软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(5):1111−1126 [doi: 10.3724/SP.J.1001.2013.04266]

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

5 轮 Salsa20 的代数-截断差分攻击*

关 杰, 张中亚

(信息工程大学 电子技术学院,河南 郑州 450004)

通讯作者: 关杰, E-mail: guanjie007@163.com

摘 要: Salsa20 流密码算法是 Estream 最终胜出的 7 个算法之一.结合非线性方程的求解及 Salsa20 的两个 3 轮高概率差分传递链,对 5 轮 Salsa20 算法进行了代数-截断差分攻击.计算复杂度不大于 $O(2^{105})$,数据复杂度为 $O(2^{11})$,存储复杂度为 $O(2^{11})$,成功率为 97.72%.到目前为止,该攻击结果是对 5 轮 Salsa20 算法攻击最好的结果.

关键词: 流密码;Salsa20;截断差分攻击;代数攻击;非线性方程

中图法分类号: TP309 **文献标识码**: A

中文引用格式: 关杰,张中亚.5 轮 Salsa20 的代数-截断差分攻击.软件学报,2013,24(5):1111−1126. http://www.jos.org.cn/1000-9825/4266.htm

英文引用格式: Guan J, Zhang ZY. Algebraic truncated differential cryptanalysis of 5-round Salsa20. Ruan Jian Xue Bao/Journal of Software, 2013,24(5):1111−1126 (in Chinese). http://www.jos.org.cn/1000-9825/4266.htm

Algebraic Truncated Differential Cryptanalysis of 5-Round Salsa20

GUAN Jie, ZHANG Zhong-Ya

(Electronic Technology Institute, Information Engineering University, Zhengzhou 450004, China)

Corresponding author: GUAN Jie, E-mail: guanjie007@163.com

Abstract: Stream cipher salsa20 is one of the seven finally victor algorithms of Estream stream cipher project. An algebraic truncated differential cryptanalysis of 5-round Salsa20 based on solving nonlinear equations and two higher differential characteristics for 3-round Salsa20 is shown, with the computational complexity of $O(2^{105})$, the date complexity of $O(2^{11})$, the space complexity of $O(2^{11})$. It also has a success rate of 97.72%, and holds the best result of analysis of 5-round Salsa20 by now.

Key words: stream cipher; Salsa20; truncated differential cryptanalysis; algebraic attack; nonlinear equation

ECRYPT 是欧洲 FP6 下的 IST 基金支持的一个为期 4 年的项目,是 NESSIE 计划结束后欧洲启动的一个更大的信息安全研究项目.它的一个主要目标就是发展安全、快速的流密码.从 2004 年 2 月 ECRYPT 启动以来,国际上流密码研究得到了极大的发展,产生了流密码的许多新设计思想与设计理念.其中,基于分组密码的结构环节或设计思想来构造流密码成为流密码设计的一个重要方向,最终胜出的一种算法 Salsa20[1]就属于这类算法.它不但采用了轮函数迭代的设计结构,而且利用了 AES 中的行移位和列混合的思想且整个算法只采用模 2^{32} 加、逐位异或和循环左移 3 种运算,进而达到软件的快速实现.

目前,对 Salsa20 算法的安全性分析主要有差分分析[2−7]、滑动攻击[8]、第二原象攻击[9]这几种.其中,差分分析的结果因其有效性更被人关注.

Crowley[2]对 5 轮 Salsa20 算法进行截断差分分析,并将复杂度降到 $O(2^{165})$.

Fischer 等人[3]用差分分析对 6 轮 Salsa20 算法进行密钥恢复攻击,其计算复杂度为 $O(2^{177})$,数据复杂度为 $O(2^{16})$.另外,他们还对 7 轮 Salsa20 进行相关密钥分析.

* 基金项目: 国家自然科学基金(61202491); 全军军事学研究生课题(2010JY0263-149)
收稿时间: 2011-05-06; 定稿时间: 2012-05-29

Tsrnoo 等人[4]对 7 轮 Salsa20 作了差分攻击,计算复杂度为 $O(2^{190})$,还研究了对 8 轮 Salsa20 差分攻击的可能性.

Philippe 等人[5]结合一种新方法(probabilistic neutral bits)对 Salsa20 作了差分分析,对 7 轮和 8 轮 Salsa20 差分攻击的计算复杂度分别为 $O(2^{151})$和 $O(2^{251})$,成功率为 2^{-1}.

李申华等人[6]得到了关于 Salsa20 算法中基本函数的一些差分性质,并给出了对 Salsa20 差分攻击的 3 个结论.另外,李申华[7]也对 Salsa20 进行了差分故障攻击及 6 轮 Salsa20 的相关密钥攻击.

Deike 等人[8]对序列密码算法 Salsa20 进行了滑动攻击,在常数确定的情况下,Salsa20 算法存在 2^{256} 对滑动对,在得到一对滑动对的基础上,恢复密钥的计算复杂度为 $O(2^{34})$.若已知一簇数据中含有滑动对,则得到滑动对的计算复杂度远低于已知数据的数据复杂度,存储复杂度约为已知数据的数据复杂度.

Cesar 等人[9]对 Salsa20 进行了第二原象攻击,对任意轮数的 Salsa20 算法,都能构造出 2^{31} 对碰撞对.

张中亚等人[10]针对 Deike 等人对 Salsa20 的滑动攻击提出了一种改进的 Salsa20 算法,并对改进算法进行了分析,分析结果表明,改进的算法能够有效地抵抗滑动攻击、第二原象攻击和差分攻击.

穆昭薇等人[11]对 Salsa20 进行了广义相关密钥攻击,当初始状态的对角线为常量时,攻击的计算复杂度为 $O(2^{96})$.

本文结合一个带模 2^n 加和逐位异或的非线性方程的求解及 Salsa20 的两个 3 轮高概率截断差分传递链,给出了一个对 5 轮 Salsa20 算法的代数-截断差分攻击,并用实验模拟了这两个高概率差分传递链.利用该攻击方法,恢复 5 轮 Salsa20 的 256 比特初始密钥的计算复杂度不大于 $O(2^{105})$,数据复杂度为 $O(2^{11})$,存储复杂度为 $O(2^{11})$,成功率为 97.72%,到目前为止,该攻击结果是对 5 轮 Salsa20 算法攻击最好的结果.

1 Salsa20 算法简介

Salsa20 算法由 Bernstein[1]设计,面向软件实现.针对对安全性的不同要求,Salsa20 算法的密钥长度有两种规模:128 比特和 256 比特.Salsa20 算法主要以字为单位进行运算,初始输入、输出和内部状态都是 16 个字.

我们记第 i 轮输出的第 j 个字为 m_j^i $(0\leqslant i\leqslant 20, 0\leqslant j\leqslant 15)$;符号"⊞"表示 mod 2^{32} 加法运算,"⋘"表示循环左移运算.

1.1 轮函数

轮函数用 R 表示,16 个字记为 $GF(2^{32})$上的一个 4×4 的矩阵

$$m=\begin{pmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{pmatrix}.$$

轮函数 $R(m)=(Q'^4(m))^T$,其中,

$$Q'(m)=\begin{pmatrix} m_5 & m_6 & m_7 & q_1 \\ m_9 & m_{10} & m_{11} & q_2 \\ m_{13} & m_{14} & m_{15} & q_3 \\ m_1 & m_2 & m_3 & q_0 \end{pmatrix},\quad \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}=Q\begin{pmatrix} m_0 \\ m_4 \\ m_8 \\ m_{12} \end{pmatrix}.$$

当 Q 函数的输入为 $y=(y_0,y_1,y_2,y_3)$,输出为 $z=(z_0,z_1,z_2,z_3)$, 即 $\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix}=Q\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$ 时,Q 函数为

$$z_1=y_1\oplus((y_0\boxplus y_3)\lll 7),$$
$$z_2=y_2\oplus((z_1\boxplus y_0)\lll 9),$$

$$z_3 = y_3 \oplus ((z_2 \boxplus z_1) \lll 13),$$
$$z_0 = y_0 \oplus ((z_3 \boxplus z_2) \lll 18).$$

1.2 加密函数

Salsa20 算法的密钥流生成函数可以表示为

$$\text{Salsa20}_k(v,i) = H\begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ i_0 & i_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix},$$

其中,$H(S)=S\boxplus R^r(S)$,$R(S)$为轮函数,r 表示圈数(此时 $r=20$),$H(S)$为输出密钥流.初始状态中,$k_{0,\ldots,7}$ 为 256 比特密钥(若选择 128 比特的密钥,则 $k_{4,\ldots,7}$ 重复 $k_{0,\ldots,3}$ 的值),$c_{0,\ldots,3}$ 为常数,$v_{0,1}$ 为初始向量,$i_{0,1}$ 为分组标号.

设明文为 M,密文为 C,加密过程为 $C=M\oplus\text{Salsa20}_k(v,i)$,解密过程为 $M=C\oplus\text{Salsa20}_k(v,i)$.

2 模 2^n 加关于逐位异或的差分分布规律

由于 Salsa20 算法的轮函数是由模 2^n 加、逐位异或和循环移位这 3 种运算构成的混合运算,那么要得到 Salsa20 算法轮函数的差分传递链,首先需要给出模 2^n 加关于逐位异或运算的差分分布规律.

