软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(7):1899−1907 [doi: 10.3724/SP.J.1001.2012.04123]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

BOMM 算法的密码学性质*

杨 笑[1,2+], 范修斌[1], 武传坤[1], 余玉银[1,2], 冯秀涛[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 100049)

Cryptographic Properties of BOMM

YANG Xiao[1,2+], FAN Xiu-Bin[1], WU Chuan-Kun[1], YU Yu-Yin[1,2], FENG Xiu-Tao[1]

[1](The State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: rident@163.com

Yang X, Fan XB, Wu CK, Yu YY, Feng XT. Cryptographic properties of BOMM. *Journal of Software*, 2012,23(7):1899−1907 (in Chinese). http://www.jos.org.cn/1000-9825/4123.htm

Abstract: BOMM is a byte-oriented mixed type algorithm with memory, which is used to disorder a given byte sequence. It has been used as a main component in a new stream cipher called Loiss for having many good cryptographic properties. This paper builds an algebraic equation system with degree 5 for BOMM, and based on this equation system, discusses the complexity of algebraic attack on Loiss. In addition, the paper also discusses the statistic weakness of BOMM and gives an analysis of the security of Loiss under a specific class of weak keys.

Key words: BOMM (byte-oriented memorial mixer); Loiss; stream cipher; algebraic attack; period

摘 要: BOMM(byte-oriented memorial mixer)算法是一种基于字节操作的混合型带记忆的序列扰乱算法,因具备良好的密码学性质,一个新的流密码算法 Loiss 使用了它作为主要组件.建立了 BOMM 算法的 5 次代数方程系统,在此基础上讨论了针对 Loiss 算法的代数攻击的复杂度.此外还发现了 BOMM 算法的一个统计弱点,并分析了 Loiss 算法在一类弱密钥下的安全性.

关键词: BOMM 算法;Loiss 算法;流密码;代数攻击;周期

中图法分类号: TP309 **文献标识码**: A

BOMM(byte-oriented memorial mixer)算法[1,2]是一种基于字节操作的混合型带记忆逻辑,对输入的字节序列与自身的内部记忆单元进行叠加和搅拌等操作,输出一个具有更优统计性质的字节序列.叠加和混淆是密码算法设计中常用的两种操作,具有实现代价小、运行速度快等特点.文献[1]说明混合使用这两种操作具有更高的安全性,认为可以有效抵抗相关攻击.由于 BOMM 算法具有这些优越性,面向字节的流密码 Loiss 算法[3]中使用了 BOMM 算法作为主要的组件.Loiss 算法包含线性反馈移位寄存器(LFSR)、非线性函数 F 和 BOMM 算法这 3 部分,LFSR 作为驱动器产生长周期状态序列,非线性函数抽取 LFSR 部分记忆单元组合一个字节序列,BOMM 算法对这个字节序列进一步扰乱混淆并输出相应的字节序列.最后,再与 LFSR 的输出字节序列逐比

* 基金项目: 国家自然科学基金(60833008, 60902024)
收稿时间: 2010-10-20; 定稿时间: 2011-09-01

特异或生成密钥流字节序列来加/解密通信中的数据流.

本文主要考虑 BOMM 算法的统计性质和代数分析,并从这两个角度分析了 Loiss 算法的安全性.统计性质是分析密码算法的常用手段,常见的线性攻击、差分攻击等都是基于概率和统计的思想.代数分析[4]是 Courtois 等人在 2003 年欧密会提出的,采用了基于代数思想的方法和技巧,将一种密码算法的安全性完全归约到求解一个超定的多变元高次方程系统的问题,这与以前大都是基于概率思想的分析方法有着很大的不同.迄今为止,已被代数攻击攻破的密码系统大多是基于 LFSR 设计的,例如 Toyocrypt,LILI-128,SFINKS,E0.

本文建立 BOMM 算法的布尔方程系统,在此基础上讨论了针对 Loiss 算法的代数攻击的复杂度.此外,我们还发现了 BOMM 算法的一个统计弱点,提出 Loiss 算法可能存在一类弱密钥.虽然目前我们认为这类弱密钥存在的概率几乎为 0,但是还不能从理论上给出很好的证明.

1 BOMM 算法的结构和性质

BOMM 算法是一种基于字节操作的混合型带记忆逻辑,包括 16 字节的记忆单元,使用一个输入输出均为 8 比特的正形置换 S_2 作为非线性变换,输入字节序列$\{x^{(t)}\}_{t\geqslant n}$,通过搅拌叠加等操作,输出字节序列$\{y^{(t)}\}_{t\geqslant 0}$.如图 1 所示.具体描述如下:

(1) 预置记忆单元 $y_0^{(0)}, y_1^{(0)}, ..., y_{15}^{(0)}$;

(2) 对输入的字节序列 $X=(x^{(0)},x^{(1)},x^{(2)},\ldots)$的每个字节 $x^{(t)}$做:

① 取出 $x^{(t)}$的高 4 比特将其记作 h,取出 $x^{(t)}$的低 4 比特将其记作 l;

② 输出 $y^{(t)} = y_h^{(t)} \oplus x^{(t)}$;

③ 更新记忆状态:

$$\begin{aligned} y_l^{(t+1)} &= y_l^{(t)} \oplus S_2(x^{(t)}) \\ y_h^{(t+1)} &= \begin{cases} y_h^{(t)} \oplus S_2(y_l^{(t+1)}), & \text{当} h \neq l \text{时} \\ y_l^{(t+1)} \oplus S_2(y_l^{(t+1)}), & \text{当} h = l \text{时} \end{cases} \\ y_i^{(t+1)} &= y_i^{(t)}, i \neq l, h \end{aligned} \tag{1}$$

