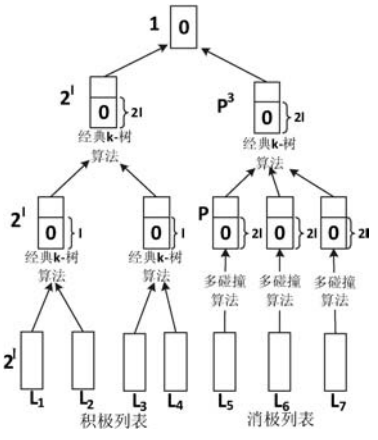

图 2 改进的 k-树算法($k>3$)

例如,$k=7$ 时意味着 $k_A=4$ 个积极列表,$k_P=4$ 个

消极列表。在 k_P 个消极列表$\overline{\mathscr{L}_{k_A+1}}\cdots\overline{\mathscr{L}_k}$中分别独立的产生部分多碰撞列表$\overline{\mathscr{L}_{k_A+1*}}\cdots\overline{\mathscr{L}_{k*}}$,其中任意列表中的元素末端 $\lambda = l\log k_A$ 长的比特值相同。假设 $v_1,\cdots,v_{k_p}$分别是多碰撞列表的碰撞值,然后计算 $v = v_1 \oplus\cdots\oplus v_{k_p}$。显然,$\overline{\mathscr{L}_P} = \overline{\mathscr{L}_{k_A+1}}\Delta\cdots\Delta\overline{\mathscr{L}_k}$列表的大小为$|\overline{\mathscr{L}_P}| = p^{k_p}$,列表$\overline{\mathscr{L}_P}$的所有元素的末端 λ 比特值皆为 v,简单起见,假设 $v=0$。接下来在 k_A 个积极列表中运用经典的树算法,每一层完成长度为 l 的零比特值匹配,每一层中的列表大小为 2^l,在第 $\log k_A$ 层中列表每个元素的后 $l\log k_A$ 比特值与 v 相同。最后在列表$\overline{\mathscr{L}_P}$和列表$\overline{\mathscr{L}_{1\cdots k_A}}$中完成剩余的 $n-\lambda$ 比特值匹配。当 $2^l p^{k_p} \geqslant 2^{n-\lambda}$ 时,该算法可以以较高的概率产生解决方案。

接下来讨论该算法的复杂度,首先是从消极列表 $\overline{\mathscr{L}_i}, i = k_A+1\cdots k$ 中产生部分多碰撞需要 $k_p 2^l$ 次查找操作,另一个方面是在积极列表中运用经典的 k-树算法需要 $k_A 2^{\frac{\lambda}{\log k_A}} = k_A 2^l$ 次查找操作。整个算法运行一次的复杂度为 $k_p 2^l + k_A 2^l = (k_p + k_A)2^l = k2^l$。试寻找 l 的最小值,把不等式 $2^l p^{k_p} \geqslant 2^{n-\lambda}$ 变为 $k_P\log p + l = n - l\log k_A$,得到:

$$l = \frac{n}{\log k_A + 1} - \frac{k_P \log p}{\log k_A + 1}$$

因此,相比较经典的 k-树算法,改进的 k-树算法复杂度之比为:

$$\frac{k2^{\frac{n}{\log k_A+1}}}{k2^l} = 2^{\frac{k_P\log p}{\log k_A+1}} = (2^{\log p})^{\frac{k_P}{\log k_A+1}} = p^{\frac{k_P}{\log k_A+1}}$$