设 $x\in Z/(2^n)$且 $x=\sum_{i=0}^{n-1}2^i x_{(i)}$, $x_{(i)}\in\{0,1\}$,记 $x=(x_{(n-1)},\ldots,x_{(0)})$是 x 的二进制表示,称 $x_{(i)}$是 x 的第 i 位,$x_{(i,\ldots,0)}=(x_{(i)},x_{(i-1)},\ldots,x_{(0)})$是 x 的第 0 到第 i 位.本文均假设 x 的实数表示与其二进制表示按上述方式一一对应,因而不加区分地使用.

设 $z=f(x,y)=x\boxplus y$,其中,$x,y,z\in GF(2^n)$,$z^*=f(x^*,y^*)=x^*\boxplus y^*$,$x^*,y^*,z^*\in GF(2^n)$,记:

$$\Delta x=x\oplus x^*,\ \Delta y=y\oplus y^*,\ \Delta z=z\oplus z^*,\ \Delta x_{(i)}=x_{(i)}\oplus x^*_{(i)},\ \Delta y_{(i)}=y_{(i)}\oplus y^*_{(i)},\ \Delta z_{(i)}=z_{(i)}\oplus z^*_{(i)}.$$

定义 1. 对于 $0\leqslant j\leqslant n-2$,令 $c_{(j+1)}=(x_{(j)}\oplus y_{(j)})c_{(j)}\oplus x_{(j)}y_{(j)}$,并约定 $c_0=0$,则我们称 c 为 $z=x\boxplus y$ 的关于模 2^n 加的进位序列.

定义 2. 对于模 2^n 加 $z=f(x,y)=x\boxplus y$,称 $DP_f(\Delta x,\Delta y,\Delta z)=(1/2^{2n})(\#\{(x,y):f(x,y)\oplus f(x\oplus\Delta x,y\oplus\Delta y)=\Delta z\})$为模 2^n 加差分转移概率,简称模 2^n 加差分概率.

定义 3. 模 2^n 加中第 i 比特的差分转移概率为

$$DP_f(\Delta x_{(i)},\Delta y_{(i)},\Delta z_{(i)})=(1/2^{2\times2})(\#\{(x_{(i)},y_{(i)}):f(x_{(i)},y_{(i)})\oplus f(x_{(i)}\oplus\Delta x_{(i)},y_{(i)}\oplus\Delta y_{(i)})=\Delta z_{(i)}\}).$$

文献[4]中已有对模 2^n 加关于逐位异或运算的差分分布规律的一些分析结果.

引理 1[4]. 当 $0\leqslant i\leqslant n-1$ 时,

$$\Delta z_{(i)}=\Delta x_{(i)}\oplus\Delta y_{(i)}\oplus\Delta c_{(i)}.$$

引理 2[4]. 当 $0\leqslant i\leqslant n-2$ 时,$(\Delta x_{(i)},\Delta y_{(i)},\Delta c_{(i)})$和$\Delta z_{(i)}/\text{Pr},\Delta c_{(i+1)}/\text{Pr}$ 的对应关系见表 1.

Table 1 Relation between $(\Delta x_{(i)},\Delta y_{(i)},\Delta c_{(i)})$ and $(\Delta z_{(i)}/\text{Pr},\Delta c_{(i+1)}/\text{Pr})$

表 1 $(\Delta x_{(i)},\Delta y_{(i)},\Delta c_{(i)})$和$(\Delta z_{(i)}/\text{Pr},\Delta c_{(i+1)}/\text{Pr})$的对应关系

$(\Delta x_{(i)},\Delta y_{(i)},\Delta c_{(i)})$	$\Delta z_{(i)}$		$\Delta c_{(i+1)}$	
	$\Delta z_{(i)}$	Pr	$\Delta c_{(i+1)}$	Pr
(0,0,0)	0	1	0	1
(0,0,1)	1	1	0/1	0.5/0.5
(0,1,0)	1	1	0/1	0.5/0.5
(0,1,1)	0	1	0/1	0.5/0.5
(1,0,0)	1	1	0/1	0.5/0.5
(1,0,1)	0	1	0/1	0.5/0.5
(1,1,0)	0	1	0/1	0.5/0.5
(1,1,1)	1	1	1	1

注:当 $i=0$ 时,$\Delta c_{(0)}=0$.

3 一个非线性方程的求解

在密码分析的过程中,我们经常会遇到非线性方程的求解问题.非线性方程的一般解法是穷举未知量.这种求解方法的计算量是指数级的,若是能够找到一种多项式时间级的非线性方程的解法,则能极大地提高攻击效率.

我们在对 5 轮 Salsa20 进行截断差分攻击时,需要求解一个关于未知量 x 的 $GF(2^n)$上的非线性方程:

$$a=[((b\boxplus x)\oplus d)\boxplus e]\oplus[((b'\boxplus x)\oplus d')\boxplus e'] \tag{1}$$

其中,已知量 a,b,d,e,b',d',e'随机.我们利用从低位开始逐比特猜测的方法求 x 值,该方法的计算复杂度期望值为 $n-1$ 次简单计算,x 的个数的期望值为 1.

在公式(1)中,设 $b\boxplus x$ 的进位序列表示为 c,$((b\boxplus x)\oplus d)\boxplus e$ 的进位序列表示为 $\tilde{c}$,$b'\boxplus x$ 的进位序列表示为 c',$((b'\boxplus x)\oplus d')\boxplus e'$的进位序列表示为 $\tilde{c}'$, 易知:

$$c_{(i+1)}=b_{(i)}x_{(i)}\oplus b_{(i)}c_{(i)}\oplus x_{(i)}c_{(i)} \tag{2}$$

$$\tilde{c}_{(i+1)}=(b_{(i)}\oplus x_{(i)}\oplus c_{(i)}\oplus d_{(i)})e_{(i)}\oplus(b_{(i)}\oplus x_{(i)}\oplus c_{(i)}\oplus d_{(i)})\tilde{c}_{(i)}\oplus e_{(i)}\tilde{c}_{(i)} \tag{3}$$

$$c'_{(i+1)}=b'_{(i)}x_{(i)}\oplus b_{(i)}c'_{(i)}\oplus x_{(i)}c'_{(i)} \tag{4}$$

$$\tilde{c}'_{(i+1)}=(b'_{(i)}\oplus x_{(i)}\oplus c'_{(i)}\oplus d'_{(i)})e'_{(i)}\oplus(b'_{(i)}\oplus x_{(i)}\oplus c'_{(i)}\oplus d'_{(i)})\tilde{c}'_{(i)}\oplus e'_{(i)}\tilde{c}'_{(i)} \tag{5}$$

且 $c_{(0)}=\tilde{c}_{(0)}=c'_{(0)}=\tilde{c}'_{(0)}=0$.

通过对公式(1)进行分析可以得到:

$$a_{(0)}=b_{(0)}\oplus d_{(0)}\oplus e_{(0)}\oplus b'_{(0)}\oplus d'_{(0)}\oplus e'_{(0)}.$$

$\forall i$:$0\leqslant i\leqslant n-2$,均有:

$$a_{(i+1)}=b_{(i+1)}\oplus d_{(i+1)}\oplus e_{(i+1)}\oplus b'_{(i+1)}\oplus d'_{(i+1)}\oplus e'_{(i+1)}\oplus c_{(i+1)}\oplus\tilde{c}_{(i+1)}\oplus c'_{(i+1)}\oplus\tilde{c}'_{(i+1)} \tag{6}$$

将公式(2)~公式(5)代入公式(6),可得到一个线性方程:

$$\alpha_{(i)}=\beta_{(i)}x_{(i)} \tag{7}$$

其中,

$$\begin{aligned}\alpha_{(i)}=a_{(i+1)}\oplus b_{(i+1)}\oplus d_{(i+1)}\oplus e_{(i+1)}\oplus b'_{(i+1)}\oplus d'_{(i+1)}\oplus e'_{(i+1)}\oplus b_{(i)}c_{(i)}\oplus b_{(i)}e_{(i)}\oplus c_{(i)}e_{(i)}\oplus d_{(i)}e_{(i)}\oplus b_{(i)}\tilde{c}_{(i)}\oplus\\ c_{(i)}\tilde{c}_{(i)}\oplus d_{(i)}\tilde{c}_{(i)}\oplus e_{(i)}\tilde{c}_{(i)}\oplus b'_{(i)}c'_{(i)}\oplus b'_{(i)}e'_{(i)}\oplus c'_{(i)}e'_{(i)}\oplus d'_{(i)}e'_{(i)}\oplus b'_{(i)}\tilde{c}'_{(i)}\oplus c'_{(i)}\tilde{c}'_{(i)}\oplus d'_{(i)}\tilde{c}'_{(i)}\oplus e'_{(i)}\tilde{c}'_{(i)}\end{aligned} \tag{8}$$

$$\beta_{(i)}=b_{(i)}\oplus c_{(i)}\oplus e_{(i)}\oplus\tilde{c}_{(i)}\oplus b'_{(i)}\oplus c'_{(i)}\oplus e'_{(i)}\oplus\tilde{c}'_{(i)} \tag{9}$$

由上可知,求解方程(1)的方法可化简为求解方程(7).具体算法如下:

算法 1. 求解非线性方程 $a=[((b\boxplus x)\oplus d)\boxplus e]\oplus[((b'\boxplus x)\oplus d')\boxplus e']$.

输入:$GF(2^n)$上的元 a,b,d,e,b',d',e'.