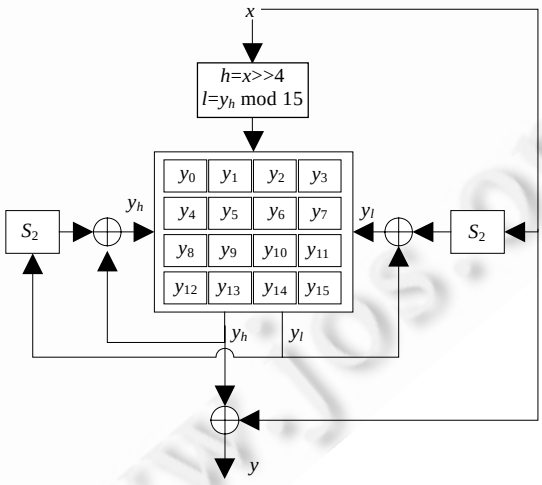


Fig.1 BOMM algorithm

图 1 BOMM 算法

BOMM 算法的设计者认为,该算法在抗击相关攻击、分别征服攻击、相关密钥攻击和区分攻击等方面都有很好的性能[1].此外,BOMM 还具备如下性质:

(1) 平衡性

将 BOMM 算法的记忆单元 $y_0^{(t)},y_1^{(t)},...,y_{15}^{(t)}$ 以及输入字节 $x^{(t)}$看作$\{0,1\}^8$ 上独立均匀分布的随机变量.称 BOMM 是平衡的,如果对任意 $a\in\{0,1\}^8$,$0\leqslant i\leqslant 15$,$t\geqslant 0$ 都有 $\Pr(y_i^{(t)}=a)=1/256$ 成立.文献[3]证明 BOMM 是平衡的,当且仅当 S_2 是正形置换.

(2) 状态更新和产生输出字节的过程可以用代数(布尔)方程描述

BOMM 算法的状态更新的过程可以表示成 72 个 2 次方程、136 个 5 次的布尔方程,同时引入 24 个中间变元.BOMM 算法的输出过程可以表示为 8 个 5 次方程和 8 个线性方程,引入了 8 个中间变量.第 3 节详细阐述了建立方程的方法以及所得方程等式的具体形式.

(3) 具有周期性的输入序列对应的输出序列表现出一定的线性关系

假设 BOMM 算法的输入字节序列$\{x^{(t)}\}_{t\geqslant 0}$是周期为 k 的周期序列,那么其输出字节序列$\{y^{(t)}\}_{t\geqslant 0}$有一组线性关系:对任意整数 $i\geqslant 0$, $z^{(t_0+ik+4k)}=z^{(t_0+ik)}$, 且这组线性关系成立的概率不小于$\frac{15}{16}+\left(\frac{14}{16}\right)^{2k-2}$.第 4 节对这一性质给出了详细证明.

2 Loiss 算法简介

面向字节的流密码算法 Loiss 是 BOMM 算法的一个应用实例,本节简要描述该算法的结构和工作流程.Loiss 算法在 128 比特的初始密钥和 128 比特的初始向量的控制下,生成密钥字节序列 z.密钥字节序列 z 可以用来加密、解密通信中的数据流.算法整体上包含 3 层逻辑结构:线性反馈移位寄存器 LFSR、非线性函数 F 和 BOMM,如图 2 所示.

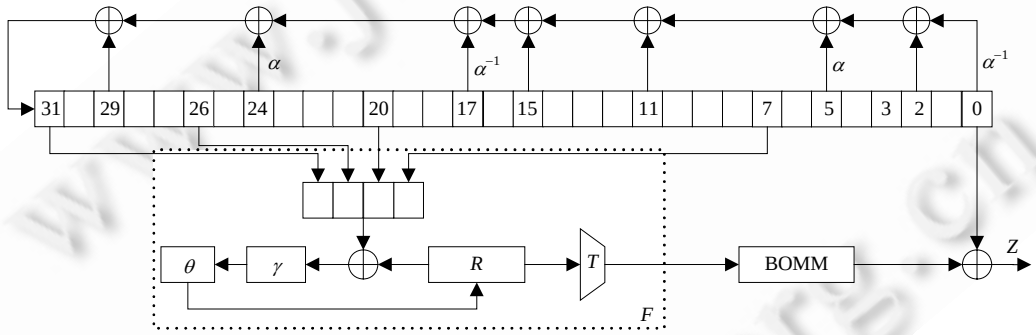


Fig.2 Structure of Loiss

图 2 Loiss 算法整体结构

线性反馈移位寄存器(LFSR)定义在有限域 F_{2^8} 上,共包含 32 个字节寄存器单元 s_i,$0\leqslant i\leqslant 31$.其特征多项式为 $f(x)=x^{32}+x^{29}+\alpha x^{24}+\alpha^{-1}x^{17}+x^{15}+x^{11}+\alpha x^5+x^2+\alpha^{-1}\in F_{2^8}[x]$,其中,$\alpha$为二元域 F_2 上多项式 $x^8+x^7+x^5+x^3+1$ 的根.设 $\underline{s}=\{s_t\}_{t\geqslant 0}$ 为 $f(x)$生成的有限域 F_{2^8} 上的序列,则对任意 $t\geqslant 0$,有

$$s_{t+32}=s_{t+29}+\alpha s_{t+24}+\alpha^{-1}s_{t+17}+s_{t+15}+s_{t+11}+\alpha s_{t+5}+s_{t+2}+\alpha^{-1}s_t \tag{2}$$

非线性函数 F 是一个从 32 比特到 8 比特的压缩函数,其内部包含一个 32 比特的记忆单元 R.输入为 LFSR 的 4 个寄存器单元 s_7,s_{20},s_{26},s_{31} 的取值,输出一个字节.其中,S_1 是一个 8×8 的 S 盒置换;γ是由 4 个 S_1 并置而成;θ 是一个线性变换,与分组密码算法 SMS4 的扩散层变换相同;T 是一个截取函数,截取当前输入的最左边 8 比特组成字节并输出.

