软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(9):2861−2873 [doi: 10.13328/j.cnki.jos.005282]

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

故障模型下 MORUS 算法的差分扩散性质研究*

施泰荣, 关 杰, 李俊志, 王森鹏

(信息工程大学,河南 郑州 450001)

通讯作者: 施泰荣, E-mail: strwanzi@163.com

摘 要: MORUS 算法是由 H. Wu 等人设计的一类认证加密算法,目前已顺利进入 CAESAR 竞赛第 3 轮竞选.研究 MORUS 算法故障模型下的差分扩散性质.采用面向比特的随机故障模型,结合差分分析技术与中间相遇思想,改进了针对 MORUS 算法的差分链搜索算法.运用该算法找到了 5 步概率为 2^{-85} 的差分链,从而实现了对初始化过程 5 步的简化版 MORUS-640-128 算法的差分-区分攻击,攻击所需的数据量和区分优势分别为 2^{89} 和 0.99965.最后,利用差分故障分析方法对认证过程 3 步的简化版 MORUS-640-128 算法进行了伪造攻击.

关键词: CAESAR 竞赛;MORUS 算法;差分故障分析;中间相遇

中图法分类号: TP309

中文引用格式: 施泰荣,关杰,李俊志,王森鹏.故障模型下 MORUS 算法的差分扩散性质研究.软件学报,2018,29(9):2861−2873. http://www.jos.org.cn/1000-9825/5282.htm

英文引用格式: Shi TR, Guan J, Li JZ, Wang SP. Research on differential diffusion property of MORUS in fault model. Ruan Jian Xue Bao/Journal of Software, 2018,29(9):2861−2873 (in Chinese). http://www.jos.org.cn/1000-9825/5282.htm

Research on Differential Diffusion Property of MORUS in Fault Model

SHI Tai-Rong, GUAN Jie, LI Jun-Zhi, WANG Sen-Peng

(Information Engineering University, Zhengzhou 450001, China)

Abstract: MORUS is a third-round CAESAR candidate of authenticated cipher designed by H. Wu *et al*. With a fault model, the diffusion property of MORUS is analyzed in this paper. By using a bit-oriented random fault model, the search algorithm for the differential chain of MORUS is improved with the usage of differential analysis and meet-in-the-middle technique. Through this algorithm, a 5-step differential chain is discovered with a probability of 2^{-85}. The differential-distinguish attack on the initialization of 5-step reduced version of MORUS-640-128 is proposed with the data complexity of 2^{89} and the distinguishing advantage of 0.99965. By using differential fault analysis method, the forgery attack on 3-step authentication of MORUS-640-128 is formed.

Key words: CAESAR competition; MORUS; differential fault attack; meet in the middle

认证加密算法能同时提供数据加密和认证两个功能,现已广泛应用于各种网络安全系统中[1].随着移动互联网的快速发展和大数据时代的来临,人们迫切需求更为安全、高效实用的认证加密算法.2013 年,在 NIST 资助下,国际密码协会 IACR 面向全球发起了征集认证加密算法的竞赛活动——CAESAR 竞赛[2].竞赛旨在加速认证加密理论的发展,选出比当前使用的 AES-GCM[3]更加高效且适应性更为广泛的认证加密算法.从 2013 年 1 月开始,CAESAR 竞赛共征集到 57 个算法,经过 3 轮评选,截止目前,有 15 个候选算法进入了第 3 轮评审,最终获胜算法于 2017 年 12 月公布.CAESAR 竞赛的进行,对候选算法的安全性分析成为了密码学界的研究热点[4−7].

MORUS 算法[8]是由 H. Wu 等人设计的基于流密码的认证加密算法,根据内部状态和密钥长度的不同,设计

* 基金项目: 国家自然科学基金(61572516, 61602514)
Foundation item: National Natural Science Foundation of China (61572516, 61602514)
收稿时间: 2016-12-17; 修改时间: 2017-02-28; 采用时间: 2017-03-13; jos 在线出版时间: 2017-03-31
CNKI 网络优先出版: 2017-03-31 21:55:04, http://kns.cnki.net/kcms/detail/11.2560.TP.20170331.2155.014.html

者提出了 3 个 MORUS 子算法:MORUS-640-128,MORUS-1280-128 和 MORUS-1280-256.MORUS 算法的设计采用了分组密码的思想,将内部状态置于 5 个相同规模的寄存器中,依次对各内部状态分组进行更新. MORUS 算法凭借其新颖的设计理念和优越的软硬件实现性能,目前进入了 CAESAR 竞赛第 3 轮评选.

故障攻击是侧信道攻击方法的一种,由 Boneh 等人于 1996 年首次提出[9].1997 年,Biham 和 Shamir 将这种攻击方法应用于对称密码体制,首次提出了“差分故障攻击”[10]的概念,并对 DES 算法进行了攻击.此后,人们先后利用该思想对许多密码算法进行了分析,包括 AES 算法[11,12]、欧洲 NESSIE 计划中的 Camellia[13,14]、e-STREAM 计划中的 Trivium[15]、Grain[16]、Mickey2.0[17]以及已经进入 CAESAR 竞赛第 3 轮评审的 ACORN 算法[18]等.差分故障分析的关键问题是如何合理地建立故障模型,包括确定故障诱导的时间、位置等.故障攻击的本质是引入差分,差分可以在算法的任何中间状态处引入,攻击原理类似差分分析[19].

目前,对 MORUS 算法的安全性分析结果主要有:

- 设计报告[8]中,设计者对算法加密阶段进行了差分分析,并直接给出分析结果.无论由 IV 引入多少比特差分,经过至多 6 步初始化运算之后,差分转移概率就会小于 2^{-201},即,可以抵抗差分分析;
- 在认证阶段,设计者采用了构造内部状态碰撞的方法分析算法抵抗伪造攻击的能力.无论算法在多少步更新后发生碰撞,发生碰撞的概率均小于 2^{-128},即,可以保证 128 比特认证标签的安全性.

文献[20]提出了在 Nonce 重用条件下的密钥流区分攻击,即:在加密阶段明文引差,再以概率 1 反应在密文上.由于攻击条件违背了 MORUS 算法的安全假设,因此并没有引起广泛关注.

文献[21]对算法初始化阶段的混乱与扩散性质进行了研究,对 4 步简化版 MORUS 算法进行了差分-区分攻击,数据量为 2^{105},区分优势为 0.999665.

总的来说,公开文献中对 MORUS 算法的分析结果还比较有限,算法的安全性还有待进一步研究.从进入 CAESAR 竞赛第 3 轮评选的 15 个候选算法目前的评估结果来看,传统的攻击方法对它们分析效果有限,而故障攻击方法是分析此类算法的一种有效手段,因此,对认证加密算法进行差分故障下的安全性分析是十分必要的.