p 的近似值可以通过如下方法得到。因为多碰撞列表的大小变化(如 $k=7$ 时,第三层多碰撞列表空间的大小为 p^3),使得 p 的值不能通过球箱问题解决了。因此,只能通过定理估测多碰撞数量 p 值,用一个简单的变换可以得到$\frac{l}{p} = \log\frac{p}{\mathrm{e}}$,等式的近似解的形式为 $p = \frac{l}{\log\frac{l}{\mathrm{e}}}$。因此,改进的 k-树算法提高因子估测为:$\left(\frac{l}{\log l/\mathrm{e}}\right)^{\frac{k_P}{\log k_A+1}}$,改进的算法每次运行成功的概率为 $P_{\text{multicollisions}} = 1 - \left(\frac{2^{n-l(\log k_A)}-1}{2^{n-l(\log k_A)}}\right)^{p\cdot 2^l}$,需要运行 $E = P^{-1}_{\text{multicollisions}}$ 次才能找到一个解决方案。故在 $k>3$ 时,改进的 k-树算法整体时间复杂度为

$$(E\cdot k\times 2^{\frac{n}{\log k_A+1}-\frac{k_P\log p}{\log k_A+1}})$$

改进的 k-树算法和以前的 k-树算法区别在于消极列表完成匹配的过程。当 k 不是 2 的幂次时,以前的算法是在消极列表中随机选择一个元素与目标向量逐比特异或的结果作为积极列表完成匹配的目标向量。Ivica Nikolic 提出新的方案,在消极列表中使用部分多碰撞算法,即在消极列表中选择满足一定条件的 p 个向量,这本质是 Wanger 算法的一种推广。通过 Ivica Nikolic 的分析,改进算法的运行时间复杂度提高了 $p^{\frac{k_P}{\log k_A+1}}$。

2　newtree 算法介绍

目前对广义生日攻击的算法改进集中在 $2 < k < n$ 的范围。通过分析多碰撞算法的优越性和经典的 k-树算法,我们提出 newtree 算法,该算法只要针对列表数目是 $k \neq 2^a$ 的情况($3 \leqslant k < n$)。该算法继承过去算法的优势,主要体现在对积极列表和消极列表的处理。主要区别有两点,第一,在消极列表经过部分多碰撞算法处理的最后一步,替换为经典的 k-树算法,统一处理得到最后的列表 $\overline{\mathscr{L}_p}$;第二,在处理积极列表的最后一步时,加入部分多碰撞算法处理后,再用原来方式处理。上述两点是针对消极列表和积极列表处理的改进。最后,列表 $\overline{\mathscr{L}_*}$ 和 $\overline{\mathscr{L}_p}$ 经过 k-树算法完成剩余比特的匹配。此改进可以在 Ivica Nikolic 算法的基础上进一步提高;但 newtree 算法对列表的个数为 $k = 2^a$ 是无效的。接下来叙述 newtree 算法。

在 $2 < k < n$ 时,当 $k \neq 2^a$ 时,把 k 个列表分为前 k_A 个积极列表和后 k_P 个消极列表;例如,当 $k = 11$ 时,$k_A = 8$, $k_p = 3$ 。在 newtree 算法中先处理消极列表,消极列表先经过部分多碰撞算法,再经过 Wanger 算法。具体如下:每个列表的大小为 2^l ,对于消极列表 $\overline{\mathscr{L}_{k_A+1}}\cdots\overline{\mathscr{L}_k}$ 分别使用部分多碰撞算法产生列表 $\overline{\mathscr{L}_{k_A+1*}}\cdots\overline{\mathscr{L}_{k*}}$,匹配长度为 $l\log k_A$,列表的表达式为:

$$\overline{\mathscr{L}_{i*}} = \{x_{i1},\cdots x_{ip} \mid low_{l\log k_A}(x_{i1}) = low_{l\log k_A}(x_{i2}) = \cdots = low_{l\log k_A}(x_{ip})\}$$