输出:所有满足上述方程的 x.

预处理: $c_{(0)}=\tilde{c}_{(0)}=c'_{(0)}=\tilde{c}'_{(0)}=0$.

Step 1. 计算 $a_{(0)}$和 $b_{(0)}\oplus d_{(0)}\oplus e_{(0)}\oplus b'_{(0)}\oplus d'_{(0)}\oplus e'_{(0)}$ 的值:若相等,则执行 Step 2;若不相等,则算法终止,输出“无解”.

Step 2. for i=0 to i=n–2, do

Step 2.1. 记$\Omega_{i-1}=\{x_{(i-1,\ldots,0)}\}$,约定$\Omega_{-1}=\varnothing$,取一个 $x_{(i-1,\ldots,0)}$的值,按照公式(8)、公式(9)计算$(\alpha_{(i)},\beta_{(i)})$.

Step 2.2. 判断:

- 当$(\alpha_{(i)},\beta_{(i)})=(0,0)$时,$x_{(i)}$=0 或 1,返回 Step 2.1,取下一个 $x_{(i-1,\ldots,0)}$;
- 当$(\alpha_{(i)},\beta_{(i)})=(0,1)$时,$x_{(i)}$=0,返回 Step 2.1,取下一个 $x_{(i-1,\ldots,0)}$;
- 当$(\alpha_{(i)},\beta_{(i)})=(1,1)$时,$x_{(i)}$=1,返回 Step 2.1,取下一个 $x_{(i-1,\ldots,0)}$;
- 当$(\alpha_{(i)},\beta_{(i)})=(1,0)$时,抛弃该 $x_{(i-1,\ldots,0)}$值,返回 Step 2.1,取下一个 $x_{(i-1,\ldots,0)}$.

取遍所有 $x_{(i-1,\ldots,0)}$的值,若都被抛弃,则 $x_{(i)}$无解,算法终止;若 $x_{(i)}$有解,则 i++;

Step 3. 令 $x_{(n-1)}$=0 或 1,输出 x,算法终止.

算法计算复杂度分析:

由上述算法可得 $x_{(i)}$的个数的期望值为

$$\Pr(\alpha_{(i)}=0,\beta_{(i)}=0)\times 2+\Pr(\alpha_{(i)}=0,\beta_{(i)}=1)\times 1+\Pr(\alpha_{(i)}=1,\beta_{(i)}=1)\times 1+\Pr(\alpha_{(i)}=1,\beta_{(i)}=0)\times 0=$$
$$\Pr(\alpha_{(i)}=0,\beta_{(i)}=0)\times 2+\Pr(\alpha_{(i)}=0,\beta_{(i)}=1)+\Pr(\alpha_{(i)}=1,\beta_{(i)}=1).$$

若 a,b,d,e,b',d',e'的值随机,则由公式(8)和公式(9)可知,$\alpha_{(i)}$和$\beta_{(i)}$的值随机,于是有:

$$\Pr(\alpha_{(i)}=0)=\Pr(\alpha_{(i)}=1)=\Pr(\beta_{(i)}=0)=\Pr(\beta_{(i)}=1)=2^{-1}.$$

且因为 a,d,d'随机,那么由于$\beta_{(i)}$的表达式(9)中不含有关于 $a_{(i)},d_{(i)},d'_{(i)}$ 的项,则$\alpha_{(i)}$和$\beta_{(i)}$可被视为相互独立的.于是有:

$$\Pr(\alpha_{(i)}=0,\beta_{(i)}=0)=\Pr(\alpha_{(i)}=0,\beta_{(i)}=1)=\Pr(\alpha_{(i)}=1,\beta_{(i)}=1)=2^{-2},$$

则方程$\alpha_{(i)}=\beta_{(i)}x_{(i)}$解的个数的期望值为

$$\Pr(\alpha_{(i)}=0,\beta_{(i)}=0)\times 2+\Pr(\alpha_{(i)}=0,\beta_{(i)}=1)+\Pr(\alpha_{(i)}=1,\beta_{(i)}=1)=1.$$

当 $a_{(0)},s_{3(0)},d_{(0)},b_{(0)},s'_{3(0)},d'_{(0)},b'_{(0)}$ 随机时,有:

$$\Pr(a_{(0)}=s_{3(0)}\oplus d_{(0)}\oplus b_{(0)}\oplus s'_{3(0)}\oplus d'_{(0)}\oplus b'_{(0)})=2^{-1}.$$

又由于不能确定 $x_{(n-1)}$的值,则取 $x_{(n-1)}$为 0 或 1.因此得到 x 的计算复杂度为 $n-1$ 次简单计算,x 的个数的期望值为 $2^{-1}\times 1\times 2=1$.

4 5 轮 Salsa20 的代数-截断差分攻击

4.1 攻击的基本思想

文献[2]中在对 Salsa20 算法进行截断差分分析的基本思想是:首先得到 $n-2$ 轮差分特征,然后对 n 轮输出逆推两轮,对密钥进行分步穷举攻击,从而对 n 轮 Salsa20 算法进行密钥恢复.

下面我们以得到 Δm_{12}^{n-2} 的差分特征为例,说明如何对 Salsa20 算法进行截断差分分析.本文所分析的 n 轮 Salsa20 算法密钥长度均为 256 比特规模.由于在对序列密码算法分析时一般都假设已知初始 IV 和输出密钥流,而在对分组密码算法分析时一般都假设已知明密文对,本文在对序列密码分析时仍采用明密文对的说法而不与已知初始 IV 和输出密钥流加以区分.

设 $n\geqslant 2$,n 轮 Salsa20 算法的轮函数输出值为 $R^n(m)=(m_0^n,m_1^n,...,m_{15}^n)$, 输出密钥流为 $H(m)=m\boxplus R^n(m)$,则由 Salsa20 算法轮函数 R 可知,第 $n-2$ 轮的输出字 m_{12}^{n-2} 可由第 n 轮输出的 $(m_4^n,m_6^n,m_7^n,m_8^n,m_9^n,m_{10}^n,m_{11}^n,m_{12}^n,m_{13}^n,m_{14}^n,m_{15}^n)$ 得到.

假设在攻击时,可由明密对得到密钥流 $H(m)$的值(一般来说,这是对序列密码攻击的可行性条件),但 m 中初始密钥 $k_{0,...,7}$ 未知,$c_{0,...,3},v_{0,1}$ 和 $i_{0,1}$ 已知,则可以确定 $(m_0^n,m_5^n,m_6^n,m_7^n,m_8^n,m_9^n,m_{10}^n,m_{15}^n)$ 的值,但不能确定 $(m_1^n,m_2^n,m_3^n,m_4^n,m_{11}^n,m_{12}^n,m_{13}^n,m_{14}^n)$ 的值.那么我们只需得到 $(m_4^n,m_{11}^n,m_{12}^n,m_{13}^n,m_{14}^n)$ 即可求出 m_{12}^{n-2} 的值,而 $(m_4^n,m_{11}^n,m_{12}^n,m_{13}^n,m_{14}^n)$ 是由密钥流 $H(m)$和 $k_{3,...,7}$ 得到的.

那么,在得到满足某差分特征输入差的两对明密文时,可猜测 $k_{3,...,7}$ 值得到 Δm_{12}^{n-2}. 在选取足够多的明密对以后,可以根据此差分特征出现的差分转移概率对出现 Δm_{12}^{n-2} 的个数设置一个限门 T,然后猜测 $k_{3,...,7}$ 的值:如果正确,则以一个很高的概率达到这个限门;反之,则达到这个限门的可能性很小.对于达到这个限门而又不正确的 $k_{3,...,7}$,可以通过穷举 $k_{0,...,2}$ 来排除.具体可利用下述算法 2 恢复密钥,所需穷举的密钥量为 $O(2^{160})$.

算法 2[2]. 利用截断差分分析对 Salsa20 算法进行密钥恢复.

输入:N 对满足某差分特征输入差的明密对.

输出:正确密钥值.

Step 1. 猜测密钥 $k_{3,...,7}$ 的一个值.

Step 2. 随机选取一组满足一定输入差的两对明密文,计算 Δm_{12}^{n-2}.

Step 3. 若与差分特征中 Δm_{12}^{n-2} 的值相等,则计数;若不等,则不计数.返回 Step 2.

Step 4. 当计数器中某些密钥的计数达到该限门 T 时,保留该密钥,否则抛弃.返回 Step 1.

Step 5. 对保留的密钥,通过穷举 $k_{0,\ldots,2}$ 来排除错误的密钥,若正确则输出,若错误则抛弃.

当有密钥输出时,算法终止.

我们给出了得到第 n–2 轮每个字的输出差时所需穷举的密钥字,具体见表 2.

