第35卷 第18期 计 算 机 工 程 2009年9月
Vol.35 No.18 Computer Engineering September 2009
·安全技术· 文章编号：1000—3428(2009)18—0125—02 文献标识码：A 中图分类号：TN918.1

改进 Py 区分攻击算法的计算复杂性分析

陈士伟，金晨辉
(解放军信息工程大学电子技术学院，郑州 450004)

摘 要：针对 Crowley P 提出的一种改进的 Py 区分攻击算法，利用直接计算的方法分析该算法的计算复杂性。基于以空间换时间的思想提出实现该算法的一种新的方法。结果表明，该方法能有效地将该区分攻击的计算复杂性降为直接计算所需计算复杂性的 1/14。
关键词：Py 算法；区分攻击；计算复杂性

Computational Complexity Analysis on Improved Py Distinguish Attack Algorithm

CHEN Shi-wei, JIN Chen-hui
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】 Using the way of computing directly, this paper analyzes the computation complexity of the improved Py distinguish attack algorithm proposed by Crowley P. Using the concept of replacing time with space, this paper presents a new method to implement the attack which can reduce the complexity to 1/14 times of that of using the way of computing directly.
【Key words】 Py algorithm; distinguish attack; computational complexity

1 概述

按分析目的可将序列密码的分析方法分为密钥恢复攻击和区分攻击 2 大类。随着序列密码设计的复杂程度的不断提高，密钥恢复攻击的难度也变得越来越大，因此，密码分析者把注意力转向了看似更为容易的区分攻击。对序列密码的区分攻击是要找出 t，并利用第 i 个乱数至第 $i+t$ 个乱数，构造出一个不在 $\{0, 1\}^m$ 上均匀分布的 m 维二元随机向量 $\xi_i(k, IV)$，并借助于此将乱数序列与随机序列区分开来，这里 $\xi_i(k, IV)$ 独立同分布。区分方法主要有 2 类：

(1)对固定的 IV 和密钥 k，变动 i，借助 $\{\xi_i(k,IV)\}_{i=1}^{\infty}$ 进行区分；

(2)对固定 i，变动 (k, IV)，借助 $\{\xi_i(k,IV):(k,IV)\in\Omega\}$ 进行区分。

Py 算法[1]是 eSTREAM 工程的候选算法之一，它是同步流密码算法，包含 1 300 Byte 的内部状态，每一步输出 8 Byte，将其作为 2 个 32 比特字来加密明文。Py 算法是 eSTREAM 工程中软件实现最快的候选算法之一。

文献[2]定义了事件 L(给定的 6 个条件同时成立的事件且 $p(L)=2^{-41.91}$)并证明了在事件 L 发生的条件下，Py 算法的 2 个输出变量 $O_{1,1}$ 和 $O_{2,3}$ 满足 $O_{1,1}=[(S\oplus A)+B]\bmod 2^{32}$ 与 $O_{2,3}=[(S\oplus B)+A]\bmod 2^{32}$，然后利用 $O_{1,1}$ 和 $O_{2,3}$ 的最低比特一定相等的特性提出了对 Py 算法的区分攻击。当要求区分优势为 0.52 时，该区分攻击需要 $2^{84.8}$ 个 IV 或密钥产生的乱数。

随后，文献[3]利用隐藏的 Markov 模型，给出 $\eta=(O_{1,1}, O_{2,3})$ 的概率取值及其平方和的快速计算算法，从而借助于随机变量 η 分布的不平衡性提出对 Py 算法的改进区分攻击，该方法需要的 IV 或密钥产生的乱数是文献[2]中方法的 1/60 552。文献[2-3]虽然对所提出的区分攻击的数据复杂性进行分析，但并没有分析区分攻击实现的计算复杂性。

本文针对 Crowley P 提出的 Py 算法区分攻击的计算复杂性进行分析，并提出一种新的实现方法，该方法有效地将该区分攻击的计算复杂性降为 $78\times 2^{68.9}$。