设 t 时刻非线性函数的输入为 $s_7^{(t)},s_{20}^{(t)},s_{26}^{(t)},s_{31}^{(t)}$,分别对应着 LFSR 寄存器单元 s_7,s_{20},s_{26},s_{31} 在 t 时刻的取值.设记忆单元 R 的取值为 $R^{(t)}$,更新为 $R^{(t+1)}$,非线性函数 F 的输出为 $x^{(t)}$.记 $W^{(t)}=s_{31}^{(t)}\,|\,s_{26}^{(t)}\,\|\,s_{20}^{(t)}\,\|\,s_7^{(t)}$,这里,$\|$表示子串

的连接,则有:

$$R^{(t+1)}=\theta(\gamma(R^{(t)}\oplus W^{(t)})),$$
$$x^{(t)}=T(R^{(t)}).$$

算法经过初始化过程后将进入密钥产生过程,在该过程中,算法每执行下列过程 1 次,输出 1 个密钥字节 z:

(1) 执行非线性函数 F 一次,更新 F 的记忆单元 R.记 F 的输出为 $x^{(t)}$,则 $x^{(t)}=F(\cdot)$;

(2) 将非线性函数 F 的输出 $x^{(t)}$作为 BOMM 的输入,执行 BOMM 操作一次,更新 BOMM 的内部状态.记 BOMM 的输出为 $y^{(t)}$,则 $y^{(t)}=BOMM(x^{(t)})$;

(3) 输出密钥字节 $z^{(t)}=y^{(t)}\oplus s_0^{(t)}$;

(4) 运行 LFSR 一次,更新 LFSR 的内部状态.

3 BOMM 算法的代数分析

记 t 时刻 BOMM 算法内部的 16 个记忆单元分别为 $y_0^{(t)},y_1^{(t)},...,y_{15}^{(t)}$,每个字节单元 $y_i^{(t)}=([y_i^{(t)}]_7,[y_i^{(t)}]_6,...,[y_i^{(t)}]_0)$,$0\leqslant i\leqslant 15$,其中,$[y_i^{(t)}]_7$ 是字节的最高比特,$[y_i^{(t)}]_0$ 是最低比特.此外,记 BOMM 的输入字节为 $x^{(t)}$,输出字节为 $y^{(t)}$.在一个节拍的运行中,BOMM 算法首先产生一个字节的输出,然后更新内部状态,下面我们用代数(布尔)方程描述这两个过程.

3.1 BOMM算法的方程系统

我们引入选择函数 $h_i(x)$和 $l_i(x)$,其中,$h_i(x)$是 $F_2^8\to F_2$ 的布尔函数,当$(x>>4)=i$ 时取值为 1,否则取值为 0.$l_i(x)$也是 $F_2^8\to F_2$ 的布尔函数,当 $x \bmod 16=i$ 时,取值为 1,否则,取值为 0.由等式(1)易见

$$y_l^{(t)}=y_0^{(t)}\cdot l_0(x^{(t)})\oplus y_1^{(t)}\cdot l_1(x^{(t)})\oplus...\oplus y_{15}^{(t)}\cdot l_{15}(x^{(t)}) \tag{3}$$

需要注意的是,这里,$y_l^{(t)}=([y_l^{(t)}]_7,[y_l^{(t)}]_6,...,[y_l^{(t)}]_0)$ 是我们引入的中间变量.等式(3)中的'·'表示数乘.$l_i(x)$是 4 次布尔函数,这样,我们就得到了 8 个 5 次方程,变元包含 $y_l^{(t)},x^{(t)}$以及 BOMM 内部状态 $y_0^{(t)},...,y_{15}^{(t)}$.

更新序号为 h 和 l 的寄存器单元时使用了 S 盒 S_2.S_2 的代数免疫阶为 2,可以生成 36 个线性独立的 2 次隐式方程,方程的变元为 S_2 的输入和输出变量(用待定系数法容易给出这 36 个隐式方程表达式,具体方法参见文献[5]).更新 $y_l^{(t)}$ 时,S_2 的输入为 $x^{(t)}$,记此时的输出为向量 $S_l^{(t)}$;更新 $y_h^{(t)}$ 时,S_2 的输入为 $y_l^{(t)}\oplus S_l^{(t)}$,记其输出为向量 $S_h^{(t)}$.在这个环节,我们共引入了 2×16 个二元中间变量($S_l^{(t)}$ 和 $S_h^{(t)}$),利用 S_2 的代数性质可以分别得到 36 个关于 $x^{(t)},S_l^{(t)}$ 的 2 次布尔方程和 36 个关于 $y_l^{(t)},S_l^{(t)}$ 和 $S_h^{(t)}$ 的 2 次布尔方程.

对 BOMM 中 16 个寄存器单元的更新可表示为如下的布尔方程:

$$\begin{pmatrix}y_0^{(t+1)}\\y_1^{(t+1)}\\\vdots\\y_{15}^{(t+1)}\end{pmatrix}\oplus\begin{pmatrix}y_0^{(t)}\\y_1^{(t)}\\\vdots\\y_{15}^{(t)}\end{pmatrix}\oplus\begin{pmatrix}S_l^{(t)}\cdot l_0(x^{(t)})\\S_l^{(t)}\cdot l_1(x^{(t)})\\\vdots\\S_l^{(t)}\cdot l_{15}(x^{(t)})\end{pmatrix}\oplus\begin{pmatrix}S_h^{(t)}\cdot h_0(x^{(t)})\\S_h^{(t)}\cdot h_2(x^{(t)})\\\vdots\\S_h^{(t)}\cdot h_{15}(x^{(t)})\end{pmatrix}=0 \tag{4}$$

这个系统包含 16×8 个布尔方程,因为 $h_i(x)$的代数次数为 4,因此这 128 个布尔方程的代数次数为 5 次.