Table 2 Required key and computational complexity for obtaining 16 word's difference of n–2 round

表 2 分别得到第 n–2 轮 16 个字的差所需的密钥字及计算量

n–2 轮输出差	所需密钥块	穷举计算量
Δm_0^{n-2}	$k_1, k_2, k_4, k_5, k_6, k_7$	2^{192}
Δm_1^{n-2}	$k_0, k_1, k_2, k_4, k_5, k_6, k_7$	2^{224}
Δm_2^{n-2}	$k_0, k_1, k_2, k_3, k_5, k_6, k_7$	2^{224}
Δm_3^{n-2}	$k_0, k_1, k_2, k_3, k_4, k_6, k_7$	2^{224}
Δm_4^{n-2}	$k_1, k_2, k_3, k_4, k_5, k_6, k_7$	2^{224}
Δm_5^{n-2}	$k_0, k_1, k_2, k_3, k_5, k_6, k_7$	2^{224}
Δm_6^{n-2}	$k_0, k_1, k_2, k_3, k_5, k_6, k_7$	2^{224}
Δm_7^{n-2}	$k_0, k_1, k_2, k_3, k_4, k_6, k_7$	2^{224}
Δm_8^{n-2}	$k_1, k_2, k_3, k_4, k_5, k_6, k_7$	2^{224}
Δm_9^{n-2}	$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7$	2^{256}
Δm_{10}^{n-2}	k_0, k_1, k_2, k_3	2^{128}
Δm_{11}^{n-2}	k_0, k_1, k_2, k_3, k_4	2^{160}
Δm_{12}^{n-2}	k_3, k_4, k_5, k_6, k_7	2^{160}
Δm_{13}^{n-2}	$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7$	2^{256}
Δm_{14}^{n-2}	$k_0, k_1, k_2, k_3, k_5, k_6, k_7$	2^{224}
Δm_{15}^{n-2}	k_3, k_4, k_6, k_7	2^{128}

由于总计算复杂度为所穷举密钥的计算量和选取的明密文对个数的乘积,我们在构造截断差分传递链时,需要综合以下两个方面的因素进行考虑:

(1) 密钥穷举的计算量尽可能地小,这可以保证最终计算复杂度尽可能地小;

(2) 截断差分转移概率尽可能地大,这是差分攻击中的一般性原则,不仅能够确保选取明密文对个数尽可能地小,而且能够降低最终的计算复杂度.

通过对第 1.1 节中 Salsa20 算法轮函数进行分析发现,对于第 n–2 轮的 16 个字的输出差,分成 4 组按下列顺序先后得到:

- $\Delta m_1^{n-2} \to \Delta m_2^{n-2} \to \Delta m_3^{n-2} \to \Delta m_0^{n-2}$;
- $\Delta m_6^{n-2} \to \Delta m_7^{n-2} \to \Delta m_4^{n-2} \to \Delta m_5^{n-2}$;
- $\Delta m_{11}^{n-2} \to \Delta m_8^{n-2} \to \Delta m_9^{n-2} \to \Delta m_{10}^{n-2}$;
- $\Delta m_{12}^{n-2} \to \Delta m_{13}^{n-2} \to \Delta m_{14}^{n-2} \to \Delta m_{15}^{n-2}$.

由上述分析显然可知,对于同等条件的输入差,在上述分析每一行中,得到的每个字的差的概率依次递减,得到第 1 个字的差的概率最大,得到最后一个字的差的概率最小.

那么,我们在对 n 轮 Salsa20 进行截断差分分析时,基于对总计算复杂度影响的两个方面因素的考虑,将综合考虑表 2 中得到第 n–2 轮各个字所需穷举的密钥量,以及得到第 n–2 轮的 16 个字的输出差的先后顺序,进而选取 n–2 轮的差分传递链.

4.2 恢复k_0田$k_{1,}k_{2,3,4}$

4.2.1 高概率差分传递链

依据上述截断差分分析的基本思想,当对 5 轮 Salsa20 进行截断差分分析时,需要基于一个 3 轮差分传递链及 2 轮的逆运算进行密钥恢复.

当轮数 r=5 时,输出密钥流为 S=m田$R^5(m)$,攻击者可知 m 中的 8 个字,其余 8 个未知的字是密钥.当输出密钥流已知时,可以直接得出 $R^5(m)$中的 8 个字,其余 8 个字由于 m 中的 8 个字是密钥而不能得出.如果我们猜测 $k_{0,1,2,3,4}$ 的值,就可以得到 $R^5(m)$中的 13 个字,如下所示,其中,"•"和"?"分别表示已知和未知的字:

$$\begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ ? & ? & ? & \bullet \end{pmatrix}.$$

根据 R^{-1},可以得到 $R^4(m)$的 12 个字:

$$\begin{pmatrix} \bullet & \bullet & \bullet & ? \\ \bullet & \bullet & \bullet & ? \\ \bullet & \bullet & \bullet & ? \\ \bullet & \bullet & \bullet & ? \end{pmatrix}.$$

根据 R^{-1},可以得到 $R^3(m)$的 2 个字:

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & \bullet & \bullet \\ ? & ? & ? & ? \end{pmatrix}.$$

由上述分析可知,当已知输出密钥流及 m 的 8 个非密钥字时,猜测 $k_{0,\ldots,4}$ 的值,就可以利用轮函数的逆求得 m_{10}^3 和 m_{11}^3. 当得到一对输出密钥流及对应的已知输入时,就可以得到 Δm_{10}^3 和 Δm_{11}^3. 若以一个高概率得到一个关于 Δm_{10}^3 或 Δm_{11}^3 的 3 轮差分传递链,就可以检验所猜测的 $k_{0,\ldots,4}$ 正确与否.确定 $k_{0,\ldots,4}$ 的值后,再确定 $k_{5,\ldots,7}$ 的值,计算复杂度远小于穷举 256 比特密钥.

观察 Salsa20 中的轮变换函数可知,某字中任一比特的变化都将会影响到下个状态的至少 3 个字,进而再影响其他字.这样的特点使得攻击者可以选取较低重量且高转移概率的输入差状态,构造出具有较高转移概率的差分传递链.我们结合对影响总计算复杂度的两个因素的分析,进行差分传递链的选择和构造.构造的具体步骤如下:

首先,我们取

$$\Delta m_8^0 = 0x80000000,\ \Delta m_i^0 = 0x00000000\ (0 \leqslant i \leqslant 15, i \neq 8),$$

可以得到一个 3 轮高概率差分传递链:

- 第 1 轮的输入差

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0x80000000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

- 第 1 轮的输出差(概率 p=1)

$$\begin{pmatrix} ? & 0 & 0x80000000 & 0x00001000 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

- 第 2 轮的输出差(概率 $p=2^{-4}$)

$$\begin{pmatrix} ? & ? & ? & ? \\ 0 & 0 & 0 & 0 \\ 0x80000000 & 0x00001000 & 0x40020000 & 0 \\ 0x00001000 & 0x00200000 & 0x02000004 & ? \end{pmatrix}.$$

- 第 3 轮的输出差(概率 $p=2^{-2}$)

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & 0x03000024 \\ ? & ? & ? & ? \end{pmatrix}.$$

上述截断差分传递链的传递概率为 $1\times2^{-4}\times2^{-2}=2^{-6}$,即

$$\Pr[\Delta m_{11}^3 = 0x03000024 \mid \Delta m_8^0 = 0x80000000, \Delta m_i^0 = 0x00000000 (0 \leqslant i \leqslant 15, i \neq 8)] = 2^{-6}.$$

对比文献[2]中的 3 轮截断差分传递链,

$$\Pr[\Delta m_{12}^3 = 0x02002802 \mid \Delta m_9^0 = 0x80000000, \Delta m_i^0 = 0x00000000 (0 \leqslant i \leqslant 15, i \neq 9)] = 2^{-12}.$$

我们所得到的差分传递链的传递概率为其 2^6 倍,在差分分析过程中具有显著的优势.

通过上面的分析,可以猜测 $k_{0,1,2,3,4}$ 的值来求出 Δm_{11}^3 值,并与 $0x03000024$ 对比是否相等,区分正、误密钥.下面将该过程逆转过来,利用 $k_{0,1,3,4}$ 和差分传递链中的 $\Delta m_{11}^3=0x03000024$ 建立一个关于 k_2 的非线性方程,并将该方程线性化,进而对 k_2 进行求解,若能求出 k_2 的值,则说明当前的 $k_{0,1,2,3,4}$ 的值能使所得的 Δm_{11}^3 值和 $0x03000024$ 相等.

在求解的过程中我们发现,k_0 和 k_1 以和的形式 $k_0 \boxplus k_1$ 出现,那么只需猜测 $k_0 \boxplus k_1, k_{3,4}$ 共 3 个字即可,能大大降低穷举密钥的计算复杂度.