其中 $i \in \{k_A+1, k\}$, $|\overline{\mathscr{L}_{i*}}| = p$ 。

k_p 个消极列表通过多碰撞产生 k_p 个 $l\log k_A$ 长的碰撞值,即 $v_1\cdots v_{k_p}$,每个值分别对应消极列表 $\overline{\mathscr{L}_{i*}}$ 的碰撞值。上述步骤和 Ivica Nikolic 算法完全一样,接下来通过 Wanger 算法处理列表 $\overline{\mathscr{L}_{i*}}$,再产生 l 长的随机碰撞值 v_P ,那么得到新列表 $\overline{\mathscr{L}_P}$ 为 $\{x_1\cdots x_{\frac{p^{k_p}}{2^l}} \mid low_{l\log k_p}(x_i) = v_1\oplus\cdots\oplus v_{k_p}, low_{[l\log k_p+1, l\log k_p+l]}(x_{ip}) = v_P\}$,其中 $i \in \left\{1, \frac{p^{k_p}}{2^l}\right\}$。此时对消极列表的处理结束,我们获得 $l(1+\log k_A)$ 长的匹配值 $(v_P, v = v_1\oplus\cdots\oplus v_{k_p})$。如 $k_p = 3$,消极列表 $\overline{\mathscr{L}_9}$、$\overline{\mathscr{L}_{10}}$、$\overline{\mathscr{L}_{11}}$ 分别经过部分多碰撞算法得到列表:

$$\overline{\mathscr{L}_{9*}} = \{x_{91}\cdots x_{9p} \mid low_{l\log 8}(x_{91}) = \cdots = low_{l\log 8}(x_{9p}) = v_1\}$$

$\overline{\mathscr{L}_{10*}} = \{x_{101}\cdots x_{10p} \mid low_{l\log 8}(x_{101}) = \cdots = low_{l\log 8}(x_{10p}) = v_2\}$

$\overline{\mathscr{L}_{11*}} = \{x_{111}\cdots x_{11p} \mid low_{l\log 8}(x_{111}) = \cdots = low_{l\log 8}(x_{11p}) = v_3\}$

上述的三个列表分别经过 Wanger 算法产生碰撞值 v_p,得到列表:

$\overline{\mathscr{L}_p} = \{x_1\cdots x_{\frac{p^3}{2^l}} \mid low_{l\log 8}(x_{111}) = \cdots = low_{l\log 8}(x_{11p}) = v = v_1 \oplus v_2 \oplus v_3, low_{[3l+1,4l]}(x_{ip}) = v_P\}$

接下来处理积极列表,k_A 个积极列表 $\overline{\mathscr{L}_1}\cdots\overline{\mathscr{L}_{k_A}}$ 先使用 Wanger 算法,再由多碰撞算法处理。假设 $k_A = 8$,列表 $\overline{\mathscr{L}_1}, \overline{\mathscr{L}_2}, \overline{\mathscr{L}_3}, \overline{\mathscr{L}_4}, \overline{\mathscr{L}_5}, \overline{\mathscr{L}_6}, \overline{\mathscr{L}_7}, \overline{\mathscr{L}_8}$ 作为 Wanger 算法的第一层,其中每个列表各有 2^l 向量。首先建立第一层的列表 $\overline{\mathscr{L}_{12}} = \overline{\mathscr{L}_1}\Delta_l\ \overline{\mathscr{L}_2} = x_1 \oplus x_2$,其中 $x_1 \in \overline{\mathscr{L}_1}$, $x_2 \in \overline{\mathscr{L}_2}$, $low_l(x_1 \oplus x_2) = low_l(v)$;列表 $\overline{\mathscr{L}_{34}} = \overline{\mathscr{L}_3}\Delta_l\ \overline{\mathscr{L}_4} = x_3 \oplus x_4$,其中 $x_3 \in \overline{\mathscr{L}_3}$, $x_4 \in \overline{\mathscr{L}_4}$, $low_l(x_3 \oplus x_4) = low_l(v)$;同理,产生列表 $\overline{\mathscr{L}_{56}}$ 和 $\overline{\mathscr{L}_{78}}$。在第一层列表 $\overline{\mathscr{L}_{12}}$, $\overline{\mathscr{L}_{34}}$, $\overline{\mathscr{L}_{56}}$, $\overline{\mathscr{L}_{78}}$ 中,每个列表中元素个数为 2^l,这些元素的末端 l 比特值相同,然后这四个列表通过 Wanger 算法得到第二层列表 $\overline{\mathscr{L}_{1234}} = \overline{\mathscr{L}_{12}}\Delta_l\ \overline{\mathscr{L}_{34}}$,同理得到 $\overline{\mathscr{L}_{5678}} = \overline{\mathscr{L}_{56}}\Delta_l\ \overline{\mathscr{L}_{78}}$。此时,在第二层的每个列表中都包含元素 2^l,而且每个元素的次末端有 l 比特值相同。上述的算法和 Ivica Nikolic 算法完全一样,但是对于最后的两个列表 $\overline{\mathscr{L}_{1234}}$ 和 $\overline{\mathscr{L}_{5678}}$,我们使用部分多碰撞算法产生列表 $\overline{\mathscr{L}_{1234*}}$ 和 $\overline{\mathscr{L}_{5678*}}$,使得这两个列表中任意元素异或值为 $low_{[2l+1,3l]}(v)$,然后通过 Wanger 算法得到最后的列表 $\overline{\mathscr{L}_*}$,表达式如下:

$\overline{\mathscr{L}_*} = \{x_1\cdots x_{\frac{p^2}{2^l}} \mid low_{l+l\log k_A}(x_1) = \cdots = low_{l+l\log k_A}(x_{\frac{p^2}{2^l}}) = v_P \parallel v\}$

式中:$\parallel$ 表示比特的链接。把消极列表和积极列表经过上述处理后,我们得到列表 $\overline{\mathscr{L}_*}$ 和 $\overline{\mathscr{L}_p}$。最后,把列表经过 Wanger 算法完成最后的 n-l-$l\log k_A$ 比特匹配,图 3 是当 $k = 7$ 时的 newtree 算法示意,一般情况下 $v = 0, v_p = 0$。

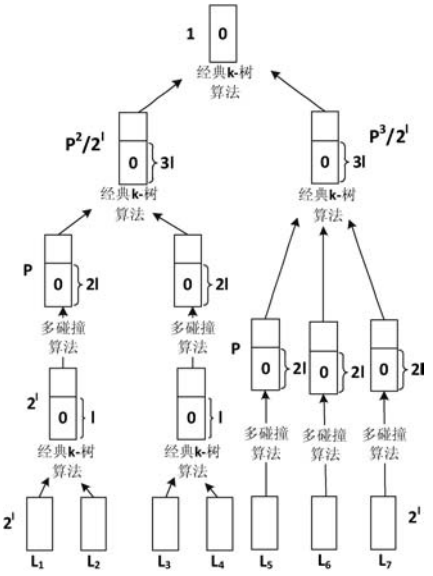

图 3 newtree 算法($k = 7$)

3 复杂度分析

首先对 k_A 个积极列表的查找操作处理需要 $k_A 2^l$ 次,随后的查找操作将随 k_A 的值减少而变小,渐进复杂度为 $\widetilde{O}(k_A 2^l)$;对于消极列表的部分多碰撞处理需要 $k_p 2^l$ 次查找操作,渐进复杂度为 $\widetilde{O}(k_p 2^l)$;整个 newtree 算法运行的时间复杂度为 $\widetilde{O}(k2^l)$,试寻找 l 的最小值,该算法如果以较高的概率产生解决方案,必须满足 $\left(\frac{p^2}{2^l}\right)\left(\frac{p^{k_P}}{2^l}\right) \geqslant 2^{n-l-l\log k_A}$,经过变换不等式得到:

$$l \geqslant \frac{n - (2 + k_P)\log p}{\log k_A + 3}$$

得到 newtree 算法的运行时间渐进复杂度为 $\widetilde{O}(k2^{\frac{n-(2+k_p)\log p}{\log k_A+3}})$,而多碰撞算法渐进时间复杂度为 $\widetilde{O}(k \times 2^{\frac{n}{\log k_A+1}} - \frac{k_p \log p}{\log k_A + 1})$,相比较而言 newtree 算法的复杂度较为提高。

newtree 算法每次运行的可以找到方案的概率为:

$$P_{\text{new}} = 1 - \left(\frac{2^{n-l(1+\log k_A)} - 1}{2^{n-l(1+\log k_A)}}\right)^{p^{(2+k_p)}/2^{2l}}$$