这样,在 BOMM 算法更新状态的过程中可以生成 36×2 个 2 次方程、16×8+8 个 5 次的布尔方程.引入新变量 3×8 个(即 $y_l^{(t)},S_l^{(t)}$ 和 $S_h^{(t)}$).

BOMM 算法的输出过程可表示为 $y^{(t)}=x^{(t)}\oplus y_h^{(t)}$,其中,$y_h^{(t)}$ 被看作中间变量.类似处理 $y_l^{(t)}$ 的方法,我们有

$$y_h^{(t)}=y_0^{(t)}\cdot h_0(x^{(t)})\oplus y_1^{(t)}\cdot h_1(x^{(t)})\oplus...\oplus y_{15}^{(t)}\cdot h_{15}(x^{(t)}) \tag{5}$$

这样,在 BOMM 的输出部分我们建立了 8 个 5 次方程和 8 个线性方程,引入了 8 个中间变量($y_h^{(t)}$).

3.2 降低方程代数次数的可能性

上一节在建立 BOMM 算法的方程系统时,我们引入了 4 次布尔函数 $l_i(x),h_i(x)$,$0\leqslant i\leqslant 15$.这些布尔函数存在

2 次隐函数,如果引入 2 次隐函数及必要的中间变量,那么就可以降低 Loiss 生成的方程系统的代数次数.本节主要讨论此方法的可行性.

$y=l_i(x)$有 4 个线性独立的 2 次隐式方程,记布尔函数 l_i 的输出为二元变量 $l_i^{(t)}$,这样,通过引入 16 个中间变量 $l_i^{(t)}, i=0,...,15$ 和 4×16 个 2 次方程,就可以将等式(3)降为 2 次方程:

$$y_l^{(t)} = y_0^{(t)} \cdot l_0^{(t)} \oplus y_1^{(t)} \cdot l_1^{(t)} \oplus ... \oplus y_{15}^{(t)} \cdot l_{15}^{(t)} \tag{6}$$

类似处理 $l_i(x)$的情况,$y=h_i(x)$也有 4 个线性独立的 2 次隐式方程,记 h_i 的输出为二元变量 $h_i^{(t)}$,通过引入这 16 个中间变量 $h_i^{(t)}, i=0,...,15$ 和 4×16 个 2 次方程,这样,等式(5)化为

$$y_h^{(t)} = y_0^{(t)} \cdot h_0^{(t)} \oplus y_1^{(t)} \cdot h_1^{(t)} \oplus ... \oplus y_{15}^{(t)} \cdot h_{15}^{(t)} \tag{7}$$

等式(4)化为

$$\begin{pmatrix} y_0^{(t+1)} \\ y_1^{(t+1)} \\ \vdots \\ y_{15}^{(t+1)} \end{pmatrix} \oplus \begin{pmatrix} y_0^{(t)} \\ y_1^{(t)} \\ \vdots \\ y_{15}^{(t)} \end{pmatrix} \oplus \begin{pmatrix} S_l^{(t)} \cdot l_0^{(t)} \\ S_l^{(t)} \cdot l_1^{(t)} \\ \vdots \\ S_l^{(t)} \cdot l_{15}^{(t)} \end{pmatrix} \oplus \begin{pmatrix} S_h^{(t)} \cdot h_0^{(t)} \\ S_h^{(t)} \cdot h_1^{(t)} \\ \vdots \\ S_h^{(t)} \cdot h_{15}^{(t)} \end{pmatrix} = 0 \tag{8}$$

通过上面的技术,我们把 Loiss 生成的方程的代数次数降到了 2 次.但是,上面引入的这些 2 次隐函数不能取代原布尔函数.以 $l_i(x)$为例:如果 $l_i(a)=1$,那么$(a,1),(a,0)$都满足 $l_i(x)$的 4 个线性无关的 2 次隐函数.也就是说,隐函数的使用扩大了方程系统的解空间.通过测试我们发现,$l_i(x)$的 3 次隐函数存在同样的问题.结合 $l_i(x),h_i(x)$的定义可以看出,上面所列出的代数方程(6)~方程(8)没有给出刻画 BOMM 算法的任何有效的数量关系.

因此,通过上面的技术虽然可以为 BOMM 算法建立一个 2 次的方程系统,但是这个系统不能准确描述 BOMM 的状态变量之间的数量关系.并且我们认为,简单地引入更多时刻输出字节,建立一个超定的方程系统[6]也不能有效解决问题,即求解这样的方程系统不可以恢复输入序列和 BOMM 的初始记忆单元.

3.3 在Loiss算法的代数分析中的应用

Loiss 算法的 LFSR 和非线性函数 F 的工作流程可以自然地转换为相应的代数(布尔)方程.Loiss 算法的内部状态共 416 比特,至少需要 416 比特的输出密钥流才能唯一确定 t 时刻的内部状态.我们使用连续 52 个时钟输出的密钥流字节建立代数方程:记这 52 个时刻的输出字节为 $Z^{(t)},Z^{(t+1)},\ldots,Z^{(t+51)}$.在连续 52 时钟里,算法运行了 51 次状态更新过程,t 时刻状态变量及输出方程引入 52 次.