2 改进 Py 区分攻击算法的计算复杂性分析

记 $\varepsilon=2^m\sum_{a\in\{0,1\}^m}[p(\xi_i(k, IV)=a)-2^{-m}]^2$，$\Phi$ 是标准正态分布的概率函数。由文献[4]的证明可知，当 N 充分大时，将一条与 $\xi_i(k, IV)$ 同分布的序列判定为不服从均匀分布的概率与将一条均匀分布的序列判定为不服从均匀分布的概率之差(即区分优势[4])是 $P=1-2\Phi(-\sqrt{N\varepsilon}/2)$，因此，当 $N=\varepsilon^{-1}$，有 $P=1-2\Phi(-0.5)\approx 0.38$；当 $N=2\varepsilon^{-1}$，有 $P\approx 0.52$。这说明在要求区分优势为 0.52 的条件下，第 1 类区分攻击所需乱数长度是 $2\varepsilon^{-1}$，第 2 类区分攻击则需 $2\varepsilon^{-1}$ 个 (k,IV) 产生的乱数片段。

在所需的样本个数 N 确定以后，利用下面具体的区分方法判断乱数序列是否服从均匀分析。具体的区分方法是由给定的 $N\geq 2\varepsilon^{-1}$ 个样本 $a^{(1)},a^{(2)},\cdots,a^{(N)}$ 计算出诸 $p(\xi=a^{(i)})$，从而计算出判据 $T=\sum_{i=1}^{N}\mathrm{lb}(2^m p(\xi=a^{(i)}))$。当 $T>0$ 时，判定乱数序列不服从均匀分布，否则判定乱数序列服从均匀分布。故 2 类区分攻击的计算复杂性均为 $N\times C$，这里 C 为计算

基金项目：河南省杰出青年科学基金资助项目(0312001800)
作者简介：陈士伟(1983－)，女，硕士，主研方向：密码学；金晨辉，教授、博士生导师
收稿日期：2009-01-20 **E-mail：**chenshiwei1012@sohu.com

$\mathrm{lb}(2^m p(\xi=a^{(i)}))$ 的计算量。因此，当数据复杂性 N 确定以后，区分攻击的计算复杂性由 $\mathrm{lb}(2^m p(\xi=a_i))$ 的计算量决定。

文献[3]利用隐藏的 Markov 模型，得到事件 L 发生的条件下，$O_{1,1}$ 和 $O_{2,3}$ 的联合概率分布的隐式计算公式：

$$p((O_{1,1},\ O_{2,3})=(a,\ b)\mid L)=(1,\ 1,\ 1,\ 1)\boldsymbol{M}_{(a_{32},\ b_{32})}\cdots\boldsymbol{M}_{(a_1,\ b_1)}(1,\ 0,\ 0,\ 0)^{\mathrm{T}}$$

其中

$$\boldsymbol{M}_{(0,\ 0)}=\begin{pmatrix}8^{-1}&0&0&0\\8^{-1}&0&0&0\\8^{-1}&0&0&0\\8^{-1}&0&0&2^{-1}\end{pmatrix}$$

$$\boldsymbol{M}_{(0,\ 1)}=\begin{pmatrix}0&8^{-1}&0&0\\0&8^{-1}&0&0\\0&8^{-1}&2^{-1}&0\\0&8^{-1}&0&0\end{pmatrix}$$

$$\boldsymbol{M}_{(1,\ 0)}=\begin{pmatrix}0&0&8^{-1}&0\\0&2^{-1}&8^{-1}&0\\0&0&8^{-1}&0\\0&0&8^{-1}&0\end{pmatrix}$$

$$\boldsymbol{M}_{(1,\ 1)}=\begin{pmatrix}2^{-1}&0&0&8^{-1}\\0&0&0&8^{-1}\\0&0&0&8^{-1}\\0&0&0&8^{-1}\end{pmatrix}$$

并给出了联合概率取值的平方和的快速递推计算算法。利用所得结果提出对 Py 算法的改进区分攻击，该区分攻击需要 $2^{68.9}$ 个 $(k,\ IV)$ 产生的乱数。因此，Crowley 提出的区分攻击的计算复杂性为 $2^{68.9}\times C$，即计算区分攻击判据的计算复杂性。

下文分析利用文献[3]的计算公式计算判据的计算复杂性，进而确定出该区分攻击的计算复杂性。计算判据 T：

$$T=\sum_{i=1}^{N}\mathrm{lb}(2^{64}p((O_{1,1},O_{2,3})=(a^{(i)},b^{(i)}))=$$

$$\sum_{i=1}^{N}\mathrm{lb}\{2^{64}[p(L)(p((O_{1,1},O_{2,3})=(a^{(i)},\ b^{(i)})\mid L)-2^{-64})+2^{-64}]\}=$$

$$\sum_{i=1}^{N}\mathrm{lb}\{[2^{64}p(L)(1,\ 1,\ 1,\ 1)\boldsymbol{M}_{(a_{32}^{(i)},\ b_{32}^{(i)})}\cdots\boldsymbol{M}_{(a_1^{(i)},\ b_1^{(i)})}(1,\ 0,\ 0,\ 0)^{\mathrm{T}})-p(L)+1]\}$$