4.2.2 建立非线性方程

依据第 4.2.1 节中所述思想,选择穷举 $k_{0,1,3,4}$ 的值,建立方程求解 k_2 的值.即如下所示,其中,"•"和"?"分别表示已知和未知的字,"Λ"表示含有未知量 k_2 及其他已知字的字:

首先,由已知输出密钥流、常数 $c_{0,\ldots,3}$、初始向量 $v_{0,1}$、分组标号 $i_{0,1}$ 和 $k_{0,1,3,4}$ 的值,可知 $R^5(m)$的 16 个字中 12 个字已知、3 字未知、1 个字只含有未知量 k_2 及其他已知字:

$$\begin{pmatrix} \bullet & \bullet & \bullet & \Lambda \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ ? & ? & ? & \bullet \end{pmatrix}.$$

根据 R^{-1} 可知,$R^4(m)$的 16 个字中,8 个字已知、4 字未知、4 个字只含有未知量 k_2 及其他已知字:

$$\begin{pmatrix} \Lambda & \bullet & \bullet & ? \\ \Lambda & \bullet & \bullet & ? \\ \Lambda & \bullet & \bullet & ? \\ \Lambda & \bullet & \bullet & ? \end{pmatrix}.$$

根据 R^{-1} 可知,$R^3(m)$的 16 个字中,14 个字未知、2 个字只含有未知量 k_2 及其他已知字:

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & \Lambda & \Lambda \\ ? & ? & ? & ? \end{pmatrix}.$$

当得到一对输出密钥流及对应的已知输入时,设 $(s_0,s_0'),(s_1,s_1'),\ldots,(s_{15},s_{15}')$ 分别为一对输出密钥流的 16 个字,我们依据上述分析建立方程,通过对算法轮函数的研究,可以得到一个非线性方程:

$$\begin{aligned}(\Delta m_{11}^3 \oplus m_{14}^4 \oplus m_{14}'^4) \ggg 13 = &[[(s_3 \boxminus k_2) \oplus ((s_1 \boxplus s_2 \boxminus (k_0 \boxplus k_1)) \lll 13)] \boxplus m_{13}^4] \oplus \\ &[[(s_3' \boxminus k_2) \oplus ((s_1' \boxplus s_2' \boxminus (k_0 \boxplus k_1)) \lll 13)] \boxplus m_{13}'^4]\end{aligned} \tag{10}$$

若令 $k_2^* = 2^{32} \boxminus k_2$,则有:

$$\begin{aligned}(\Delta m_{11}^3 \oplus m_{14}^4 \oplus m_{14}'^4) \ggg 13 = &[[(s_3 \boxplus k_2^*) \oplus ((s_1 \boxplus s_2 \boxminus (k_0 \boxplus k_1)) \lll 13)] \boxplus m_{13}^4] \oplus \\ &[[(s_3' \boxplus k_2^*) \oplus ((s_1' \boxplus s_2' \boxminus (k_0 \boxplus k_1)) \lll 13)] \boxplus m_{13}'^4]\end{aligned} \tag{11}$$

$m_{14}^4 \oplus m_{14}'^4$ 的表达式中只有 1 个密钥字 k_4, m_{13}^4 和 $m_{13}'^4$ 的表达式中均只有 1 个密钥字 k_3,则若我们猜测 $k_{0,1,3,4}$ 的值, $m_{14}^4 \oplus m_{14}'^4$, m_{13}^4 和 $m_{13}'^4$ 均为已知.

从公式(11)可以直接看出,k_0 和 k_1 以和的形式 $k_0 \boxplus k_1$ 出现,也即,我们在猜测 $k_{0,1,3,4}$ 时,只需猜测 3 个字 $k_0 \boxplus k_1$, $k_{3,4}$ 即可.

由上述分析,当猜测 3 个字 $k_0 \boxplus k_1, k_{3,4}$ 后,公式(11)的结构和公式(1)相同,那么我们可以利用解公式(1)的算法求解公式(11).

与求解公式(1)相同,求解 k_2 的计算复杂度期望值为 31 次简单计算,k_2 的个数的期望值为 1.由于对 k_2 的 2^{32} 次穷举计算降为 31 次简单运算,穷举的计算量会大幅度降低.

4.2.3 恢复密钥

由第 4.2.2 节可知,当得到满足第 4.2.1 节中的差分特征输入差的两对明密对时,可由 $k_0 \boxplus k_1, k_{3,4}$ 的值和差分特征中 Δm_{12}^3 的值得到 k_2.选取足够多的明密对后,可以对出现 $k_0 \boxplus k_1, k_{2,3,4}$ 的情况进行计数,并设置一个计数值 ξ_1,如果 $k_0 \boxplus k_1, k_{2,3,4}$ 的值正确,则其计数会以一个很高的概率超过这个值;相反,超过这个数值的可能性很小.具体利用算法 3 来恢复密钥.

算法 3. 恢复 $k_0 \boxplus k_1, k_{2,3,4}$.

输入:N_1 对满足输入差为 $\Delta m_8^0 = 0x80000000, \Delta m_i^0 = 0x00000000 (0 \leqslant i \leqslant 15, i \neq 8)$ 的明密对.

输出:4 个密钥字 $k_0 \boxplus k_1, k_{2,3,4}$ 的候选值.

Step 1. 猜测密钥 $k_0 \boxplus k_1, k_{3,4}$ 的一个值.

Step 2. 随机选取一组满足上述输入差的两对明密文,通过 Δm_{11}^3=0x03000024 建立方程(11),调用算法 1 计算 k_2 的值.

Step 3. 若得出 k_2 的值,则对 $k_0 \boxplus k_1, k_{2,3,4}$ 计数;若得不出 k_2 的值,则不计数,返回 Step 2.

当明密对穷举结束后,执行 Step 4.

Step 4. 当计数器中某个密钥出现的个数达不到 ξ_1 时抛弃,当达到或超过时则为正确候选密钥,若正确则输出,若错误则抛弃.返回 Step 1;

当密钥 $k_0 \boxplus k_1, k_{3,4}$ 穷举结束后,算法终止.

至此,我们恢复了 $k_0 \boxplus k_1, k_{2,3,4}$ 的值,下面我们来恢复 $k_{0,1,5,6,7}$.

4.3 恢复 $k_{0,1,5,6,7}$

为了恢复 $k_{0,1,5,6,7}$,我们重新构造了另外一条高概率差分传递链,构造的具体步骤如下:

首先,我们取

$$\Delta m_7^0 = 0x80000000,\ \Delta m_i^0 = 0x00000000\ (0 \leqslant i \leqslant 15,\ i \neq 7),$$

则可以得到一个 3 轮高概率差分传递链:

- 第 1 轮的输入差

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0x80000000 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

- 第 1 轮的输出差(概率 p=1)

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0x80000000 & 0x00001000 & ? \end{pmatrix}.$$

- 第 2 轮的输出差(概率 $p=2^{-4}$)

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0x00001000 & 0x40020000 & 0 & 0x80000000 \\ 0x00200000 & 0x02000004 & ? & 0 \\ ? & ? & ? & ? \end{pmatrix}.$$

- 第 3 轮的输出差(概率 $p=2^{-2}$)

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & 0x03000024 & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{pmatrix}.$$

上述截断差分传递链的传递概率为 $1\times2^{-4}\times2^{-2}=2^{-6}$,即

$$\Pr[\Delta m_6^3 = 0x03000024 \mid \Delta m_7^0 = 0x80000000, \Delta m_i^0 = 0x00000000 (0 \leqslant i \leqslant 15, i \neq 7)] = 2^{-6}.$$

由第 4.2 节可知,我们得到了 $k_0 \boxplus k_1, k_{2,3,4}$ 的值.如果我们再猜测 $k_{0,5,6,7}$(由于已知 $k_0 \boxplus k_1$ 的值,则猜测 k_0 或 k_1 是等价的),那么当得到一对输出密钥流及对应的已知输入时,就可以得到 $R^5(m)$中的所有 16 个字,进而求出 Δm_6^3 的值,并与上述差分传递链中 Δm_6^3 的值 0x03000024 对比是否相等,从而区分正、误密钥.

与第 4.2.2 节相同,利用 $k_{0,5,6}$ 和差分传递链中的 Δm_6^3 =0x03000024 建立一个关于 k_7 的非线性方程,并将该方程线性化,进而对 k_7 进行求解,若能求出 k_7 的值,则说明当前的 $k_{0,5,6,7}$ 能使所得的 Δm_6^3 值和 0x03000024 相等.

当得到一对输出密钥流及对应的已知输入时,设 $(s_0, s_0'), (s_1, s_1'), \ldots, (s_{15}, s_{15}')$ 分别为一对输出密钥流的 16 个字,我们依据上述分析建立方程,通过对算法轮函数的研究,则可以得到一个非线性方程:

$$\begin{aligned}(\Delta m_6^3 \oplus m_9^4 \oplus m_9'^4) \ggg 13 = &[[(s_{14} \boxminus k_7) \oplus ((s_{12} \boxplus s_{13} \boxminus (k_5 \boxplus k_6)) \lll 13)] \boxplus m_8^4] \oplus \\ &[[(s_{14}' \boxminus k_7) \oplus ((s_{12}' \boxplus s_{13}' \boxminus (k_5 \boxplus k_6)) \lll 13)] \boxplus m_8'^4]\end{aligned} \tag{12}$$

令 $k_7^* = 2^{32} \boxminus k_7$,则有:

$$\begin{aligned}(\Delta m_6^3 \oplus m_9^4 \oplus m_9'^4) \ggg 13 = &[[(s_{14} \boxplus k_7^*) \oplus ((s_{12} \boxplus s_{13} \boxminus (k_5 \boxplus k_6)) \lll 13)] \boxplus m_8^4] \oplus \\ &[[(s_{14}' \boxplus k_7^*) \oplus ((s_{12}' \boxplus s_{13}' \boxminus (k_5 \boxplus k_6)) \lll 13)] \boxplus m_8'^4]\end{aligned} \tag{13}$$

$m_9^4 \oplus m_9'^4$ 的表达式中只有 1 个密钥字 k_3, m_8^4 和 $m_8'^4$ 的表达式中均只有 3 个密钥字 $k_{0,1,2}$,由于已经得到 $k_0 \boxplus k_1$, $k_{2,3,4}$ 的值,若猜测 k_0 的值, $m_9^4 \oplus m_9'^4$, m_8^4 和 $m_8'^4$ 均为已知.

