第 36 卷 第 2 期 计 算 机 工 程 2010 年 1 月
Vol.36 No.2 Computer Engineering January 2010
·安全技术· 文章编号：1000—3428(2010)02—0150—02 文献标识码：A 中图分类号：TN918.1

模 2^n 加的异或差分概率的快速计算方法

张庆贵
(解放军信息工程大学电子技术学院，郑州 450004)

摘 要：分析模 2^n 加变换的异或差分概率计算算法的计算复杂性，利用以空间换时间的思想，将该算法中的矩阵乘积运算预先计算并予以存储，从而以查表运算替代多个矩阵乘积运算等方法对模 2^n 加变换的异或差分概率计算算法进行改进，改进后算法的计算复杂性小于现有方法计算复杂性的 7.7%。

关键词：模 2^n 加；异或差分概率；快速计算

Fast Computing Method to XOR Differential Probability of Addition Modulo 2^n

ZHANG Qing-gui
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】 This paper analyzes the computational complexity of the algorithm for computing the XOR differential probability of addition modulo 2^n. With the idea of time-memory trade-off, by computing and storing the products of matrices algorithm in advance, and replacing the computing of matrices products with looking up tables, it improves the computing algorithm. The computational complexity of new algorithm is less than 7.7% times of that of the existing method.

【Key words】 additional modulo 2^n; XOR differential probability; fast computing

1 概述

差分攻击方法[1-2]是分析分组密码的重要方法之一。在对分组密码算法进行差分分析时，首先要对圈函数的各个编码环节差分信息泄漏的规律进行分析，然后利用其差分信息泄漏的特点，构造出分组密码算法的一个高概率的差分对应、或者构造出不可能差分对应、或者找出其差分分布的可预测性。因此，对于一个密码函数，能够计算出其给定差分对应的概率是对该密码算法进行差分分析的前提。

模 2^n 加变换是分组密码设计中经常使用的编码环节，如 IDEA[3], RC6[4], SAFER 等密码算法都使用了这个变换。要分析使用了模 2^n 加变换的分组密码抗异或差分攻击的能力，首先要弄清模 2^n 加变换的异或差分概率的内在规律。因此，计算模 2^n 加变换的差分概率对于分析分组密码的抗差分攻击能力有一定的实用价值。文献[5]给出了模 2^n 加变换 $f(x,y)=(x+y) \bmod 2^n$ 的异或差分概率的一种计算方法，该方法首先计算出 8 个 2 级方阵，然后利用输入差和输出差从中选出 n 个矩阵，最后利用这 n 个矩阵和 2 个二维向量计算出该差分对应的概率。本文分析了该计算算法的计算复杂性，并基于以空间换时间的思想对该算法进行了改进，改进算法的计算复杂性小于现有方法计算复杂性的 7.7%，从而有效地降低了计算差分概率的计算复杂性。

2 计算模 2^n 加变换异或差分概率计算复杂性分析

模 2^n 加变换 $f(x,y)=(x+y) \bmod 2^n$ 可以看作 $\{0,1\}^n \times \{0,1\}^n$ 至 $\{0,1\}^n$ 的函数。$\forall x \in Z/(2^n)$，记 $x=\sum_{i=1}^{n} x_i 2^{i-1}$ 且 $x_i \in \{0,1\}$，则 x 与 $(x_n, x_{n-1}, \cdots, x_1) \in \{0,1\}^n$ 一一对应。以下本文中 x_i 的含义均如上所示，并称 x_i 为 x 的第 i 比特，并记+为模 2^n 加，$\oplus$ 为逐位模 2 加，记

$$p((\alpha,\beta)\to\gamma)=\frac{1}{2^{2n}}\#\{(x,y)\in\{0,1\}^n\times\{0,1\}^n:$$
$$[(x\oplus\alpha)+(y\oplus\beta)]\oplus(x+y)=\gamma\}$$