所以,newtree 至少需要运行 $E = P_{\text{new}}^{-1}$ 次才能找到一个解决方案。在 $3 \leqslant k \neq 2^a \leqslant n$ 的情况下,整个 newtree算法能够找到解决 Problem1 的渐进复杂度为 $\widetilde{O}(P_{\text{new}}^{-1} k 2^{\frac{n-(2+k_p)\log p}{\log k_A+3}})$。显然,Ivica Nikolic 多碰撞算法成功的概率与 newtree 算法相比,在 $n - l(1 + \log k_A)$ 较大的情况下,P_{new} 与 $P_{\text{multicollisions}}$ 的大小比较可以归结为指数 $p^{(2+k_p)}/2^{2l}$ 与 $p2^l$ 的比较,当 $\frac{p^{(2+k_p)}}{2^{2l}} \geqslant p2^l$ 时,即 $(1 + k_p)\ln p > 2.079l$ 时,$P_{\text{new}}^{-1} \geqslant P_{\text{multicollisions}}^{-1}$,得到 newtree 算法的复杂度 $\widetilde{O}(P_{\text{new}}^{-1} k 2^{\frac{n-(2+k_p)\log p}{\log k_A+3}})$ 优于多碰撞算法的渐进复杂度 $\widetilde{O}(P_{\text{multicollisions}}^{-1} \cdot k \times 2^{\frac{n}{\log k_A+1} - \frac{k_p \log p}{\log k_A+1}})$。

4 结 语

基于纠错码问题(规则字译码问题[11])设计的加密体制是目前抵抗量子计算机攻击的主要方案之一。本文提出的 newtree 算法,这种攻击算法可以对规则字译码问题进行攻击,其复杂度优于多碰撞算法的复杂度。

本文中我们给出了最近提出的多碰撞算法的一个改进算法 newtree,主要思路体现在对积极列表和消极列表的处理方式上,从结果的公式比较可以得到渐进

复杂度有所提高,但是 newtree 算法的局限性是它只适用于 $2 < k \neq 2^a < n$ 的情况,以后的工作可以研究 $2 < k = 2^a < n$的情况。

参 考 文 献

[1] Mosteller F. Understanding the Birthday Problem[J]. Mathematics Teacher, 1962, 55(5):322 - 325.

[2] Wagner D. A Generalized Birthday Problem[M]//Advances in Cryptology—CRYPTO 2002. Springer Berlin Heidelberg, 2002:288 - 304.

[3] Nikolić I, Yu S. Refinements of the k-tree Algorithm for the Generalized Birthday Problem[C]//Proceedings, Part II, of the 21st International Conference on Advances in Cryptology-ASIACRYPT 2015, 9453:683 - 703.

[4] Hoffstein J, Pipher J, Silverman J H. NTRU: A ring-based public key cryptosystem[M]//Algorithmic Number Theory. Springer Berlin Heidelberg, 1998:267 - 288.

[5] Lyubashevsky V, Micciancio D, Peikert C, et al. SWIFFT: A Modest Proposal for FFT Hashing[M]//Fast Software Encryption. Springer Berlin Heidelberg, 2008:54 - 72.

[6] Finiasz M, Gaborit P, Manuel S, et al. SHA-3 proposal: FSB. Submission to the SHA-3 NIST Competition (2008)[OL]. http://www-rocq. inria. fr/secret/CBCrypto/index. php? pg = fsb.

[7] Meziani M, Özgür Dagdelen, Cayrel P L, et al. S-FSB: An Improved Variant of the FSB Hash Family[M]//Information Security and Assurance. Springer Berlin Heidelberg, 2011: 132 - 145.