本文分析了 MORUS 算法故障模型下的差分扩散特性.结合文献[22]中条件差分分析过程中所使用的中间相遇思想,改进了针对 MORUS 算法的差分自动搜索技术,采用面向比特的故障模型,实现了对初始化过程为 5 步的 MORUS 算法的区分攻击,以及 3 步认证过程的伪造攻击.

1 MORUS 算法简介

MORUS 算法分为初始化、关联数据处理、加密、认证码生成和解密与验证这 5 个阶段.本文中,我们主要考虑算法的初始化阶段、加密阶段和认证阶段.本文以 MORUS-640-128 算法为研究对象,若无特别声明,本文中的“MORUS”均指 MORUS-640-128 算法.下面首先说明本文所使用符号的意义,再给出 MORUS 的算法描述.

1.1 符号说明

- S^i:第 i 步的内部状态,$i\geqslant -16$;
- S_k^i:第 i 步更新第 k 轮的内部状态,$0\leqslant k\leqslant 4$;
- $S_{k,l}^i$:第 i 步第 k 轮内部状态的第 l 块 128 比特分组,$0\leqslant l\leqslant 4$;
- $S_{k,l,j}^i$:第 i 步第 k 轮内部状态第 l 块分组的第 j 比特,$0\leqslant j\leqslant 127$;
- $Rotl(x,n)$:将 128 比特长的 x 分成 4 块 32 比特字,每块循环左移 n 比特;
- 0^n:n 比特 0;
- e_i:128 比特串,其中,第 i 位为 1,其余位均为 0,$0\leqslant i\leqslant 127$;
- $(a)_4$:a 的 16 进制表示;
- $W(\alpha)$:α的汉明重量,即α二进制表示中比特 1 的个数;
- $const_0$:128 比特常数$(000101020305080d1522375990e97962)_4$;
- $const_1$:128 比特常数$(db3d18556dc22ff12011314273b528dd)_4$.

1.2 MORUS算法状态更新函数

在 MORUS 算法的每一步运算中,都有 5 轮相似运算来更新内部状态 S,其中,消息分组 m_i 参与第 2 轮至第 5 轮的更新,但没有参与第 1 轮的更新.下面给出 MORUS 算法的状态更新函数 $S^{i+1}=StateUpdate(S^i,m_i)$.

第 1 轮.

$S_{1,0}^i = Rotl(S_{0,0}^i \oplus (S_{0,1}^i \,\&\, S_{0,2}^i) \oplus S_{0,3}^i, 5);$

$S_{1,3}^i = S_{0,3}^i <<< 32;$

$S_{1,1}^i = S_{0,1}^i;$

$S_{1,2}^i = S_{0,2}^i;$

$S_{1,4}^i = S_{0,4}^i;$

第 2 轮.

$S_{2,1}^i = Rotl(S_{1,1}^i \oplus (S_{1,2}^i \,\&\, S_{1,3}^i) \oplus S_{1,4}^i \oplus m_i, 31);$

$S_{2,4}^i = S_{1,4}^i <<< 64;$

$S_{2,0}^i = S_{1,0}^i;$

$S_{2,2}^i = S_{1,2}^i;$

$S_{2,3}^i = S_{1,3}^i;$

第 3 轮.

$S_{3,2}^i = Rotl(S_{2,2}^i \oplus (S_{2,3}^i \,\&\, S_{2,4}^i) \oplus S_{2,0}^i \oplus m_i, 7);$

$S_{3,0}^i = S_{2,0}^i <<< 96;$

$S_{3,1}^i = S_{2,1}^i;$

$S_{3,3}^i = S_{2,3}^i;$

$S_{3,4}^i = S_{2,4}^i;$

第 4 轮.

$S_{4,3}^i = Rotl(S_{3,3}^i \oplus (S_{3,4}^i \,\&\, S_{3,0}^i) \oplus S_{3,1}^i \oplus m_i, 22);$

$S_{4,1}^i = S_{3,1}^i <<< 64;$

$S_{4,0}^i = S_{3,0}^i;$

$S_{4,2}^i = S_{3,2}^i;$

$S_{4,4}^i = S_{3,4}^i;$

第 5 轮.

$S_{0,4}^{i+1} = Rotl(S_{4,4}^i \oplus (S_{4,0}^i \,\&\, S_{4,1}^i) \oplus S_{4,2}^i \oplus m_i, 13);$

$S_{0,2}^{i+1} = S_{4,2}^i <<< 32;$

$S_{0,0}^{i+1} = S_{4,0}^i;$

$S_{0,1}^{i+1} = S_{4,1}^i;$

$S_{0,3}^{i+1} = S_{0,3}^i;$

S^{i+1} 更新:For k=0 to 4, $S_{0,k}^{i+1} = S_{0,k}^i$.

状态更新函数如图 1 所示.