则称 $p((\alpha,\beta)\to\gamma)$ 为二元模 2^n 加变换的差分对应 $(\alpha,\beta)\to\gamma$ 的异或差分概率。

文献[5]给出了 $p((\alpha,\beta)\to\gamma)$ 的一个计算算法，该算法将 $p((\alpha,\beta)\to\gamma)$ 表示为若干矩阵的乘积，记

$$\boldsymbol{A}_0=\frac{1}{2}\begin{pmatrix}2&0\\0&0\end{pmatrix},\boldsymbol{A}_1=\frac{1}{2}\begin{pmatrix}0&1\\0&1\end{pmatrix},\boldsymbol{A}_2=\frac{1}{2}\begin{pmatrix}0&1\\0&1\end{pmatrix},\boldsymbol{A}_3=\frac{1}{2}\begin{pmatrix}1&0\\1&0\end{pmatrix}$$
$$\boldsymbol{A}_4=\frac{1}{2}\begin{pmatrix}0&1\\0&1\end{pmatrix},\boldsymbol{A}_5=\frac{1}{2}\begin{pmatrix}1&0\\1&0\end{pmatrix},\boldsymbol{A}_6=\frac{1}{2}\begin{pmatrix}1&0\\1&0\end{pmatrix},\boldsymbol{A}_7=\frac{1}{2}\begin{pmatrix}0&0\\0&2\end{pmatrix}$$

和 $L=(1\quad 1)$，$C=(1\quad 0)^t$，且 $1\le i\le n$，记 $w_i=4\alpha_i+2\beta_i+\gamma_i$，则有

$$p((\alpha,\beta)\to\gamma)=LA_{w_1}A_{w_2}\cdots A_{w_n}C$$

下面分析直接利用矩阵乘积的方法计算 $p((\alpha,\beta)\to\gamma)$ 的计算复杂性：

(1)该方法需要计算出诸 w_i，因而需要分别计算出 α,β,γ 的二进制表示：

$$(\alpha_1,\alpha_2,\cdots,\alpha_n),(\beta_1,\beta_2,\cdots,\beta_n),(\gamma_1,\gamma_3,\cdots,\gamma_n)$$

由于取一个计算机字的最高比特和最低比特需 1 次操作、取其他位置上的一个比特均需 2 次操作，因此该过程约需要 $(2n-2)\times 3$ 条指令。

(2)该算法需要对于所有的 $1\le i\le n$，计算出 $w_i=4\alpha_i+2\beta_i+\gamma_i$ 的值并据此选出矩阵 $\boldsymbol{A}_{w_i}$，该过程需要

基金项目：河南省杰出青年科学基金资助项目(0312001800)
作者简介：张庆贵(1963－)，男，博士，主研方向：密码学
收稿日期：2009-10-23 **E-mail**：zhangqingui@126.com

$4n+n=5n$ 条指令。

(3)该方法需要利用选出的矩阵 $\boldsymbol{A}_{w_i}$ 计算出 $LA_{w_1}A_{w_2}\cdots A_{w_n}C$ 的值。由于 $L=(1\ \ 1)$，$C=(1\ \ 0)^t$，因此该过程需要做 1 次取 $\boldsymbol{A}_{w_n}$ 的第 1 行操作(2 条指令)、$n-1$次二维向量与 2×2 矩阵相乘(6 条指令)和 1 次二维向量的 2 个分量相加(1 条指令)，共需 $6(n-1)+2+1=6n-3$条指令。因此，计算出 $p((\alpha,\beta)\to\gamma)$ 的值共需$17n-9$ 条指令。

根据上述分析结果，当 $n=32$ 时，计算出 $p((\alpha,\beta)\to\gamma)$ 的值需要 535 条指令。假若在某个密码分析问题中，需要对固定的输出差 γ，找出在 α,β 遍历所有可能时，使 $p((\alpha,\beta)\to\gamma)$ 达到最大的差分对应 $(\alpha,\beta)\to\gamma$，则此时的计算复杂性为 $(17n-9)\times4^n$，该计算复杂性将随着 n 的增大而增大。当 n 较大时，就会因其计算量过大而难以实现。因此，需要研究并给出计算 $p((\alpha,\beta)\to\gamma)$ 的一种新的实现方法。