容易列出 Loiss 算法运行过程对应的方程系统,得到的方程等式分为两组:一组是关于 t 时刻状态和输出变量的,另一组是关于状态更新过程的,这两组方程的次数、引入中间变量个数等信息见表 1.在最终包含 52 个时刻的方程系统中使用的中间变量有 56×51+16×52=3688 个,状态变量 52×416=21632 个,共 25 320 个变元.生成方程 652×51+24×52=34500 个.其中,线性方程 16×52+288×51=15520 个,这 15 520 个线性方程可以直接消掉 15 520 个变元.因此,我们最终建立的非线性方程组的变元数为 9 800 个,方程数为 18 980.

Table 1 Parameters of equations about output variables and state update at clock t

表 1 t 时刻输出和状态更新方程的参数

	t 时刻状态及输出方程	t 时刻状态更新方程
引入中间变量数	16	56
线性方程数	16	288
2 次方程数	0	228
5 次方程	8	136
方程总数	24	652

前面我们列出了方程系统,共 9 800 个变元,18 980 个布尔方程,最高代数次数为 5.现在简述求解 Loiss 生成的方程的复杂度.对方程的求解算法,主要有 Gröbner 基算法、XL 算法等.

在运用 Gröbner 基算法解方程组时,通过消去法得到的单变元多项式的次数随着未知变量的个数 n 呈现指数增长.在最坏情况下,Buchberger’s 算法甚至呈现双指数增长,这让 Gröbner 基算法在 n 不是很大时已没有实际

作用.具体说来,在方程个数 $m=n$ 的情况下,如果 $K=GF(2)$,用 Gröbner 基算法解方程组的复杂度为 $O(2^{2^n})$(这个复杂度比穷举搜索的复杂度 2^n 要高).Loiss 生成的方程系统变元数为 9 800,所以不能用此方法求解.

给定 m 个 n 变元的 2 次方程组,运用 XL 算法求解时,进行变元替换后线性独立方程组的个数至少要比变元个数多时才有可能求出解.因此,参数 D 的选取会有一个下界:$D\geqslant \frac{n}{\sqrt{m}}$.具体分析如下:

(1) $m\approx n$ 时,假设最后所得到的线性方程组中大部分是线性独立的,那么 XL 算法攻击成功时有 $D\approx\sqrt{n}$,此时,算法的复杂度下界为 $e^{\omega\sqrt{n}(\ln n/2+1)}$ (其中,当 XL 算法中用到高斯消元法时 $\omega=3$,用一种改进后的算法时,$\omega=2.3766$),此时,XL 的复杂度至少是亚指数;

(2) $m=n$ 时,参数 $D=2^n$,此时,XL 算法的复杂度不低于指数级别;

(3) 当 $m=n+C(C\geqslant 1)$时,文献[7]认为,此时 D 的取值会比较小,因此当 C 比较大时,总的复杂度有取到多项式时间内的可能.但 Diem 等人[8]认为,文献[7]中的估计过于乐观,并得出当 $C\geqslant 1$ 时,有 $D_{\min}\geqslant \frac{n}{\sqrt{C-1}+1}$.因此,用 XL 解方程组时多项式时间的复杂度几乎是不可能存在的.

这样,对于共 9 800 个变元、18 980 个布尔方程组成的 2 次系统,所需要的时间复杂度不小于 $\left(\sum_{i=0}^{D_{\min}}\binom{n}{i}\right)^{\omega}\approx 2^{2420.88}$.而 Loiss 生成的方程系统的方程中有大量的高次方程,因此,用 XL 算法求解这个方程组的复杂度不会低于 $2^{2420.88}$,这超出了现有计算机的计算能力.综上所述,本文没有发现比穷举法更好的代数攻击方法.

4 BOMM 算法的统计性质

4.1 BOMM算法的统计弱点

记 BOMM 算法在 t 时刻的输入为 $x^{(t)}$,BOMM 的输出字节 $y^{(t)}$只由寄存器某内部状态字节和 $x^{(t)}$确定,即 $y^{(t)}=y_h^{(t)}\oplus x^{(t)}$,其中,$y_h^{(t)}=y_{x^{(t)}>>4}^{(t)}$.假设$\{x^{(t)}\}_{t\geqslant 0}$是周期为 k 的序列,即 $x^{(t+k)}=x^{(t)},t\geqslant 0$,现在我们观察 BOMM 算法输出的情况.

以 $k=1$ 的情况举例.此时,输入 $x^{(t)}$为常数保持不变,因此 h,l 也是常数.

不妨设 $x^{(t)}=0x09$,那么 $h=0,l=0,y^{(t)}=0x09\oplus y_0^{(t)}$, $S_2(0x09)=0x72$.BOMM 寄存器单元更新:

$$y_9^{(t+1)}=y_9^{(t)}\oplus S_2(0x09)=y_9^{(t)}\oplus 0x72,$$
$$y_0^{(t+1)}=y_0^{(t)}\oplus S_2(y_9^{(t+1)})=y_0^{(t)}\oplus S_2(y_9^{(t)}\oplus 0x72).$$

易见 $y_9^{(t+2)}=y_9^{(t)},t\geqslant 0$.利用这个关系,计算

$$\begin{aligned}y_0^{(t+4)}&=y_0^{(t+3)}\oplus S_2(y_9^{(t+4)})\\&=y_0^{(t)}\oplus S_2(y_9^{(t+1)})\oplus S_2(y_9^{(t+2)})\oplus S_2(y_9^{(t+3)})\oplus S_2(y_9^{(t+4)})\\&=y_0^{(t)}.\end{aligned}$$

即 $\{y_0^{(t)}\}_{t\geqslant 0}$ 的周期为 4.进一步地,输出序列$\{y^{(t)}\}_{t\geqslant 0}$的周期也是 4.