从公式(13)可以直接看出,k_5 和 k_6 以和的形式 $k_5 \boxplus k_6$ 出现,也即,我们猜测两个字 $k_5 \boxplus k_6$ 和 k_0 以后,公式(13)的结构与公式(1)相同,于是就可以利用求解公式(1)的算法求解公式(13).由于对密钥 k_5 和 k_6 只需猜测 $k_5 \boxplus k_6$,能够大幅度降低穷举的计算量.

与求解公式(1)相同,求解 k_7 的计算复杂度期望值为 31 次简单计算,k_7 的个数的期望值为 1.由于对 k_7 的 2^{32} 次穷举计算降为 31 次简单运算,穷举的计算量会大幅度降低.

选取足够多的明密对后,可以对出现 $k_5 \boxplus k_6, k_{0,7}$ 的情况进行计数,并设置一个计数值 ξ_2,如果 $k_5 \boxplus k_6, k_{0,7}$ 的值正确,则其计数会以一个很高的概率超过这个值;反之,超过这个数值的可能性则很小.具体利用算法 4 恢复密钥.

算法 4. 恢复 $k_5 \boxplus k_6, k_{0,7}$.

输入:N_2 对满足输入差为 $\Delta m_7^0 = 0x80000000, \Delta m_i^0 = 0x00000000 (0 \leqslant i \leqslant 15, i \neq 7)$ 的明密对.

输出:3 个密钥字 $k_5 \boxplus k_6, k_{0,7}$ 的候选值.

Step 1. 猜测密钥 k_5⊞k_6,k_0 的一个值.

Step 2. 随机选取一组满足上述输入差的两对明密文,通过 Δm_6^3=0x03000024 建立方程(13),调用算法 1 计算 k_7 的值.

Step 3. 若得出 k_7 的值,则对 k_5⊞k_6,$k_{0,7}$ 计数;若得不出 k_7 的值,则不计数,返回 Step 2.

当明密对穷举结束后,执行 Step 4.

Step 4. 计数器中某个密钥出现的个数达不到ξ_2时则抛弃,达到或超过时则为正确候选密钥,正确则输出,错误则抛弃.返回 Step 1.

密钥 k_5⊞k_6,k_0 穷举结束后,算法终止.

至此,我们得到了 k_0,k_1,k_2,k_3,k_4,k_5⊞k_6,k_7 的候选值.最后,可以穷举 k_5 或 k_6 中任意 1 个字,得到全部 256 比特初始密钥.

下面我们对整个密钥恢复过程进行复杂度和成功率分析.

4.4 复杂度和成功率分析

在恢复 k_0⊞k_1,$k_{2,3,4}$ 时,我们称满足 Δm_{11}^3=0x03000024 的输入对为正确对,否则为错误对.由于正确密钥碰到正确对时被计数一次,碰到错误对时该密钥被计数的概率为 2^{-32},错误密钥被计数的概率也为 2^{-32},那么可知,在密钥恢复算法中,正确密钥被计数的次数大于或等于正确对的个数.根据差分传递链的转移概率选取合适的ξ_1和 N_1,可以保证算法 3 的成功率接近 1.

我们通过实验对差分传递概率进行了模拟,由于存在多路径差分,通过模拟实验得到的差分传递概率约为 2^{-5},大于理论差分传递概率值 2^{-6}.我们选取 2^{20} 组数据对正确对出现的个数 T_1 进行模拟实验,设定δ_1=1,…,6,分别统计了 T_1 大于等于δ_1 的概率 $p(T_1 \geqslant \delta_1)$.实验结果见表 3.

Table 3 Probability of $T_1 \geqslant \delta_1$ when the number of data pair is N_1

表 3 数据对个数取 N_1 时 $T_1 \geqslant \delta_1$ 的概率

δ_1	N_1				
	2^6 (%)	2^7 (%)	2^8 (%)	2^9 (%)	2^{10} (%)
1	79.18	90.93	96.91	99.77	99.995
2	55.81	79.28	92.63	99.06	99.975
3	34.91	66.03	87.57	97.78	99.913
4	19.50	52.44	81.68	95.97	99.758
5	9.92	39.50	75.00	93.72	99.459
6	4.60	28.14	67.71	91.16	98.985

由表 3 可以看出,概率 $p(T_1 \geqslant \delta_1)$随着$\delta_1$ 的递增而递减,随着 N_1 的递增而递增,图 1 给出了 $p(T_1 \geqslant \delta_1)$与$\delta_1$,$N_1$ 的关系.

由于正确密钥碰到正确对时一定被计数一次,碰到错误对时被计数的概率为随机,错误密钥被计数的概率也为随机,则通过检测的密钥 k_0⊞k_1,$k_{2,3,4}$ 的个数为

$$\#[(k_0 \boxplus k_1, k_{2,3,4})_{\text{pass}}] = 1 + N_1 \times 2^{128-32\times\xi_1}.$$

随着ξ_1 的增大,通过检测的密钥个数逐渐减少,直至为 1.当 $N_1 \leqslant 2^{32}$ 时,图 2 模拟了通过检测的密钥 k_0⊞k_1,$k_{2,3,4}$ 的个数.

穷举 k_0⊞k_1,$k_{3,4}$ 的计算复杂度为 $2^{32\times3}=2^{96}$.执行算法 1 计算 k_2 的计算量为 31 次简单运算,相对于一次算法加密可忽略不计,则算法 3 的计算复杂度为

$$Complexity_3 = 2^{96} \times N_1.$$

由于表 3 和图 1 刻画了 N_1 对满足一定输入差的明密对中含有大于等于δ_1 个正确对的概率,因此它也就是算法 3 取(N_1,ξ_1=δ_1)的成功率,记为 $P_{(N_1,\xi_1=\delta_1)}$.

在恢复 k_5⊞k_6,$k_{0,7}$ 时,我们称满足 Δm_6^3=0x03000024 的输入对为正确对,否则为错误对.与恢复 k_0⊞k_1,$k_{2,3,4}$ 时相同,由于正确密钥碰到正确对时被计数一次,碰到错误对时该密钥被计数的概率为 2^{-32},错误密钥被计数的概

率也为 2^{-32},由此可知,在攻击算法中正确密钥被计数的次数大于或等于正确对的个数.根据差分传递链的转移概率选取合适的ξ_2和 N_2,可以保证算法 4 的成功率接近 1.

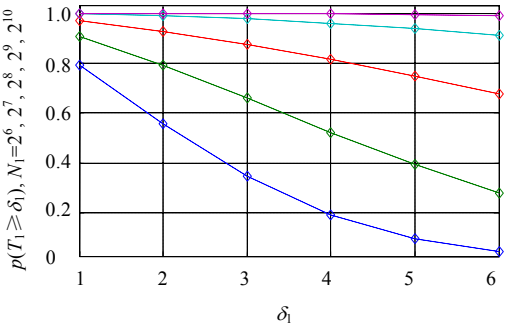

Fig.1 Relation between $p(T_1 \geqslant \delta_1)$ and (δ_1, N_1)

图 1 $p(T_1 \geqslant \delta_1)$与$(\delta_1, N_1)$的关系

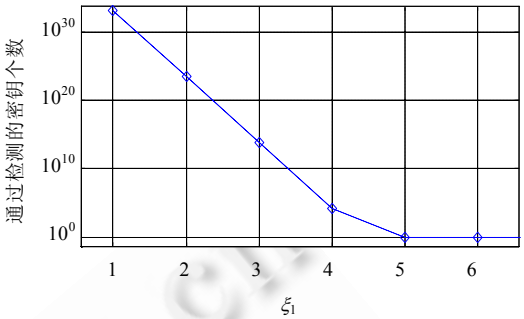

Fig.2 Number of passed candidate key $(k_0 \boxplus k_1, k_{2,3,4})$

图 2 通过检测的候选密钥 $k_0 \boxplus k_1, k_{2,3,4}$ 的个数

同样地,由于存在多路径差分,通过模拟实验得到的差分传递概率约为 2^{-5},大于理论差分传递概率值 2^{-6}.我们仍选取 2^{20} 组数据对正确对的个数 T_2 进行模拟实验,设定δ_2=1,…,6,分别统计了 T_2 大于等于δ_2 的概率 $p(T_2 \geqslant \delta_2)$.实验结果见表 4.

Table 4 Probability of $T_2 \geqslant \delta_2$ when the number of data pair is N_2

表 4 数据对个数取 N_2 时 $T_2 \geqslant \delta_2$ 的概率

δ_2	N_2				
	2^6 (%)	2^7 (%)	2^8 (%)	2^9 (%)	2^{10} (%)
1	83.18	96.72	99.84	99.999	100
2	57.56	87.11	98.96	99.99	100
3	33.75	71.85	96.39	99.94	100
4	17.05	54.22	91.33	99.77	100
5	7.50	37.77	83.64	99.33	100
6	2.91	24.37	73.77	98.44	99.999

由表 4 可以看出,概率 $p(T_2 \geqslant \delta_2)$随着$\delta_2$ 的递增而递减,随着 N_2 的递增而递增,图 3 给出了 $p(T_2 \geqslant \delta_2)$与$\delta_2, N_2$的关系.

由于正确密钥碰到正确对时一定被计数一次,碰到错误对时被计数的概率为随机,错误密钥被计数的概率也为随机,则通过检测的密钥 $k_5 \boxplus k_6, k_{0,7}$ 的个数为

$$\#[(k_5 \boxplus k_6, k_{0,7})_{\text{pass}}]=1+N_2 \times 2^{96-32\times\xi_2}.$$

随着ξ_2的增大,通过检测的密钥逐渐减少,直至为 1.当 $N_2 \leqslant 2^{32}$ 时,图 4 模拟了通过检测的候选密钥 $k_5 \boxplus k_6, k_{0,7}$ 的个数.