3 计算模 2^n 加变换异或差分概率的改进方法

下面对计算模 2^n 加变换的异或差分概率直接计算方法进行改进，并提出一种新的实现方法。该方法的主要思路是首先对 (α,β,γ) 的若干连续分量的每个可能取值，预先计算出矩阵的乘积，并以 (α,β,γ) 的这些连续分量为地址造表存储，从而采取以空间换时间的方法，大幅度降低计算复杂性。因此，在具体计算时，而只需由输入差和输出差的值 (α,β,γ) 查出对应矩阵的乘积，并将其相乘得到最终结果。由于显著减少了矩阵乘积的个数，且不需计算出 w 的各个分量，因此这种以空间换时间的思想，大大降低了其计算的复杂性。

由于在现有的密码算法中，对输入和输出大多数是按照字节或字节的倍数进行分组，因此下文中仅考虑 $n\geq8$ 且 $n \bmod 8=0$ 的情况。

预处理算法：

对于任意的 $a=(a_8,\cdots,a_2,a_1),b=(b_8,\cdots,b_2,b_1),c=(c_8,\cdots,c_2,c_1)\in\{0,1\}^8$，计算出 $w_1,w_2,\cdots,w_8$，其中，$w_i=4a_i+2b_i+c_i$。当 $n>8$ 时，计算出 2×2 矩阵 $\boldsymbol{M}(a,b,c)=A_{w_1}A_{w_2}\cdots A_{w_8}$ 和 2 个 2 维向量：

$$\boldsymbol{M}_L(a,b,c)=LA_{w_1}A_{w_2}\cdots A_{w_8}=L\times\boldsymbol{M}(a,b,c)$$

$$\boldsymbol{M}_R(a,b,c)=A_{w_1}A_{w_2}\cdots A_{w_8}C=\boldsymbol{M}(a,b,c)\times C$$

并将它们存储；当 $n=8$ 时，只需直接计算并存储

$$p((\alpha,\beta)\to\gamma)=LA_{w_1}A_{w_2}\cdots A_{w_n}C$$

差分概率的计算算法：

输入 输入差和输出差 (α,β,γ)

输出 差分对应 $(\alpha,\beta)\to\gamma$ 的概率

Step1 计算出 α,β,γ 的各个字节 $\alpha^{(1)},\cdots,\alpha^{(n/8)},\beta^{(1)},\cdots,\beta^{(n/8)},\gamma^{(1)},\cdots,\gamma^{(n/8)}$，其中，序号小的是低位字节。

Step 2 当 n=8时，通过查表查出 $p((\alpha,\beta)\to\gamma)$；当 n>8时，计算出差分概率：

$$p((\alpha,\beta)\to\gamma)=\boldsymbol{M}_L(\alpha^{(1)},\beta^{(1)},\gamma^{(1)})\times\boldsymbol{M}(\alpha^{(2)},\beta^{(2)},\gamma^{(2)})\times\cdots\times\boldsymbol{M}(\alpha^{(n/8-1)},\beta^{(n/8-1)},\gamma^{(n/8-1)})\times\boldsymbol{M}_R(\alpha^{(n/8)},\beta^{(n/8)},\gamma^{(n/8)})$$

下面分析上述以空间换时间的新算法的计算复杂性。

由于预处理过程与具体的输入差和输出差无关，因此可以预先计算好并存储下来，故其计算复杂性不必计入计算差分概率时的计算复杂性，以下假设 $n>8$。

在差分概率的计算算法中，如果输入差和输出差都是以字节方式给出，则 Step1 可以省略。Step2 需要完成 $n/8-2$ 次 2×2 矩阵与 2 维向量的乘积的计算(6 条指令)，完成 1 次 2 维行向量与 2 维列向量的乘积的计算(3 条指令)，因而完成 Step2 需要执行 $(n/8-2)\times6+3=3n/4-9$ 条指令。此时其计算复杂性是直接计算方法的 $\dfrac{3n/4-9}{17n-9}=(1-\dfrac{585}{51n-27})\times\dfrac{3}{17}$ 倍。