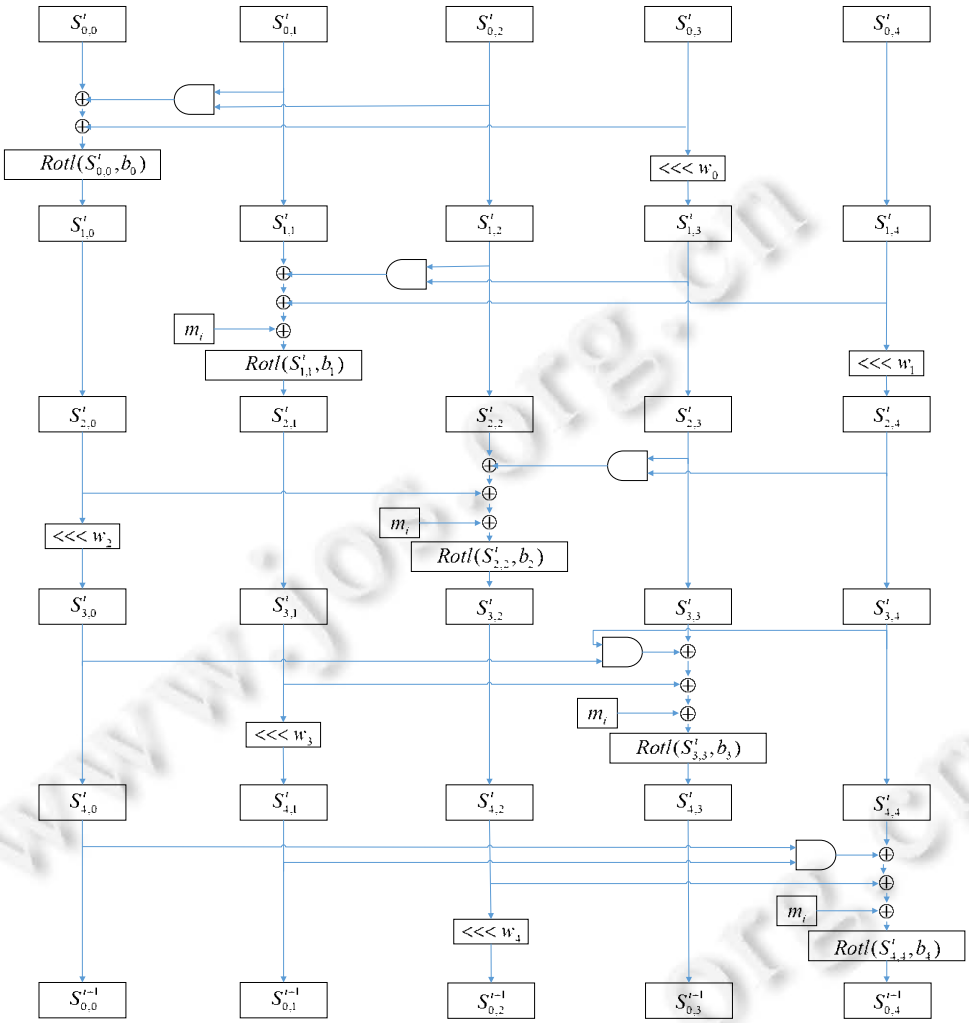

Fig.1 Update function of MORUS

图 1 MORUS 算法的状态更新函数

1.3 MORUS算法的初始化阶段

在 MORUS 的初始化阶段,将密钥 K 和初始向量 IV 注入到内部状态中,并执行 16 步状态更新函数.密钥和 IV 的加载方式如下.

$S^{-16}_{0,0} = IV$;

$S^{-16}_{0,1} = K$;

$S^{-16}_{0,2} = 1^{128}$;

$S^{-16}_{0,3} = const_0$;

$S^{-16}_{0,4} = const_1$.

将密钥 K 和初始向量 IV 注入后,进行 16 步更新.

For i=−16 to −1,

S^{i+1}=$StateUpdate(S^i,0)$;

然后,密钥与第 2 个状态分组进行异或运算,完成初始化过程.

$S_{0,1}^{0} = S_{0,1}^{0} \oplus K$.

1.4 MORUS算法的加密阶段

在算法的加密阶段,16 字节的明文分组 P_i 用于更新内部状态,同时,P_i 与密钥流进行异或运算得到密文 C_i.设明文消息的长度为 $msglen$,令 $v=\lceil msglen/128 \rceil$.

For i=0 to v−1,

$C_i = P_i \oplus S_{0,0}^{i} \oplus (S_{0,1}^{i} <<< 96) \oplus (S_{0,2}^{i} \,\&\, S_{0,3}^{i})$;

$S^{i} = StateUpdate(S^{i}, P_i)$.

1.5 MORUS算法的认证阶段

在加密阶段结束后,算法执行 8 步状态更新函数来生成认证标签 T.令 $tmp = S_3^{v} \oplus (adlen \,\|\, msglen)$,其中,$adlen$ 和 $msglen$ 均为 64 比特二进制表示.具体流程如下.

$S_{0,4}^{v} = S_{0,4}^{v} \oplus S_{0,0}^{v}$;

For i=v to v+7,

$S^{i+1} = StateUpdate(S^{i}, tmp)$;

$T = \oplus_{i=1}^{4} S_{0,i}^{v+7}$.

2 MORUS 算法的差分扩散性质分析

MORUS 算法初始化过程有 16 步,我们期望在诱导故障比特数较少的情况下,构造出轮数更长且差分转移概率更大的差分传递链,从而可以攻击更多步数的算法.下面,我们首先寻找 MORUS 算法差分自动推演规则,其次提出 MORUS 算法差分自动推演算法.

2.1 MORUS算法的差分自动推演规则

MORUS 算法内部状态较大,手工推导差分链较困难.丁林等人在文献[23]中提出了针对 Trivium 算法的差分自动推演算法,其基本思想是:根据算法状态更新函数的差分传递特征设计差分传递规则,再按照该规则使用差分值代替变元数值执行状态刷新过程.张沛等人在文献[21]中将常数的因素(初态部分已知)列入考虑,提出了针对 MORUS 算法的差分自动推演算法,进而得到一条长度为 4、转移概率为 2^{-101} 的差分传递链.

MORUS 算法的初始化过程中,第 i 步更新可以表示成如下方式.

$$S_{(k+1)\bmod 5,k}^{i} = Rotl(S_{k,k}^{i} \oplus (S_{k,(k+1)\bmod 5}^{i} \,\&\, S_{k,(k+2)\bmod 5}^{i}) \oplus S_{k,(k+3)\bmod 5}^{i}, b_k);$$

$$S_{(k+1)\bmod 5,(k+3)\bmod 5}^{i} = S_{k,(k+3)\bmod 5}^{i} <<< w_k;$$

$$S_{(k+1)\bmod 5,(k+1)\bmod 5}^{i} = S_{k,(k+1)\bmod 5}^{i};$$

$$S_{(k+1)\bmod 5,(k+2)\bmod 5}^{i} = S_{k,(k+2)\bmod 5}^{i};$$

$$S_{(k+1)\bmod 5,(k+4)\bmod 5}^{i} = S_{k,(k+4)\bmod 5}^{i}.$$

更新过程中,对差分值有影响的只有函数:

$$S_{(k+1)\bmod 5,k}^{i} = Rotl(S_{k,k}^{i} \oplus (S_{k,(k+1)\bmod 5}^{i} \,\&\, S_{k,(k+2)\bmod 5}^{i}) \oplus S_{k,(k+3)\bmod 5}^{i}, b_k).$$

而 $Rotl$ 函数只影响差分的位置,不会改变差分的值,因此可以将影响差分值的函数统一描述为

$$f(x_0,x_1,x_2,x_3)=x_1 \oplus x_2 \& x_2 \oplus x_3.$$

假设已知各变元的输入差分别为 $\Delta_0,\Delta_1,\Delta_2,\Delta_3$,$f$ 函数的输出差记为 Δ,则输出差满足等式:

$$\Delta=f(x_0,x_1,x_2,x_3) \oplus f(x_0 \oplus \Delta_0, x_1 \oplus \Delta_1, x_2 \oplus \Delta_2, x_3 \oplus \Delta_3)=\Delta_0 \oplus \Delta_3 \oplus x_1 \& \Delta_2 \oplus x_2 \& \Delta_1 \oplus \Delta_1 \& \Delta_2.$$

文献[21]利用 MORUS 完全性算法首先判断变元是否为常数:若是常数,则差分以概率 1 传递;否则,用差分值代替变元数值,以消掉变元使差分传递链能够传递下去.据此,设计出如下差分传递的规则.