如果直接计算上式，则对于任意一个 $(a^{(i)},\ b^{(i)})\in\{0,\ 1\}^{64},\ i=1,\ 2,\cdots,\ N$，需要首先计算出 $(a^{(i)},\ b^{(i)})$ 的各个比特 $((a_1^{(i)},\ b_1^{(i)}),\ (a_2^{(i)},\ b_2^{(i)}),\cdots,\ (a_{32}^{(i)},\ b_{32}^{(i)}))$，该过程需要 124 条指令；再依据 $(a_j^{(i)},\ b_j^{(i)})$ 的值得到 $\boldsymbol{M}_{(a_j^{(i)},\ b_j^{(i)})}$，该过程需 32 条指令；然后计算 32 次 4×4 矩阵与 4 维向量相乘、1 次向量与向量相乘、2 次乘法运算、1 次减法运算、1 次加法运算及 1 次对数运算，才可得到 $\mathrm{lb}\{[2^{64}p(L)(1,\ 1,\ 1,\ 1)\boldsymbol{M}_{(a_{32}^{(i)},\ b_{32}^{(i)})}\cdots\boldsymbol{M}_{(a_2^{(i)},\ b_2^{(i)})}\boldsymbol{M}_{(a_1^{(i)},\ b_1^{(i)})}(1,\ 0,\ 0,\ 0)^{\mathrm{T}})-p(L)+1]\}$ 的值，这些运算共需 1 064 条指令，因此，该区分攻击所需的计算复杂性为 $1\ 064\times2^{68.9}$。

由于直接计算判据所需的计算量太大，本文提出一种新的实现方法以降低判据计算的计算量。

3 一种新的 Crowley 的 Py 区分攻击的实现方法

利用空间换时间的思想，首先将输出乱数 $(a^{(i)},b^{(i)})$ 分为 4 Byte，针对 1 Byte 的所有取值，预计算出所对应的矩阵的乘积，然后以字节为索引地址将结果存储。因此，在具体计算时只需依据 $(a^{(i)},\ b^{(i)})$ 各个字节的值查表得到对应的矩阵乘积，然后将结果相乘可得最终结果，这样就显著减少了矩阵相乘的个数，从而大大降低了区分攻击的计算复杂性。算法预处理过程为：对任意 $a=(a_1,\ a_2,\cdots,a_8),\ b=(b_1,\ b_2,\cdots,b_8)\in\{0,\ 1\}^8$，计算

$$\boldsymbol{M}(a,\ b)=\boldsymbol{M}_{(a_8,\ b_8)}\boldsymbol{M}_{(a_7,\ b_7)}\cdots\boldsymbol{M}_{(a_1,\ b_1)}$$

$$\boldsymbol{M}_R(a,b)=\boldsymbol{M}_{(a_8,b_8)}\cdots\boldsymbol{M}_{(a_2,b_2)}\boldsymbol{M}_{(a_1,b_1)}(1000)^{\mathrm{T}}$$

$$\boldsymbol{M}_L(a,\ b)=(1111)\boldsymbol{M}_{(a_8,\ b_8)}\boldsymbol{M}_{(a_7,\ b_7)}\cdots\boldsymbol{M}_{(a_1,\ b_1)}$$

记

$$L(a^{(i)},\ b^{(i)})=\ln\{[2^{64}p(L)((1,\ 1,\ 1,\ 1)\boldsymbol{M}_{(a_{32}^{(i)},b_{32}^{(i)})}\cdots\boldsymbol{M}_{(a_2^{(i)},b_2^{(i)})}\boldsymbol{M}_{(a_1^{(i)},b_1^{(i)})}$$

$$\boldsymbol{M}_{(a_2^{(i)},b_2^{(i)})}\boldsymbol{M}_{(a_1^{(i)},b_1^{(i)})}(1,\ 0,\ 0,\ 0)^{\mathrm{T}})-\ p(L)+1]\}$$