穷举 $k_5 \boxplus k_6, k_0$ 的计算复杂度为 $2^{32\times2}=2^{64}$.计算 k_7 的计算量为 31 次简单运算,相对于一次算法加密可忽略不计,则算法 4 的计算复杂度为

$$Complexity_4=2^{64}\times N_2.$$

由于表 4 和图 3 刻画了 N_2 对满足一定输入差的明密对中含有大于等于δ_2 个正确对的概率,因此它也就是算法 4 取$(N_2, \xi_2=\delta_2)$的成功率,记为 $P_{(N_2, \xi_2=\delta_2)}$.

下面我们综合算法 2 和算法 3 分析整个密钥恢复过程的复杂性及成功率.

由上述分析可知,当 $k_0 \boxplus k_1, k_{2,3,4}$ 的值不是唯一确定时,通过检测的 $k_0 \boxplus k_1, k_{2,3,4}$ 候选密钥个数为

$$1+N_1 \times 2^{128-32\times\xi_1}.$$

由于在恢复 $k_5 \boxplus k_6, k_{0,7}$ 过程中 k_4 没有参与运算,因此 $k_{0,1,2,3,4,7}, k_5 \boxplus k_6$ 的候选密钥个数必不大于两个阶段候选值的乘积,即

$$\#[(k_{0,1,2,3,4,7}, k_5 \boxplus k_6)_{\text{pass}}] \leqslant (1+N_1\times 2^{128-32\times\xi_1})\times(1+N_2\times 2^{96-32\times\xi_2})$$
$$= 1+N_1\times 2^{128-32\times\xi_1}+N_2\times 2^{96-32\times\xi_2}+N_1\times N_2\times 2^{224-32\times(\xi_2+\xi_1)}.$$

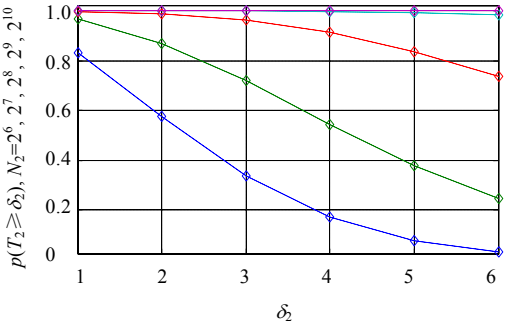

Fig.3 Relation between $p(T_2\geqslant\delta_2)$ and (δ_2,N_2)

图 3 $p(T_2\geqslant\delta_2)$与(δ_2,N_2)的关系

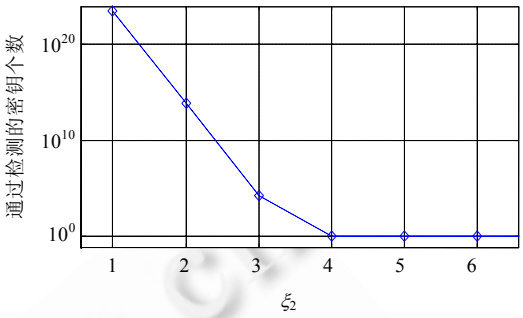

Fig.4 Number of passed candidate key $(k_5\boxplus k_6, k_{0,7})$

图 4 通过检测的候选密钥 $k_5\boxplus k_6, k_{0,7}$ 的个数

穷举 k_5, k_6 中任意一个字的计算量为 2^{32},故穷举的计算复杂度为

$$Complexity_{search} \leqslant (1+N_1\times 2^{128-32\times\xi_1}+N_2\times 2^{96-32\times\xi_2}+N_1\times N_2\times 2^{224-32\times(\xi_2+\xi_1)})\times 2^{32}$$
$$= 2^{32}+N_1\times 2^{160-32\times\xi_1}+N_2\times 2^{128-32\times\xi_2}+N_1\times N_2\times 2^{256-32\times(\xi_2+\xi_1)}.$$

总计算复杂度为

$$Complexity = Complexity_3 + Complexity_4 + Complexity_{search} \leqslant 2^{32}+N_1\times 2^{96}+N_2\times 2^{64}+N_1\times 2^{160-32\times\xi_1}+ N_2\times 2^{128-32\times\xi_2}+N_1\times N_2\times 2^{256-32\times(\xi_2+\xi_1)} \quad (14)$$

总数据复杂度为 $2\times(N_1+N_2)$个明密文对.

总存储复杂度:为了使密钥恢复过程中间不进行数据存储,我们可以在具体密钥恢复过程中将算法 3、算法 4 的运算过程结合起来:当算法 3 得到一个 $k_0\boxplus k_1, k_{2,3,4}$ 的候选值时即执行算法 4;当执行算法 4 得到 $k_0, k_1, k_2, k_3, k_4, k_5\boxplus k_6, k_7$ 一个候选值后,即穷举 k_5 或 k_6 中任意 1 个字,进而求出 256 比特初始密钥.因此,总存储复杂度为 $2\times(N_1+N_2)$个明密文对.

整个密钥恢复算法的成功率为 $P_{success}=P_{(N_1,\xi_1=\delta_1)}\times P_{(N_2,\xi_2=\delta_2)}$.

由上述复杂度分析可知,对于不同的(N_1,ξ_1,N_2,ξ_2),有不同的计算、数据、存储复杂度和成功率.由于公式(14)含有 $N_1\times 2^{96}$ 项,若使计算复杂度尽可能地小,则必须有以下 4 个算式同时成立:

$$N_2\times 2^{64}\leqslant N_1\times 2^{96} \quad (15)$$
$$N_1\times 2^{160-32\times\xi_1}\leqslant N_1\times 2^{96} \quad (16)$$
$$N_2\times 2^{128-32\times\xi_2}\leqslant N_1\times 2^{96} \quad (17)$$
$$N_1\times N_2\times 2^{256-32\times(\xi_2+\xi_1)}\leqslant N_1\times 2^{96} \quad (18)$$

由于选取的数据 N_1, N_2 均为 $2^6, 2^7, 2^8, 2^9, 2^{10}$ 这 5 个值中的一个,公式(15)显然成立.

若使公式(16)成立,只需$\xi_1\geqslant 2$ 即可.

若使公式(17)成立,则有两种可能:

(1) $2^{128-32\times\xi_2}=2^{96}$,且 $N_2\leqslant N_1$,此时,取$\xi_2=1$ 即可;

(2) $2^{128-32\times\xi_2}<2^{96}$,只需$\xi_2>1$ 即可.

若使公式(18)成立,则只需 $2^{256-32\times(\xi_2+\xi_1)}\leqslant 2^{64}$,此时,取$\xi_1+\xi_2\geqslant 6$ 即可.

当式(15)~式(18)均成立时,由表 3 和表 4 可知,为使成功率尽可能地大,需要ξ_1 和ξ_2 值尽可能地小.

综合上述条件以及表 3、表 4,我们得到符合要求的不同(N_1,ξ_1,N_2,ξ_2)对应的不同的复杂度和成功率,见表 5(存储复杂度和数据复杂度相同).

Table 5 Computational complexity and success rate for different (N_1,ξ_1,N_2,ξ_2) up to the mustard

表 5 符合要求的不同(N_1,ξ_1,N_2,ξ_2)对应的复杂度和成功率

N_1	ξ_1	N_2	ξ_2	计算复杂度	数据复杂度	成功率(%)
2^8	2	2^9	4	$\leqslant O(2^{105})$	$2^{10}+2^9$	92.42
2^8	2	2^{10}	4	$\leqslant O(2^{105})$	$2^{11}+2^9$	92.63
2^9	2	2^9	4	$\leqslant O(2^{106})$	2^{11}	98.83
2^9	2	2^{10}	4	$\leqslant O(2^{106})$	$2^{11}+2^{10}$	99.06
2^9	**3**	**2^9**	**3**	**$\leqslant O(2^{105})$**	**2^{11}**	**97.72**
2^9	3	2^{10}	3	$\leqslant O(2^{105})$	$2^{11}+2^{10}$	97.78
2^{10}	2	2^9	4	$\leqslant O(2^{107})$	$2^{11}+2^{10}$	99.75
2^{10}	2	2^{10}	4	$\leqslant O(2^{107})$	2^{12}	99.98
2^{10}	3	2^9	3	$\leqslant O(2^{106})$	$2^{11}+2^{10}$	99.86
2^{10}	3	2^{10}	3	$\leqslant O(2^{106})$	2^{12}	99.91
2^{10}	4	2^8	2	$\leqslant O(2^{106})$	$2^{11}+2^9$	98.72
2^{10}	4	2^9	2	$\leqslant O(2^{106})$	$2^{11}+2^{10}$	99.75
2^{10}	4	2^{10}	2	$\leqslant O(2^{106})$	2^{12}	99.76
2^{10}	5	2^7	1	$\leqslant O(2^{106})$	$2^{11}+2^8$	96.20
2^{10}	5	2^8	1	$\leqslant O(2^{106})$	$2^{11}+2^9$	99.30
2^{10}	5	2^9	1	$\leqslant O(2^{106})$	$2^{11}+2^{10}$	99.46
2^{10}	5	2^{10}	1	$\leqslant O(2^{107})$	2^{12}	99.46

我们可以根据所需成功率的不同选择不同一组的(N_1,ξ_1,N_2,ξ_2)值.为了确保整个密钥恢复算法尽可能小的复杂度及较高的成功率,权衡分析表5中各项数据,我们取$(N_1,\xi_1)=(2^9,3)$,$(N_2,\xi_2)=(2^9,3)$,整个攻击的复杂度和成功率如下:

- 计算复杂度:不大于 $O(2^{105})$;
- 数据复杂度:$O(2^{11})$;
- 存储复杂度:$O(2^{11})$;
- 成功率:97.72%.