[8] Overbeck R. A multiple birthday attack on NTRU[J]. IEEE Comput Graph Appl, 2011, 31(6):45 - 55.

[9] Bernstein D J, Lange T, Niederhagen R, et al. Implementing Wagner's generalized birthday attack against the SHA-3 round-1 candidate FSB[C]. Cryptosith Org, 2010, 2009.

[10] Coron J S, Joux A. Cryptanalysis of a Provably Secure Cryptographic Hash Function[C]//G Brassard Advances in Cryptology-Crypto Lncs, 2004:416 - 427.

[11] Finiasz M, Gaborit P, Sendrier N. Improved Fast Syndrome Based Cryptographic Hash Functions[C]//Proceedings of ECRYPT Hash Workshop 2007.

[12] Suzuki K, Tonien D, Kurosawa K, et al. Birthday Paradox for Multi-collisions[C]//International Conference on Information Security and Cryptology. Springer, Berlin, Heidelberg, 2006:29 - 40.

(上接第 247 页)

[8] Ho I S F, Deris S, Zaiton M H S. A Combination of PSO and Local Search in University Course Timetabling Problem[C]//International Conference on Computer Engineering and Technology. IEEE Computer Society, 2009: 492 - 495.

[9] Li H. Narrowing Support Searching Range in Maintaining Arc Consistency for Solving Constraint Satisfaction Problems[J]. IEEE Access, 2017, 5(99): 5798 - 5803.

[10] Di M M, Forti M, Nistri P, et al. Nonsmooth Neural Network for Convex Time-Dependent Constraint Satisfaction Problems[J]. IEEE Transactions on Neural Networks & Learning Systems, 2016, 27(2): 295 - 307.

[11] Luo T, Dolan M, Davidson E, et al. Assessment of a new constraint satisfaction problem based active demand control approach to address distribution network constraints[J]. ET Generation, Transmission & Distribution, 2015, 9(15): 2363 - 2373.

[12] Jiang X, Cui P, Xu R, et al. An action guided constraint satisfaction technique for planning problem[C]//International Conference on Cognitive Informatics & Cognitive Computing. IEEE, 2017: 167 - 173.

[13] Shen J, Mei D. The Freuder Width in a General Model of Constraint Satisfaction Problem[C]//International Symposium on Computational Intelligence and Design. IEEE, 2017: 303 - 306.

[14] Chen H, Valeriote M, Yoshida Y. Testing Assignments to Constraint Satisfaction Problems[C]//Foundations of Computer Science. IEEE, 2016: 525 - 534.

[15] Rouahi A, Salah K B, Ghydira K. Belief Constraint Satisfaction Problems[C]//Computer Systems and Applications. IEEE, 2016: 1 - 4.

[16] Mallick S, Ghoshal S P, Acharjee P, et al. Optimal Static State Estimation Using hybrid Particle Swarm-Differential Evolution Based Optimization[J]. Energy & Power Engineering, 2017, 5(4): 670 - 676.

[17] Jin H L, Song J Y, Kim D W, et al. Particle Swarm Optimization Algorithm with Intelligent Particle Number Control for Optimal Design of Electric Machines[J]. IEEE Transactions on Industrial Electronics, 2018, 65(2): 1791 - 1798.

[18] 项铁铭, 王建成. 改进的多目标粒子群优化算法[J]. 计算机应用与软件, 2017, 34(9):302 - 305.

[19] Tassopoulos I X, Beligiannis G N. Solving effectively the school timetabling problem using particle swarm optimization[J]. Expert Systems with Applications, 2012, 39(5): 6029 - 6040.

[20] Dechter R, Frost D. Backjump-based backtracking for constraint satisfaction problems[J]. Artificial Intelligence, 2002, 136(2): 147 - 188.

[21] Irene S F H, Deris S, Zaiton M H S. A Study on PSO-Based University Course Timetabling Problem[C]//International Conference on Advanced Computer Control. IEEE Computer Society, 2009: 648 - 651.