更一般地,我们有:

定理 1. 假设 BOMM 算法的输入字节序列$\{x^{(t)}\}_{t\geqslant 0}$是周期为 k 的周期序列,那么其输出字节序列$\{y^{(t)}\}_{t\geqslant 0}$有一组线性关系:对任意 $i\geqslant 0$, $z^{(t_0+ik+4k)}=z^{(t_0+ik)}$ 且这组线性关系成立的概率不小于 $\frac{15}{16}+\left(\frac{14}{16}\right)^{2k-2}$.

证明:$k=1$ 的情况上面已有讨论.当 $k>1$ 时,假设时刻 t_0 满足对所有 $t\in\{t_0+1,t_0+2,\ldots,t_0+k-1\}$都有 $h^{(t)}\neq h^{(t_0)}$, $h^{(t)}\neq l^{(t_0)},l^{(t)}\neq h^{(t_0)},l^{(t)}\neq l^{(t_0)}$ 且 $h^{(t_0)}\neq l^{(t_0)}$ 成立.对任意整数 $i\geqslant 0$,记 $T_i=t_0+ik$,由于 $x^{(t+k)}=x^{(t)}$,所以有 $h^{(t+k)}=h^{(t)}$, $l^{(t+k)}=l^{(t)}$,即对所有 $i\geqslant 0$, $x^{(T_i)},h^{(T_i)},l^{(T_i)}$ 的值保持不变,分别记为 x,h,l.由状态更新的关系式可知,序号为 h,l 的记忆单元满足下面关系:

$$y_h^{(T_{i+1})} = y_h^{(T_i)} \oplus S_2(y_l^{(T_{i+1})}),$$
$$y_l^{(T_{i+1})} = y_l^{(T_i)} \oplus S_2(x).$$

易见 $y_l^{(T_{i+2})} = y_l^{(T_i)}$,所以,

$$\begin{aligned} y_h^{(T_{i+4})} &= y_h^{(T_{i+3})} \oplus S_2(y_l^{(T_{i+4})}) \\ &= y_h^{(T_i)} \oplus S_2(y_l^{(T_{i+1})}) \oplus S_2(y_l^{(T_{i+2})}) \oplus S_2(y_l^{(T_{i+3})}) \oplus S_2(y_l^{(T_{i+4})}) \\ &= y_h^{(T_i)}. \end{aligned}$$

进一步对算法输出字节序列$\{y^{(t)}\}_{t\geqslant 0}$有 $y^{(T_{i+4})} = y^{(T_i)}$, 即 $y^{(t_0+ik+4k)} = y^{(t_0+ik)}$, 其中,$i\geqslant 0$.

不难看出,在 $x^{(t)}, t\in\{t_0,t_0+1,t_0+2,\ldots,t_0+k-1\}$独立均匀分布的假设下,对所有 $t\in\{t_0+1,t_0+2,\ldots,t_0+k-1\}$,都有 $h^{(t)}\neq h^{(t_0)}, h^{(t)}\neq l^{(t_0)}, l^{(t)}\neq h^{(t_0)}, l^{(t)}\neq l^{(t_0)}$ 且 $h^{(t_0)}\neq l^{(t_0)}$ 成立的概率为 $\frac{15}{16}+\left(\frac{14}{16}\right)^{2k-2}$.

综上所述,原命题成立. □

4.2 Loiss算法在一类弱密钥下的安全性

由上节分析的BOMM算法输出序列的统计弱点我们发现,Loiss算法可能存在一类弱密钥.记Loiss算法的初始密钥为 *IK*,初始向量为 *IV*,则 Loiss 算法可能存在一类弱(*IK*,*IV*)对[9],使得初始化过程后,算法的 LFSR 的内部状态字节全部为 0.如果同时非线性函数 *F* 的记忆单元 *R* 等于某些特定值(我们称其为低阶 *F*-不动点),那么,该组件的输出序列$\{x^{(t)}\}_{t\geqslant 0}$就会成为周期序列.根据上节的分析结果,此时 Loiss 算法密钥流序列将表现出定理 1 所述的统计性质.本节分析在使用一类特定的弱密钥下,Loiss 算法输出密钥流的统计性质.

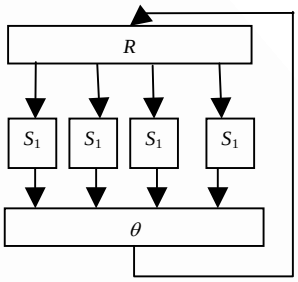


Fig.3 Function $\theta\circ\gamma$
图 3 $\theta\circ\gamma$函数

假定 Loiss 算法在初始化过程后 LFSR 的内部状态字节全部为 0,此时非线性函数 *F* 的状态更新过程退化为 $R^{(t+1)}=\theta(\gamma(R^{(t)}))$,如图 3 所示.其中,

- $\gamma(R_t)=(S_1(R_0^{(t)}),S_1(R_1^{(t)}),S_1(R_2^{(t)}),S_1(R_3^{(t)}))$, $R_i^{(t)}$ 表示 $R^{(t)}$的第 i 个字节, $0\leqslant i\leqslant 3$;
- θ是 $F_2^{32}\to F_2^{32}$ 的线性置换, $\forall x\in F_2^{32}$, $\theta(x)=x\oplus(x<<<2)\oplus(x<<<10)\oplus(x<<<18)\oplus(x<<<24)$.

定义 1. f 是 $F_2^n\to F_2^n$ 的映射,若存在 $a\in F_2^n$ 和某个整数 $k>0$ 使得 $f^k(a)=a$ 成立,那么我们称 a 为 f 的 k 阶不动点.其中,$f^k(\cdot)$表示 f 的 k 次迭代作用.若 $k>0$ 是使得 $f^k(a)=a$ 成立的最小整数,那么我们称 a 为 f 的严格 k 阶不动点.