规则 1. MORUS 算法状态已知时,差分自动推演规则.

(1) 当$\Delta_1=0,\Delta_2=0$ 时,$\Delta=\Delta_0\oplus\Delta_3$;

(2) 当$\Delta_1=0,\Delta_2=1$ 时,若 x_1 是常数,则$\Delta=\Delta_0\oplus\Delta_3\oplus x_1$;否则,规定$\Delta=0$;

(3) 当$\Delta_1=1,\Delta_2=0$ 时,若 x_2 是常数,则$\Delta=\Delta_0\oplus\Delta_3\oplus x_2$;否则,规定$\Delta=0$;

(4) 当$\Delta_1=1,\Delta_2=1$ 时,若 x_1 和 x_2 均是常数,则$\Delta=\Delta_0\oplus\Delta_3\oplus x_1\oplus x_2\oplus 1$;否则,规定$\Delta=0$.

其中,情形(1)成立的概率为 1;假设 x_3 和 x_4 是随机分布的(在 MORUS 中,因 x_1 和 x_2 是不同的变元且近似独立,因此该假设成立),情形(2)~情形(4)中,$\Delta=0$ 成立的概率分别为 0.5.接下来,我们根据文献[23],针对 Trivium 算法的差分自动推演算法,给出 MORUS 算法内部状态全部未知时的差分自动推演规则.

规则 2. MORUS 算法状态未知时,差分自动推演规则.

(1) 若$\Delta_1=\Delta_2=0$,则$\Delta=\Delta_0\oplus\Delta_3$;

(2) 若$\Delta_1|\Delta_2=1$,则规定$\Delta=0$.

从上述规则 2 可以看出:情形(1)成立的概率为 1;假设 x_1 和 x_2 是随机分布的(在 MORUS 中,因 x_1 和 x_2 是不同的变元且近似独立,因此该假设成立),情形(2)成立的概率为 0.5.

为方便描述,我们称 MORUS 算法状态已知时,差分自动推演规则为规则 1;MORUS 算法状态未知时,差分自动推演规则为规则 2.接下来,这两个规则将会应用到 MORUS 算法差分传递链搜索算法的具体推演过程中.

2.2 MORUS算法的差分传递链搜索算法

2012 年 8 月,在 C-QUARK 算法的设计报告中[22],SimonKnellwolf 等人提出并使用了一种高级的条件差分分析方法.该方法将原先单方向差分扩散的控制分为两个过程,按照时刻增长顺序,从两个方向进行差分推导.

- 逆向控制过程:多比特差分到单比特差分;
- 正向控制过程:单比特差分到多比特差分.

设第 i 步更新时内部状态差分为ΔS^i,利用此方法,下面对 MORUS 算法进行差分传递链的搜索,图 2 为 MORUS 算法在该方法下的差分扩散示意图.

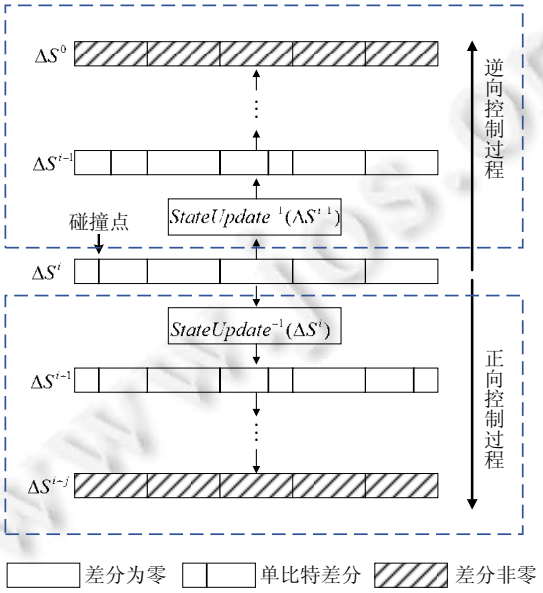

Fig.2 Differential diffusion of MORUS I

图 2 MORUS 算法差分扩散示意图 I

文献[24]将中间的单比特差分称为碰撞点;而在本文中,碰撞点的汉明重量可以大于 1.直接采用上述流程,

从“碰撞点”分别向两边推导,由中间碰撞点$\Delta S^i\neq 0$ 逆推 $i(i>0)$步得到初始差分:$\Delta S^i\to\Delta S^0$,再由中间碰撞点正推 j 步:$\Delta S^i\to\Delta S^{i+j}$,从而得到一条完整的差分传递链$\Delta S^0\to\Delta S^i\to\Delta S^{i+j}$.然而,这样做存在一个问题:对于算法的初态 S^0 来说,除了密钥和 IV 外,其余位置都是已知的常数,因此在完成逆推得到ΔS^0 后,在$\Delta S^0\to\Delta S^i$ 的前几步过程中,差分传递是受常数影响的,从而$\Delta S^0\to\Delta S^i$ 不一定成立,$\Delta S^i\to\Delta S^{i+j}$ 也不一定存在.

因此,本节借鉴文献[22]的思路,利用中间相遇的思想,寻找更加准确的差分传递链.下面给出一种针对 MORUS 算法差分推导模型.

- 逆向控制过程:多比特差分到碰撞点;
- 正向控制过程:多比特差分到碰撞点再到多比特差分.

我们采用规则 2 逆向控制得到的多比特初始差分作为正向控制过程的候选输入差分,再采用规则 1 重新推导若干轮,找到较好的差分传递链.图 3 为利用中间相遇思想得到的 MORUS 算法差分扩散示意图.