如果输入差和输出差都是以 32 bit 的方式给出，则 Step1 计算出一个字节平均需要 1.5 条指令，因而此时新的实现方法共需要 $(n/8)\times1.5\times3+3n/4-9=1.312\ 5n-9$ 条指令，此时其计算复杂性是直接计算方法的 $\dfrac{1.312\ 5n-9}{17n-9}\approx0.077-\dfrac{10.87}{22.3n-24.9}<0.077$ 倍。当 $n=32$ 时，利用新的实现方法计算出 $p((\alpha,\beta)\to\gamma)$ 的值只需 33 条指令，如果直接按定义计算，则需要 535 条指令，因而是直接计算时复杂性的 6%；当 $n=64$ 时，新的实现方法的计算复杂性是现有方法的 7%，故本文提出的新实现方法有效降低了计算模 2^n 加变换的异或差分概率的计算复杂性。特别地当需要计算一大批差分对应的最大值差分概率时，所降低计算复杂性相当可观。

最后分析该算法的存储复杂性。新的实现方法需要存储一个 2×2 矩阵 $\boldsymbol{M}(a,b,c)$和 2 维向量 $\boldsymbol{M}_L(a,b,c)$与 $\boldsymbol{M}_R(a,b,c)$，因而存储量是 $4\times2^{24}+2\times2\times2^{24}=2^{27}$。当利用 C 语言实现时，由于 2^{27} 太大，因此无法采取开数组的方法而只能采取写入文件的方案实现。如果希望进一步降低实现算法的存储量，可以按 5 bit 为一块或 4 bit 为一块对输入差和输出差进行分拆，采取类似的方法进行预计算和实际计算。

当以 4 bit 为一块进行分拆时，预计算算法中的 $a,b,c\in\{0,1\}^4$，此时的存储量是 $4\times2^{12}+2\times2\times2^{12}=2^{15}$，完全可以采取开辟数组的方法利用 C 语言实现，此时的计算复杂性是 $(n/4)\times1.5\times3+[(n/4-2)\times6+3]=2.625n-9$ 条指令，是直接计算方法的 $\dfrac{2.625n-9}{17n-9}\approx0.154-\dfrac{7.61}{17n-9}<0.154$倍。当 $n=32$ 时，其计算复杂性是 75 条指令，是直接计算时复杂性的 14%；当 $n=64$ 时，其计算复杂性是 159 条指令，是直接计算时复杂性的 14.7%。

4 结束语

本文分析了文献[5]提出的计算模 2^n 加变换的异或差分概率的计算复杂性，并利用以空间换时间的思想对该算法进行了改进，改进后的算法有效地将计算模 2^n 加变换的异或差分概率的计算复杂性降低为直接计算时计算复杂性的 7.7%以下，从而大大降低了计算模 2^n 加变换的差分概率的计算复杂性。

参考文献

[1] Biham E, Shamir A. Differential Cryptanalysis of DES-like Cryptosystems[J]. Journal of Cryptology, 1991, 4(1): 3-74.

[2] Biham E, Shamir A. Differential Cryptanalysis of the Data Encryption Standard[M]. [S. 1.]: Springer-Verlag, 1993.

[3] Rivest R, Robshaw M, Sidney R, et al. The RC6 Block Cipher[C]//Proc. of the 1st AES Candidate Conference. NY, USA: [s. n.], 1998.

[4] Lai Xuejia. On the Design and Security of Block Ciphers[M]. [S. 1.]: Hartung-Gorre Verlag, 1992.

[5] WalleÂn J. On the Differential and Linear Properties of Addition[EB/OL]. (2003-12-20). http://www.tcs.hut.fi.

编辑 索书志