在 Loiss 算法中,记忆单元 *R* 的更新由θ和γ的复合函数完成,简称复合函数$\theta\circ\gamma$的不动点为 *F*-不动点.通过枚举验证,我们找到了一些低阶 *F*-不动点.表 2 列出所有 1~8 阶严格 *F*-不动点,注意,没有严格 2 阶~5 阶 *F*-不动点.

Table 2 Lower order fixed points of *F*

表 2 低阶 *F*-不动点

严格 1 阶 *F*-不动点有 4 个	0x0924*cf*89, 0x24*cf*8909, 0x*cf*890924, 0x890924*cf*
严格 6 阶 *F*-不动点有 6 个	0x3*a*993*a*99, 0x*cc*54*cc*54, 0x54*cc*54*cc*, 0x*dc*54*dc*54, 0x993*a*993*a*, 0x54*dc*54*dc*
严格 7 阶 *F*-不动点有 28 个	0x002811*e*6, 0x11*e*60028, 0x15333792, 0x01*ac*5*f*469, 0x22537*ca*8, 0x022*ffc*42*c*, 0x2811*e*600 0x2*c*22*ffc*4, 0x33379215, 0x37921533, 0x3*af*65*b*89, 0x537*ca*822, 0x5*b*893*af*6, 0x691*ac*5*f*4 0x6977*d*5*fd*, 0x77*d*5*fd*69, 0x7*ca*82253, 0x893*af*65*b*, 0x92153337, 0x*a*822537*c*, 0x*c*42*c*22*ff* 0x*c*5*f*4691*a*, 0x*d*5*fd*6977, 0x*e*6002811, 0x*f*4691*ac*5, 0x*f*65*b*893*a*, 0x*fd*6977*d*5, 0x*ffc*42*c*22
严格 8 阶 *F*-不动点有 16 个	0xa0a0a0a0, 0x169*e*3*fa*4, 0x21212121, 0x3*f*3*f*3*f*3*f*, 0x3*fa*4169*e*, 0x47474747, 0x6*b*6*b*6*b*6*b*, 0x7*fcffc*94 0x8*e*8*e*8*e*8*e*, 0x947*fcffc*, 0x98989898, 0x9*e*3*fa*416, 0x*a*4169*e*3*f*, 0xc0c0c0c0, 0x*cffc*947*f*, 0x*fc*947*fcf*

假设 Loiss 算法在初始化完成后, LFSR 的所有记忆单元全部为 0,且记忆单元 *R* 的值为某个 *k* 阶 *F*-不动点,在此条件下,Loiss 内部记忆单元有如下表现:

- 现象 1:LFSR 的内部状态字节始终为 0,即$\forall t\geqslant 0, s_i^{(t)}=0$,i=0,1,…,31;
- 现象 2:寄存器 R 的状态周期为 k,记 $R^{(t)}$为寄存器 R 在 t 时刻的值,则有 $R^{(t+k)}=R^{(t)}$;
- 现象 3:记 BOMM 算法在 t 时刻的输入为 $x^{(t)}$,这里,$x^{(t)}$是由 $R^{(t)}$导出的,由 $R^{(t)}$的周期性推出$\{x^{(t)}\}_{t\geqslant 0}$的周期为 k,即 $x^{(t+k)}=x^{(t)}$,$t\geqslant 0$.

根据定理 1,在上面假设成立的前提下,Loiss 算法输出密钥流字节序列$\{z^{(t)}\}_{t\geqslant 0}$有线性关系:对任意 $i\geqslant 0$,$z^{(t_0+ik+4k)}=z^{(t_0+ik)}$. 这组线性关系成立的概率不小于$\frac{15}{16}+\left(\frac{14}{16}\right)^{2k-2}$.

值得注意的是,本节弱密钥假设成立概率很小,几乎为 0.假设 Loiss 算法的初始化过程是一个随机函数,即初始化过程后每个字节单元的取值可以看作是$\{0,1\}^8$上独立均匀分布的随机变量.这样,弱密钥假设成立的概率小于 $1/2^{256}$,而作为初始化输入的(IK,IV)对的数量只有 2^{256},因此满足假设的弱密钥对的数量小于 1.但是这个结论是基于很强的随机性假设之上的,所以我们还不能肯定这样的弱密钥对一定不存在.

5 结 论

本文建立了 BOMM 算法的布尔方程系统,在此基础上讨论了针对 Loiss 算法的代数攻击的复杂度.我们认为,受到现有解方程技术的限制,代数攻击不能在实际中攻破 Loiss 算法.此外,我们还发现了 BOMM 算法的一个统计弱点,提出在某类弱密钥存在的条件下 Loiss 输出密钥流将表现出很强的线性关系.虽然我们认为这类弱密钥存在的概率几乎为 0,但是还不能从理论上给出证明.

References:

[1] Zhang YA, Feng DG. Word-Oriented memorable logics in stream cipher design. Journal of Beijing University of Posts and Telecommunications, 2006,29(2):16−17 (in Chinese with English abstract).

[2] Feng DG, Zhang YA. A sequence disruption algorithm featured in memorable logics wih single-byte operations. Int. Cl.: H04L 9/00 China Patent 200510051171, 2005-03-02 (in Chinese).

[3] Feng DG, Feng XT, Zhang WT, Fan XB, Wu CK. Loiss: A byte-oriented stream cipher. In: Chee YM, ed. Proc. of the IWCC. Berlin, Heidelberg: Springer-Verlag, 2011. 109−125.

[4] Nicolas TC, Willi. M. Algebraic attacks on stream ciphers with linear feedback. In: Biham E, ed. Proc. of the EUROCRYPT 2003. LNCS 2656, Berlin Heidelberg: Springer-Verlag, 2003. 345−359.