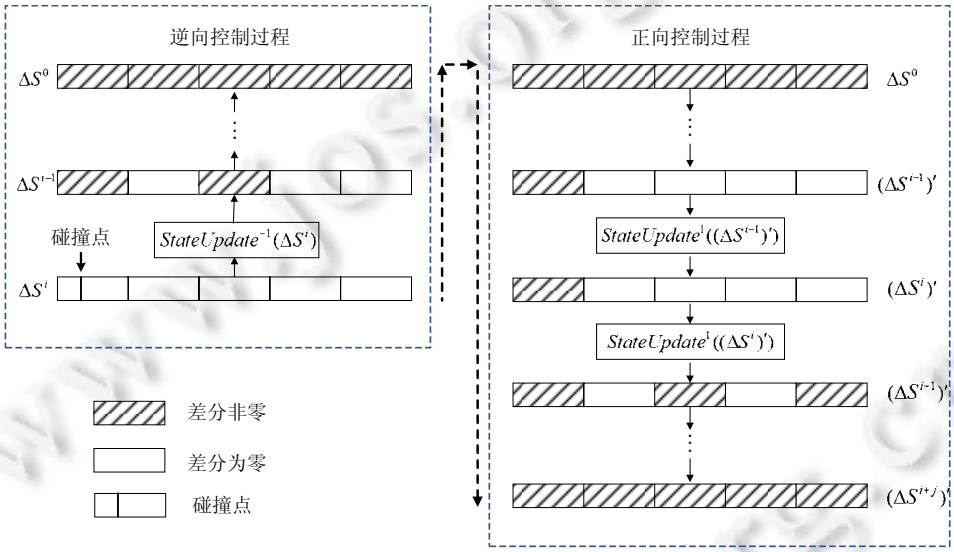

Fig.3 Differential diffusion of MORUS II

图 3 MORUS 算法差分扩散示意图 II

(1) 逆向控制过程

从初始化的第 $i(i>0)$步出发,在内部状态 S^i 中的任意 $S^i_{0,l}(0\leqslant l\leqslant 4)$ 上引入若干比特差分,利用规则 2 逆向推导 i 步得到初始输入差分ΔS^0.

运行 MORUS 完全性算法[21]发现:初始化过程第 1 步结束之后,算法内部状态的每个比特都包含了密钥信息.由于逆推是从第 $i(i>0)$步的内部状态开始推导,所以内部状态的真实值是未知的,因此采用 MORUS 算法状态未知时差分自动推演规则.

(2) 正向控制过程

在逆向控制过程中,我们得到逆推不同步的初始状态差分值.对于 MORUS 算法的初态来说,除了密钥和 IV 外,其余位置都是已知的常数,因此在差分值正向传递的过程中,不应简单地使用差分值来代替变元数值,应首先判断变元的具体数值能否确定,我们采用规则 1 作为差分链的推导规则,这样可以给出更准确的差分传递链.

我们令$\Omega_0=\{\beta_k|\beta_k$ 对应于碰撞点的差分取值ΔS^i,且 $W(\Delta S^i)>0,0\leqslant k\leqslant N-1\}$,下面给出上述过程的算法表示:

算法 1. MORUS 算法的差分链搜索算法.

输入:逆推步数 $i(i>0)$,正推步数 $j(j>0)$,候选碰撞点集合;

输出:逆推初始差分及概率集合Ω_1 和候选初始差分及概率集合Ω_2.

步骤 1. 构建逆推初始差分及概率集合$\Omega_1=\{\phi\}$,候选初始差分及概率集合$\Omega_2=\{\phi\}$,并置初始参数值 $k=0$.

步骤 2. 对碰撞点$\beta_k\in\Omega_0$,将其作为逆推的输入差分ΔS^i,利用规则 2 逆推 i 步得到ΔS^0及其相应的差分转移概率 p 存储到Ω_1,其中,$(\Delta S^0,p)$以队列中的位置序号 $k(k=0,1,2,\ldots,N-1)$为索引,记为$(\gamma_k,p_{1,k})$.

步骤 3. 对逆推初始差分γ_k,令$\Delta S^0=\gamma_k$,利用规则 1 正推 j 步,并记差分转移概率为 $p_{2,k}$,将$(\gamma_k,p_{2,k})$存储在Ω_2.

步骤 4. $k=k+1$,若 $k\leqslant N-1$,则返回步骤 2;否则,将退出.

其中,步骤 2 为逆向控制过程,由候选碰撞点出发逆推 i 步寻找候选初始差分;步骤 3 为正向控制过程,遍历候选初始差分,正向推导 j 步,得到各候选初始差分对应的差分转移概率.

通常情况下,输入差分重量较小时差分传递链的传递效果更好,所以我们在碰撞点的汉明重量不超过 3 且只分布在一个内部状态分组的情况下,即$\forall\beta\in\Omega_0,W(\beta)\leqslant 3$,编程实现算法寻找了所有逆推$i=1,2,3,4$步,正推$j=3,4,5$步 MORUS 初始化算法的差分传递链.表 1 给出了碰撞点位于内部状态的不同分组 $S^i_{0,k}(0\leqslant k\leqslant 4)$ 时最大的差分转移概率.

Table 1 Maximum differential probabilities of the initialization in MORUS under various steps

表 1 推导不同步数下 MORUS 初始化算法最大差分转移概率

推导方向	推导步数	碰撞点位置				
		$S^i_{0,0}$	$S^i_{0,1}$	$S^i_{0,2}$	$S^i_{0,3}$	$S^i_{0,4}$
逆向推导(概率对应 p_1)	1	2^{-4}	2^{-4}	2^{-5}	2^{-2}	2^{-3}
	2	2^{-13}	2^{-14}	2^{-17}	2^{-8}	2^{-10}
	3	2^{-34}	2^{-39}	2^{-46}	2^{-23}	2^{-27}
	4	2^{-82}	2^{-96}	2^{-106}	2^{-59}	2^{-69}
正向推导(概率对应 p_2)	4	2^{-32}	2^{-155}	2^{-101}	2^{-109}	2^{-141}
	5	2^{-85}	2^{-319}	2^{-207}	2^{-240}	2^{-246}
	6	2^{-183}	2^{-525}	2^{-342}	2^{-423}	2^{-396}

通过 MORUS 算法的差分链搜索算法得到.

对于 MORUS 算法的初始化阶段,以概率大于 2^{-128} 为界:当碰撞点分别位于的第 3 比特(e_3), $S^i_{0,2}$ 的第 95 比特(e_{95})和 $S^i_{0,3}$ 的第 0 比特(e_0)上时,存在 3 条长度为 4 的差分传递链,差分转移概率分别为 $2^{-32},2^{-101},2^{-109}$;当碰撞点位于 $S^i_{0,0}$ 的第 3 比特(e_3)时,存在一条长度为 5 的差分转移概率为 2^{-85} 的差分传递链.

3 故障模型下的 MORUS 算法的扩散性研究 $S^i_{0,0}$

3.1 故障模型和基本假设

尽管 MORUS 算法是面向字设计实现的,但算法中的运算均是基于比特的,为了充分利用差分传播性质,本文采用面向比特的差分故障诱导模型,其基本假设如下.

(1) 攻击者可以向状态更新过程的中间状态注入单比特故障,使其发生单比特反转,而且故障的具体位置已知;

(2) 攻击者可以精确地控制故障注入的时刻.

3.2 单比特故障下MORUS算法的差分链搜索

对第 3 节中利用差分链搜索算法得到的候选差分传递链进行进一步分析,按照下列 3 个条件进行筛选,找到最有效的差分传递链并给出故障引入位置.