5 结束语

本文给出了一种带模 2^n 加和逐位异或的非线性方程的逐比特求解算法,将解该非线性方程的计算复杂度由 $O(2^n)$降为 $n-1$ 次简单运算.得到了两个 Salsa20 的 3 轮高概率差分传递链,差分传递概率均为 2^{-6},并经过编程对差分传递链进行了模拟实验.对比文献[2]中的截断差分传递链,差分传递概率为其2^6倍,并分别利用这两个差分传递链,结合带模 2^n 加和逐位异或的非线性方程的求解,采用分步求解密钥的方法对 5 轮 Salsa20 进行了代数-截断差分攻击.计算复杂度不大于 $O(2^{105})$,数据复杂度为 $O(2^{11})$,存储复杂度为 $O(2^{11})$,成功率为 97.72%.到目前为止,该攻击结果是对 5 轮 Salsa20 最好的攻击结果.

本文中的逐比特求解的方法能够广泛应用于各类带模 2^n 加和逐位异或的方程的求解上,可降低求解此类非线性方程的计算复杂度.如何将此方法推广,对更多轮的 Salsa20 算法进行差分分析,以及与其他密码分析技术相结合,对更多算法进行分析,值得我们做进一步的研究.

References:

[1] Bernstein DJ. Salsa20 specification. 2005. http://cr.yp.to/snuffle/spec.pdf

[2] Crowley P. Truncated differential cryptanalysis of five rounds of Salsa20. In: Workshop Record of SASC 2006: The State of the Art of Stream Ciphers. 2006. http://www.ecrypt.eu.org/stream/papers.html

[3] Fischer S, Meier W, Berbain C, Biasse JF, Robshaw MJB. Non-Randomness in eSTREAM Candidates Salsa20 and TSC-4. In: Barua R, Lange T, eds. Proc. of the Progress in Cryptology—INDOCRYPT 2006. LNCS 4329, 2006. 2−16. [doi: 10.1007/11941378_2]

[4] Tsunoo Y, Saito T, Kubo H, Suzaki T, Nakashima H. Differential cryptanalysis of Salsa20/8. In: Workshop Record of SASC 2007: The State of the Art of Stream Ciphers. 2007. http://www.ecrypt.eu.org/stream/papers.html

[5] Aumasson JP, Fischer S, Khazaei S, Meier W, Rechberger C. New features of Latin dances: Analysis of Salsa, ChaCha, and Rumba. In: Nyberg K, ed. Proc. of the Fast Software Encryption 2008. LNCS 5086, 2008. 470−488. [doi: 10.1007/978-3-540-71039-4_30]

[6] Li SH, Zheng SH, Song CY. Research on differential of Salsa20. Computer Engineering and Applications, 2008,44(1):5−7 (in Chinese with English abstract).

[7] Li SH. Cryptanalysis of two symmetric encryption algorithms ARIA and Salsa20 [Ph.D. Thesis]. Ji'nan: Shandong University, 2008 (in Chinese).

[8] Deike PS, Biryukov A. Slid Pairs in Salsa20 and Trivium. In: Chowdhury DR, Rijmen V, Das A, eds. Proc. of the Progress in Cryptology—INDOCRYPT 2008. LNCS 5365, 2008. 1−14.

[9] Julio CHC, Tapiador JME, Quisquater JJ. On the Salsa20 core function. In: Nyberg K, ed. Proc. of the Fast Software Encryption, 15th Int'l Workshop (FSE 2008). LNCS 5086, 2008. 462−469. [doi: 10.1007/978-3-540-71039-4_29]

[10] Zhang ZY, Guan J, Ding L. An improved Salsa20 stream cipher. Acta Scientiarum Naturalium Universitatis Pekinensis, 2011,47(2): 201−207 (in Chinese with English abstract).

[11] Mu ZW. Security research of the stream cipher Salsa20 [MS. Thesis]. Xi'an: Xi'an University of Electronic Science and Technology, 2011 (in Chinese).

附中文参考文献:

[6] 李申华,郑世慧,宋春燕.流密码 Salsa20 的差分研究.计算机工程与应用,2008,44(1):5−7.

[7] 李申华.对称密码算法 AIRA 和 SALSA20 的安全性分析[博士学位论文].济南:山东大学,2008.

[10] 张中亚,关杰,丁林.一个改进的 Salsa20 流密码算法.北京大学学报(自然科学版),2011,47(2):201−207.

[11] 穆昭薇.流密码算法 Salsa20 的安全性研究[硕士学位论文].西安:西安电子科技大学,2011.

附录 1. MPL 的 BNF 表示

由于 $\mathrm{Salsa20}_k(v,i)=H\begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ i_0 & i_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}$,这里,$H(S)=S\boxplus R^r(S)$,易推得:

$$m_3^5=s_3\boxminus k_2;m_1^5=s_1\boxminus k_0;m_2^5=s_2\boxminus k_1.$$

根据轮函数 R 定义,第 n 轮轮函数的逆可以表示为

- $\begin{cases} m_0^{n-1}=m_0^n\oplus((m_2^n\boxplus m_3^n)\lll 18) \\ m_{12}^{n-1}=m_3^n\oplus((m_1^n\boxplus m_2^n)\lll 13) \\ m_8^{n-1}=m_2^n\oplus((m_1^n\boxplus m_0^{n-1})\lll 9) \\ m_4^{n-1}=m_1^n\oplus((m_0^{n-1}\boxplus m_{12}^{n-1})\lll 7) \end{cases}$;
- $\begin{cases} m_5^{n-1}=m_5^n\oplus((m_4^n\boxplus m_7^n)\lll 18) \\ m_1^{n-1}=m_4^n\oplus((m_6^n\boxplus m_7^n)\lll 13) \\ m_{13}^{n-1}=m_7^n\oplus((m_6^n\boxplus m_5^{n-1})\lll 9) \\ m_9^{n-1}=m_6^n\oplus((m_5^{n-1}\boxplus m_1^{n-1})\lll 7) \end{cases}$;
- $\begin{cases} m_{10}^{n-1}=m_{10}^n\oplus((m_8^n\boxplus m_9^n)\lll 18) \\ m_6^{n-1}=m_9^n\oplus((m_{11}^n\boxplus m_8^n)\lll 13) \\ m_2^{n-1}=m_8^n\oplus((m_{11}^n\boxplus m_{10}^{n-1})\lll 9) \\ m_{14}^{n-1}=m_{11}^n\oplus((m_{10}^{n-1}\boxplus m_6^{n-1})\lll 7) \end{cases}$;

- $\begin{cases} m_{15}^{n-1} = m_{15}^{n} \oplus ((m_{13}^{n} \boxplus m_{14}^{n}) \lll 18) \\ m_{11}^{n-1} = m_{14}^{n} \oplus ((m_{12}^{n} \boxplus m_{13}^{n}) \lll 13) \\ m_{7}^{n-1} = m_{13}^{n} \oplus ((m_{12}^{n} \boxplus m_{15}^{n-1}) \lll 9) \\ m_{3}^{n-1} = m_{12}^{n} \oplus ((m_{15}^{n-1} \boxplus m_{11}^{n-1}) \lll 7) \end{cases}$.

由上述等式组可得:

$$m_{11}^{3} \oplus m_{14}^{4} = (m_{12}^{4} \boxplus m_{13}^{4}) \lll 13 \tag{19}$$

$$m_{12}^{4} = m_{3}^{5} \oplus ((m_{1}^{5} \boxplus m_{2}^{5}) \lll 13) \tag{20}$$

将 $m_{12}^{4}, m_{3}^{5}, m_{1}^{5}, m_{2}^{5}$ 代入公式(19)可得:

$$(m_{11}^{3} \oplus m_{14}^{4}) \ggg 13 = [(s_3 \boxminus k_2) \oplus ((s_1 \boxplus s_2 \boxminus (k_0 \boxplus k_1)) \lll 13)] \boxplus m_{13}^{4}.$$

故公式(10)

$$(\Delta m_{11}^{3} \oplus m_{14}^{4} \oplus m_{14}'^{4}) \ggg 13 = [[(s_3 \boxminus k_2) \oplus ((s_1 \boxplus s_2 \boxminus (k_0 \boxplus k_1)) \lll 13)] \boxplus m_{13}^{4}] \oplus$$
$$[[(s_3' \boxminus k_2) \oplus ((s_1' \boxplus s_2' \boxminus (k_0 \boxplus k_1)) \lll 13)] \boxplus m_{13}'^{4}].$$

得证.

同理可证得公式(12).

关杰(1974－),女,河南郑州人,博士,副教授,主要研究领域为密码学,信息安全.
E-mail: guanjie007@163.com

张中亚(1985－),男,硕士,主要研究领域为密码学,信息安全.
E-mail: mickia10@163.com