则计算 $L(a^{(i)},b^{(i)})$ 的算法如下：

输入 $a^{(i)},\ b^{(i)}$

输出 $L(a^{(i)},b^{(i)})$

(1)计算 $\mathrm{a}^{(i)},\mathrm{b}^{(i)}$ 的各个字节 $(\mathrm{a}_{\mathrm{byte1}}^{(i)},\ \mathrm{b}_{\mathrm{byte1}}^{(i)})$，$(\mathrm{a}_{\mathrm{byte2}}^{(i)},\ \mathrm{b}_{\mathrm{byte2}}^{(i)})$，$(\mathrm{a}_{\mathrm{byte3}}^{(i)},\ \mathrm{b}_{\mathrm{byte3}}^{(i)})$，$(\mathrm{a}_{\mathrm{byte4}}^{(i)},\ \mathrm{b}_{\mathrm{byte4}}^{(i)})$，其中，$(\mathrm{a}_{\mathrm{byte1}}^{(i)},\ \mathrm{b}_{\mathrm{byte1}}^{(i)})$ 为最低位字节；

(2)查表得到 $\mathbf{M}_{\mathrm{L}}(\mathrm{a}_{\mathrm{byte4}}^{(i)},\ \mathrm{b}_{\mathrm{byte4}}^{(i)})$，$\mathbf{M}(\mathrm{a}_{\mathrm{byte3}}^{(i)},\ \mathrm{b}_{\mathrm{byte3}}^{(i)})$，$\mathbf{M}(\mathrm{a}_{\mathrm{byte2}}^{(i)},\ \mathbf{M}(\mathrm{a}_{\mathrm{byte2}}^{(i)},\ \mathrm{b}_{\mathrm{byte2}}^{(i)})$，$\mathbf{M}_{\mathrm{R}}(\mathrm{a}_{\mathrm{byte1}}^{(i)},\ \mathrm{b}_{\mathrm{byte1}}^{(i)})$ 的值，并计算其乘积：

$$\mathrm{A}_{(i)}=\mathbf{M}_{\mathrm{L}}(\mathrm{a}_{\mathrm{byte4}}^{(i)},\ \mathrm{b}_{\mathrm{byte4}}^{(i)})\times\mathbf{M}(\mathrm{a}_{\mathrm{byte3}}^{(i)},\ \mathrm{b}_{\mathrm{byte3}}^{(i)})\times\mathbf{M}(\mathrm{a}_{\mathrm{byte2}}^{(i)},\ \mathrm{b}_{\mathrm{byte2}}^{(i)})\times\mathbf{M}_{\mathrm{R}}(\mathrm{a}_{\mathrm{byte1}}^{(i)},\ \mathrm{b}_{\mathrm{byte1}}^{(i)})$$

(3)计算 $\ln\{[2^{64}\mathrm{p(L)A}_{(i)}-\mathrm{p(L)}+1]\}$，并输出 $\mathrm{L}(\mathrm{a}^{(i)},\mathrm{b}^{(i)})$。

若输出乱数以字节存储，则(1)需 6 条指令；在(2)中，查表需 4 条指令，而向量与矩阵相乘需要 $63=28\times2+7(7=4$次乘法$+3$次加法$)$ 条指令；(3)须进行 1 次减法运算、1 次加法运算、2 次乘法运算和 1 次对数运算，共 5 条指令。故共需 78 条指令即可得到 $L(a^{(i)},\ b^{(i)})$ 的值，因此，利用上述算法可将文献[3]的区分攻击的计算复杂性降为 $78\times2^{68.9}$。

4 结束语

Crowley P 提出的对 Py 的改进区分攻击的计算复杂性主要取决于区分攻击判据的计算量。本文详细分析了该区分攻击的计算复杂性，并利用空间换时间的思想将区分攻击判据的计算复杂性降为直接计算所需计算复杂性的 1/14，从而将区分攻击实现的计算复杂性降为 $78\times2^{68.9}$。

参考文献

[1] Biham E, Neito S J G. Py(Roo): A Fast and Secure Stream Cipher using Rolling Arrays[EB/OL]. (2005-04-29). http://www.ecrypt.eu.org/stream.

[2] Sekar G, Paul S, Preneel B. Distinguishing Attacks on the Stream Cipher Py[EB/OL]. (2005-03-04). http://www.ecrypt.eu.org/stream.

[3] Crowley P. Improved Cryptanalysis of Py[EB/OL]. (2006-01-02). http://www.ecrypt.eu.org/stream.

[4] Baigneres T, Junod P, Vandenay S. How Far Can We Go Beyond Linear Cryptanalysis?[C]//Proc. of ASIACRYPT'04. Berlin, Germany: Springer Verlag, 2004.

编辑　金胡考