(1) 初始输入差满足 $\Delta S^0_{0,1}=0$.由于 $S^0_{0,1}$ 在初始化过程加载密钥 K,考虑到现实攻击条件有限,我们只进行选择 IV 攻击,不改变 K 值;

(2) 由差分传递链决定的所需要的故障比特数尽量少;

(3) 差分传递链尽可能长,差分转移概率尽可能大.

在一台普通的 PC 机器(CPU 为 2.4GHz Intel Core i7-5500U,内存为 4GB)上,使用 C 语言(VisualC++6.0)编程

实现了搜索算法,并在计算机上对诱导故障下的状态更新进行了仿真模拟.

实验结果表明:对于 MORUS 算法的初始化阶段,以 $S_{0,0}^1$ 上的第 0 比特作为碰撞点($\Delta S_{0,0}^1 = e_0$,其余位置为 0),逆推 1 步,得到初始差分 $\Delta S_{0,0}^0 = e_{101}$,在第 3 个寄存器 $S_{0,3}^0$ 的第 96 比特上注入 1 比特故障,以此为初始输入差正向推导,可以得到一条步数为 4 转移概率为 2^{-32} 的传递链,以及一条步数为 5 转移概率为 2^{-85} 的传递链.具体见表 2,其中,$p((A,B)\to C)$表示状态输入差为 A、故障为 B、状态输出差为 C 时的差分转移概率.

Table 2 Results of differential chain research in MORUS with a single fault

表 2 单比特故障下 MORUS 算法的差分链搜索结果

差分链长度 i	状态输入差ΔS^0	故障比特	状态输出差ΔS^i	差分转移概率 p
4	$(e_{101},0^{128},0^{128},0^{128},0^{128})$	$\Delta S_{0,3}^0 = e_{96}$	(00000000000400000000000000004000, 00400010400000000000000010000000, 00000000002200000000000000002800000, 00000000000400000400100000100008, 00040050000010000000004400000400)$_4$	$p((\Delta S^0,\Delta S_{0,3}^0)\to\Delta S^3) = 2^{-32}$
5	$(e_{101},0^{128},0^{128},0^{128},0^{128})$	$\Delta S_{0,3}^0 = e_{96}$	(02080100000000000000000080020000, 00000022080002000020002020000000, 10000001010000404400800000000000, 08000900000900040a80040080020000, 00088000008022008002002012000880)$_4$	$p((\Delta S^0,\Delta S_{0,3}^0)\to\Delta S^4) = 2^{-85}$

3.3 区分攻击及结果分析

通过上述讨论,我们得到关于初态和若干步更新后内部状态的差分传递链及对应的差分转移概率,接下来将其转化为关于初态和输出密钥流的差分传递链及对应的差分转移概率.基于区分攻击的思想[9],我们使用假设检验模型,由具体的差分传递链构造区分器,用来检验算法的输出序列是否为随机序列,攻击的效率由所需数据量和区分优势来衡量.

我们对初始化 3 步、4 步的简化版 MORUS 算法进行了差分-区分攻击,给出了对应的数据量和区分优势,并与文献[21]的差分-区分攻击进行了比较,具体见表 3.

Table 3 Comparison of two types of distinguish attack on MORUS

表 3 针对 MORUS 算法区分攻击结果比较

文献	方法	步数	差分转移概率	数据量	区分优势
文献[21]	完全性-区分	3	1	2	0.999 985
	差分-区分	3	2^{-38}	2^{42}	0.999 665
	差分-区分	4	2^{-101}	2^{105}	0.999 665
本文	差分故障(1 比特)	4	2^{-26}	2^{30}	0.999 999 89
	差分故障(1 比特)	5	2^{-73}	2^{76}	0.999 665

在 ASIACRYPT2004 上发表的《How Far Can We Go Beyond Linear Cryptanalysis?》一文中,对区分攻击进行了系统的理论分析.在附录中,我们将以 4 步 MORUS 算法的区分攻击为例,根据文献[25]的理论详细给出其数据量和区分优势的计算过程.

通过比较两种攻击的结果,与文献[21]中结果相比,在攻击相同 4 步初始化算法时,我们降低了数据复杂度,提高了区分优势;在相同区分优势下,我们以更小的数据复杂度将攻击步数提高了一步.结果显示:5 步简化版 MORUS 算法对差分模型下的差分-区分攻击是脆弱的,但由于 MORUS 算法初始化过程共有 16 步,我们相信,完整的 MORUS 算法能够较好地抵抗故障下的差分-区分攻击.

4 对简化版 MORUS 算法认证部分的故障攻击

对认证加密算法消息认证部分的安全性分析分为两种:一种是内部状态碰撞攻击,即,在加密过程中对明文上引入差分,若干步加密后内部状态产生碰撞,从而实现伪造攻击;另一种是加密过程引入的差分传递到认证过程,形成一条有效的差分传递链,进而以一定的概率达到伪造认证码的目的.对于 MORUS,如果伪造认证码成功

的概率大于 2^{-128} 即算成功.

本节主要研究第 2 种情况,MORUS 算法生成认证码的过程有 8 步更新.与初始化阶段的差分故障分析类似,我们首先需要确定加密过程的差分引入和故障引入,才能攻击更多步的认证算法.设加密过程完成后内部状态为 S^v,由于 MORUS 算法在认证过程中每步都会将 *tmp* 作为消息分组来更新内部状态,若在 *tmp* 引 *n* 比特差分,相当于在内部状态上引入 4*n* 比特差分.同时,我们在 *tmp* 或任意 S_j^i 上分别引入相同比特重量的差分时进行了大量实验,得到如下观察.

观察 1. 对于状态更新函数 $S^{i+1}=StateUpdate(S^i,tmp)$,在 *tmp* 或任意 S_j^i 上分别引入相同比特重量的差分时,前者的差分扩散速度快于后者.

根据认证过程的算法特点,我们提出以下两个准则来控制差分引入和故障注入位置.

准则 I. $\Delta S_{0,3}^v=0$.

MORUS 算法在认证过程中,每步都会将 *tmp* 作为消息分组来更新内部状态,而 $tmp=S_{0,3}^v\oplus(adlen\,||\,msglen)$,由观察 1 可知,*tmp* 对差分扩散速度有很大影响.而 *adlen*||*msglen* 是固定的,因此若控制 *tmp* 没有差分,需要 $\Delta S_{0,3}^v=0$.

准则 II. $\Delta S_{0,0}^v=0$.