[5] Nicolas TC, Josef P. Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng YL, ed. Proc. of the Asiacrypt 2002. LNCS 2501, Berlin, Heidelberg: Springer-Verlag, 2002. 267−287. [doi: 10.1007/3-540-36178-2_17]

[6] Willi M, Enes P, Claude C. Algebraic attacks and decomposition of Boolean functions. In: Cachin C, Camenisch J, eds. Proc. of the EUROCRYPT 2004. LNCS 3027, Berlin: Springer-Verlag, 2004. 474−491. [doi: 10.1007/978-3-540-24676-3_28]

[7] Nicolas C, Alexander K, Jacques P, Shamir A. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel B, ed. Proc. of the EUROCRYPT 2000. LNCS 1807, Berlin: Springer-Verlag, 2000. 392−407.

[8] Claus D. The XL-algorithm and a conjecture from commutative algebra. In: Lee PJ, ed. Proc. of the ASIACRYPT 2004. LNCS 3329, Berlin: Springer-Verlag, 2004. 323−337.

[9] Fluhrer S, Mantin I, Shamir A. Weaknesses in the key scheduling algorithm of RC4. In: Vaudenay S, Youssef A, eds. Proc. of the SAC 2001. LNCS 2259, Berlin: Springer-Verlag, 2001. 1−24.

附中文参考文献:

[1] 张玉安,冯登国.序列密码设计中的整字带记忆逻辑.北京邮电大学学报,2006,29(2):16−17.

[2] 冯登国,张玉安.基于单字节操作、以带记忆为特征的序列扰乱方法. Int. Cl.: H04L 9/00 中国专利 200510051171, 2005-03-02.

附录 1. Loiss 算法中的 *S* 盒

Table 3 *S*-Box S_1

表 3 *S* 盒 S_1

55	C2	63	71	3B	C8	47	86	9F	3C	DA	5B	29	AA	FD	77
8C	C5	94	0C	A6	1A	13	00	E3	A8	16	72	40	F9	F8	42
44	26	68	96	81	D9	45	3E	10	76	C6	A7	8B	39	43	E1
3A	B5	56	2A	C0	6D	B3	05	22	66	BF	DC	0B	FA	62	48
DD	20	11	06	36	C9	C1	CF	F6	27	52	BB	69	F5	D4	87
7F	84	4C	D2	9C	57	A4	BC	4F	9A	DF	FE	D6	8D	7A	EB
2B	53	D8	5C	A1	14	17	FB	23	D5	7D	30	67	73	08	09
EE	B7	70	3F	61	B2	19	8E	4E	E5	4B	93	8F	5D	DB	A9
AD	F1	AE	2E	CB	0D	FC	F4	2D	46	6E	1D	97	E8	D1	E9
4D	37	A5	75	5E	83	9E	AB	82	9D	B9	1C	E0	CD	49	89
01	B6	BD	58	24	A2	5F	38	78	99	15	90	50	B8	95	E4
D0	91	C7	CE	ED	0F	B4	6F	A0	CC	F0	02	4A	79	C3	DE
A3	EF	EA	51	E6	6B	18	EC	1B	2C	80	F7	74	E7	FF	21
5A	6A	54	1E	41	31	92	35	C4	33	07	0A	BA	7E	0E	34
88	B1	98	7C	F3	3D	60	6C	7B	CA	D3	1F	32	65	04	28
64	BE	85	9B	2F	59	8A	D7	B0	25	AC	AF	12	03	E2	F2

Table 4 *S*-Box S_2

表 4 *S* 盒 S_2

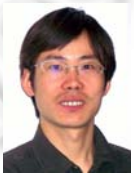
61	97	FF	E9	66	56	F1	F3	54	72	CC	4D	85	52	7A	70
D0	2E	4C	58	BE	88	7F	5A	2F	1B	47	AF	9B	D5	BF	81
C3	4E	86	2D	6A	9C	CE	20	2B	53	6D	FD	3C	BC	33	22
F7	59	C9	63	6E	8D	DD	F2	E3	1A	75	DA	13	1D	68	42
A4	3F	B7	46	90	12	73	EB	FA	F6	09	40	A5	E0	B4	B1
51	8E	06	34	7D	DF	99	6F	AA	0B	80	95	25	EA	87	CD
DC	0C	43	FB	A7	BD	9E	FC	EE	9F	74	B6	CF	EF	16	0F
78	D1	92	64	D6	84	48	41	08	60	5D	2A	B8	4F	E2	69
01	C1	31	5F	62	49	B2	93	00	CB	04	18	07	71	17	E4
AC	8B	B0	7E	F8	44	5B	AD	98	A0	27	4B	3A	B5	F0	83
F9	14	E7	23	77	D2	10	AE	B3	36	30	3B	1C	03	82	38
0E	7B	50	A6	1F	7C	CA	C2	02	2C	A9	8A	39	15	F4	D9
A3	55	32	96	C8	8C	C0	05	67	1E	EC	19	29	89	F5	21
37	BB	E1	57	A2	C7	E6	8F	AB	91	35	28	D3	D7	79	BA
A1	6C	B9	DE	A8	5E	FE	6B	C5	ED	65	9A	45	C6	C4	9D
94	24	0D	0A	E5	76	3D	E8	26	5C	D4	4A	D8	11	DB	3E

注:表格中数字采用 16 进制表示.

杨笑(1982－),男,北京人,博士生,主要研究领域为流密码的分析与设计.

余玉银(1985－),男,博士生,主要研究领域为密码函数及有限域.

范修斌(1966－),男,博士,研究员,博士生导师,主要研究领域为密码学,网络安全,代数学,代数数论,概率论.



冯秀涛(1978－),男,博士,主要研究领域为对称密码算法.

武传坤(1964－),男,博士,研究员,博士生导师,主要研究领域为密码学,信息安全.