由于在进入认证过程的状态更新函数之前,$S_{0,4}^v$ 进行了又一次更新 $S_{0,4}^v=S_{0,4}^v\oplus S_{0,0}^v$,如果 $S_{0,0}^v$ 上有差分,在认证阶段开始前,会通过“⊕”运算扩散到 $S_{0,4}^v$ 上,将会增加认证过程初始差分重量,所以我们将 $S_{0,0}^v$ 的差分设置为 0.

基于以上两个准则,利用第 4 节中 MORUS 算法的差分传递链搜索算法,我们在加密过程的最后一步明文 P_{v-1} 上引入 1 比特差分,在加密过程最后一步后(认证开始第 1 步)的第 4 个寄存器 $S_{0,3}^v$ 上的相邻第 10,11 比特诱导发生故障翻转,将差分限定在内部状态的第 2 个~第 4 个寄存器上.图 4 展示了该差分的传播路径.

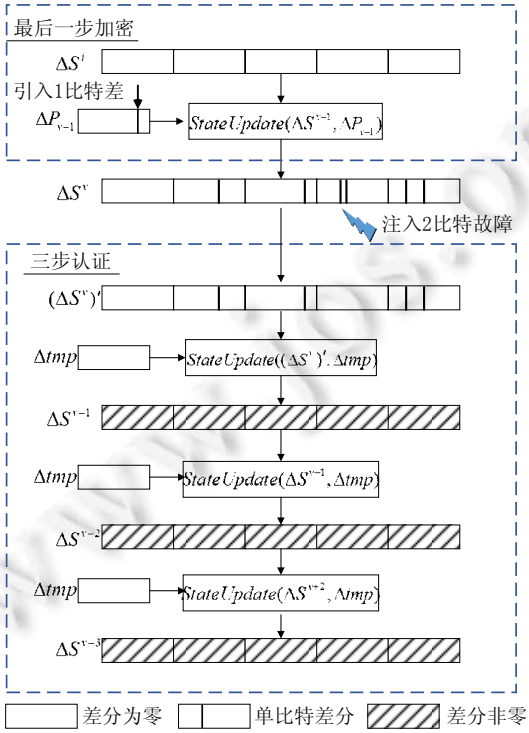

Fig.4 Differential path of the authenticated process in MORUS

图 4 MORUS 算法认证阶段的差分传播路径

表 4 给出认证过程的故障攻击结果.

Table 4 Results of fault attack on the authenticated process

表 4 认证过程的故障攻击结果

攻击步数	故障比特个数	差分概率
2	2	2^{-39}
3	2	2^{-121}

攻击过程中,篡改最后一步明文 P_{v-1} 的第 0 比特 $\Delta P_{v-1}=e_0$ 诱导 $S_{0,3}^{v}$ 上的相邻第 10,11 比特诱导发生故障翻转,经过 3 步认证后,得到的认证码 T' 以 2^{-121} 的概率成为 $T'=T\oplus\Delta T$,其中,

$$\Delta T=(02004800800801400408400104930002)_4.$$

5 结束语

本文利用 MORUS 认证加密算法的特点,给出 MORUS 算法故障模型下的差分自动搜索算法,通过诱导故障,建立起关于初态和输出密钥流的差分传递链,实现对初始化 5 步的简化版 MORUS 算法的区分攻击,攻击步数及复杂度都优于现有分析结果;对认证阶段进行分析,找到影响差分传递的关键比特位置,利用故障诱导关键比特位发生反转,首次给出了对 3 步认证的简化版 MORUS 算法的伪造攻击.由于算法不允许 Nonce 重用,我们的攻击条件限制在单比特故障下的选择 *IV* 攻击,所以目前还不能对完整算法实施攻击.

差分故障分析在实际中高效可行,尽管该攻击方法目前还不能威胁到完整的 MORUS 算法,但是可以利用这种攻击方法进一步挖掘 MORUS 的设计弱点以实施更有效的攻击;同时,本文的工作对 MORUS 算法硬件防护方面的研究具有一定的借鉴意义.

References:

[1] Bellare M, Namprempre C. Authenticated encryption: Relations among notions and analysis of thegeneric composition paradigm. Journal of Cryptology, 2008,21(4):469−491.

[2] Bernstein DJ. CAESAR call for submissions. 2014. http://competitions:cr:yp:to/caesar-call.html

[3] Mcgrew DA, Viega J. The Galois/counter mode of operation (GCM). 2004. http://www.mindspring.com/~dmcgrew/gcm-nist-6.pdf

[4] Dobraunig C, Eichlseder M, Mendel F. Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In: Proc. of the ASIACRYPT 2015. Berlin: Springer-Verlag, 2015. 490−509.

[5] Dey P, Rohit RS, Sarkar S, Adhikari A. Differential fault analysis on Tiaoxin and AEGIS family of ciphers. In: Proc. of the Security in Computing and Communications. Berlin: Springer-Verlag, 2016. 74−86.

[6] Peyrin T, Sim SM, Wang L, Zhang G. Cryptanalysis of JAMBU. In: Proc. of the Fast Software Encryption 2015. Berlin: Springer-Verlag, 2015. 264−281.

[7] Salam MI, Bartlett H, Dawson E, *et al*. Investigating cube attack on the authenticated encryption stream cipher ACORN. In: Proc. of the Applications and Techniques in Information Security 2016. Berlin: Springer-Verlag, 2016. 15−26.

[8] Wu H, Huang T. The authenticated cipher MORUS. 2014. http://competitions.cr.yp.to/caesar-submissions.html

[9] Boneh D, Demillo RA, Lipton RJ. On the importance of checking cryptographic protocols for faults. In: Proc. of the Int'l Conf. on Theory and Application of Cryptographic Techniques. Berlin: Springer-Verlag, 1997. 1175−1213.

[10] Biham E, Shamir A. Differential fault analysis of secret key cryptosystem. In: Proc. of the CRYPTO'97. Berlin: Springer-Verlag, 1997. 513−525.

[11] Piret G, Quisquater JJ. A differential fault attack technique against SPN structures, with application to the AES and Khazad. In: Proc. of the Cryptographic Hardware and Embedded Systems (CHES 2003). Berlin: Springer-Verlag, 2003. 77−88.

[12] Takahashi J, Fukunaga T, Yamakoshi K. DFA mechanism on the AES key schedule. In: Proc. of the Workshop on Fault Diagnosis and Tolerance in Cryptography. New York: IEEE, 2007. 62−74.

[13] Zhou YB, Wu WL, Xu NN, Feng DG. Differential fault attack on Camellia. Chinese Journal of Electronics, 2009,18(1):13−19.

[14] Zhao XJ, Wang T, Guo SZ. An improved differential fault analysis on Camellia. Chinese Journal of Computers, 2011,34(4): 613−627 (in Chinese with English abstract).

[15] Dey P, Adhikari A. Improved multi-bit differential fault analysis of Trivium. In: Proc. of the INDOCRYPT 2014. Berlin: Springer-Verlag, 2014. 37−52.

[16] Banik S, Maitra S, Sarkar S. A differential fault attack on the grain family of stream ciphers. In: Proc. of the Cryptographic Hardware and Embedded Systems (CHES 2012). Berlin: Springer-Verlag, 2012. 122−139.

[17] Karmakar S, Chowdhury DR. Differential fault analysis of MICKEY-128 2.0. In: Proc. of the Fault Diagnosis and Tolerance in Cryptography (FDTC). New York: IEEE, 2013. 52−59.

[18] Dey P, Rohit RS, Adhikari A. Full key recovery of ACORN with a single fault. Journal of Information Security & Applications, 2016,29:57−64.

[19] Wei YC, Li L, Li RL, Li C. Differential fault analysis on SHACAL-2. Journal of Electronics & Information Technology, 2010, 32(2):318−322 (in Chinese with English abstract).

[20] Mileva A, Dimitrova V, Velichkov V. Analysis of the authenticated cipher MORUS (v1). In: Proc. of the Cryptography and Information Security in the Balkans. Berlin: Springer-Verlag, 2016. 45−59.

[21] Zhang P, Guan J, Li JZ, Shi TR. Research on the confusion and diffusion properties of the initialization of MORUS. Journal of Cryptologic Research, 2015,2(6):536−548 (in Chinese with English abstract).

[22] Aumasson JP, Knellwolf S, Meier W. Heavy quark for secure AEAD. 2012. http://www.hyperelliptic.org/DIAC/slides/cQuark.pdf

[23] Ding L, Guan J. Differential cryptanalysis based on automatic deduction of Trivium stream cipher. Chinese Journal of Electronics, 2014,42(8):1647−1652 (in Chinese with English abstract).

[24] Zang K. Security analysis on three typical mixed symmetric ciphers [MS. Thesis]. Zhengzhou: Information Engineering University, 2013 (in Chinese with English abstract).

[25] Baignères T, Junod P, Vaudenay S. How far can we go beyond linear cryptanalysis? In: Proc. of the Advances in Cryptology—ASIACRYPT 2004. Berlin: Springer-Verlag, 2004. 432−450.

附中文参考文献:

[14] 赵新杰,王韬,郭世泽.一种针对 Camellia 的改进差分故障分析.计算机学报,2011,34(4):613−627.

[19] 魏悦川,李琳,李瑞林,等.SHACAL-2 算法的差分故障攻击.电子与信息学报,2010,32(2):318−322.

[21] 张沛,关杰,李俊志,等.MORUS 算法初始化过程的混乱和扩散性研究.密码学报,2015,2(6):536−548.

[23] 丁林,关杰.Trivium 流密码的基于自动推导的差分分析.电子学报,2014,42(8):1647−1652.

[24] 张凯.三类典型混合对称密码算法的安全性分析[硕士学位论文].郑州:解放军信息工程大学,2013.

附 录

在区分攻击中,攻击者需要设计区分器检验输出序列是否为随机序列,区分器的效率用区分优势(记为 adv)进行衡量.使用区分器时,存在如下两种误判.

- 误判 1:输出序列为随机的,但判断其为非随机序列;
- 误判 2:输出序列为非随机的,但判断其为随机序列.

记以上两种误判发生的概率分别为α和β,则根据文献[25]的结论可得如下引理.

引理 1[25]. 最优区分器的区分优势满足:

$$adv=1-(\alpha+\beta),$$

其中,α和β分别代表两种误判发生的概率.这里,我们以对 4 步 MORUS 算法的区分攻击为例,给出其数据量和区分优势的计算过程.

在选择 IV 的攻击模型下,给定一个未知的密钥 K,任意选择一个 128 比特 IV,将其中的第 101 比特取反得到 IV^*.将(K,IV)和(K,IV^*)分别加载到 MORUS 算法中,经过 4 轮的初始化过程,生成 128 比特密钥流,分别记为 Z 和 Z^*,检测密钥流输出差是否为给定的差值:若符合,则称为事件 L 发生.若攻击者随机选择 M 个(K,IV)和相应的

(K,IV^*)进行检测,由于 $\Pr(L)=2^{-26}$ 成立,则 M 次检测中事件 L 至少发生一次的概率为

$$P_1=1-(1-2^{-26})^M.$$

而在随机情况下,事件 L 发生的概率为 2^{-128},M 次检测中事件 L 至少发生一次的概率为

$$P_2=1-(1-2^{-128})^M.$$

已知 $\lim\limits_{n\to+\infty}\left(1-\frac{1}{n}\right)^n=\mathrm{e}^{-1}$,其中,e 为自然数.则可得:

$$P_2\approx 1-\mathrm{e}^{-\frac{M}{2^{128}}},P_1\approx 1-\mathrm{e}^{-\frac{M}{2^{26}}}.$$

为使攻击具有较高的区分优势,M 的选择应使得 P_1 十分接近于 1 且远远大于 P_2.当选择 2^{30} 个(K,IV)和相应的(K,IV^*)对 MORUS 算法进行区分攻击时,P_1 十分接近于 1 且远远大于 P_2,将事件 L 发生作为判据,2^{30} 个选择 IV 的数据量可以保证事件 L 以接近 1 的概率发生.当事件 L 发生时,攻击者判断该输出序列为 4 步 MORUS 算法的密钥流;当事件 L 不发生时,攻击者判断该输出序列为随机序列.有可能发生的误判有两种,分别为:

- 误判 1:事件 L 发生,而输出序列为随机序列;
- 误判 2:事件 L 不发生,而输出序列为简化版本的 MORUS 的密钥流.

误判 1 发生意味着在输出序列为随机的情况下事件 L 发生,误判 2 发生意味着在输出序列为密钥流的情况下事件 L 不发生.故易知误判 1 发生的概率为 $\alpha=P_2=1-e^{-2^{-98}}$,而误判 2 发生的概率为$\beta=1-P_1=1-0.99999989=0.00000011$.根据引理 1 易知,对 4 步 MORUS 算法区分攻击的区分优势为

$$Adv=1-(\alpha+\beta)=\mathrm{e}^{-2^{-98}}-0.00000011\approx 1-0.00000011=0.99999989.$$

施泰荣(1992－),女,山东临沂人,硕士,主要研究领域为对称密码设计与分析.

李俊志(1990－),男,硕士,主要研究领域为序列密码.

关杰(1974－),女,教授,博士生导师,主要研究领域为密码技术与分析.

王森鹏(1990－),男,硕士,主要研究领域为密码算法设计与分析.