DOI:10.13229/j.cnki.jdxbgxb2012.05.024

基于抵抗故障引入攻击的 RC4 算法的改进

胡 亮[1],迟 令[1],袁 巍[2],初剑峰[2],徐小博[2]

(1. 吉林大学 软件学院,长春 130012;2. 吉林大学 计算机科学与技术学院,长春 130012)

摘 要:由 Adi Shamir 提出的故障引入攻击是目前破解 RC4 算法的非常有效的方法,其攻击的目标是 RC4 的伪随机生成阶段。为了抵抗这种攻击,增加伪随机生成阶段的安全性是十分有效的解决方案。本文提出了两种改进方案,其思想是在原算法的伪随机生成阶段中加入行移位和列移位过程,使其非线性变换更复杂。在对改进算法的正确性、安全性以及使用故障引入攻击破解改进算法的效率进行分析之后,证明了攻击者用故障引入攻击破解 RC4 算法的效率低于穷举攻击,可以抵抗故障引入攻击。对两种改进算法的效率和安全性进行比较,改进算法 1 更有效率,改进算法 2 更安全。

关键词:计算机系统结构; RC4 算法;故障引入攻击;非线性变换;行移位

中图分类号:TP309.7 **文献标志码**:A **文章编号**:1671-5497(2012)05-1231-06

Improvements against fault induction attack for RC4 algorithm

HU Liang[1], CHI Ling[1], YUAN Wei[2], CHU Jian-feng[2], XU Xiao-bo[2]

(1. *College of Software, Jilin University, Changchun* 130012, *China*; 2. *College of Computer Science and Technology, Changchun* 130012, *China*)

Abstract: The Fault Induction Attack (FIA) proposed by Adi Shamir is an effective method in cracking RC4 algorithm. The target of the FIA is the Pseudo-Random Generation (PGR) phase of the RC4 algorithm. To resist this attack, it is necessary to enhance the security of the PGR phase. This paper presents two improvements that make the nonlinear transformation more complex by adding the row shift and column shift to the PGR phase. After analyzing the validity, security and the efficiency of the FIA, it is demonstrated that the efficiency of using FIA is lower than using exhaustive attack, which means that the improvements could resist FIA. Comparison of the two improvements shows that the first improvement is more efficient and the second improvement is more secure.

Key words: computer system organization; RC4 algorithm; fault induction attack; nonlinear transformation; line shift

Rivest 设计的 RC4 算法[1-3]是流密码攻击者的主要攻击对象。文献[4]提出故障引入攻击,该攻击的目标是通过在密钥加密的中间过程 PGRA 中引入“暂时随机”的字节错误,利用错误

收稿日期:2011-06-27.

基金项目:国家自然科学基金项目(61103197,61073009);“973”国家重点基础研究发展规划项目(2009CB320706);“863”国家高技术研究发展计划项目(2011AA010101);吉林省重大科技攻关项目(2011ZDGG007).

作者简介:胡亮(1968-),男,教授,博士生导师. 研究方向:网格计算与网络安全. E-mail:hul@mail.jlu.edu.cn

通信作者:徐小博(1978-),男,博士研究生. 研究方向:网格计算与网络安全. E-mail:goshawkchi@yahoo.com.cn

的加密结果,通过判断错误的类型并加以分析,可以恢复 PGRA 过程中的初始状态 S_0,这样,不需要破解出种子密钥就可以破解 RC4 算法。故障引入攻击由于可以利用正确的和错误的加密结果进行分析比较,也被称为差分错误分析。进一步分析这些攻击的模式和特点[5-7]后发现,目前针对 RC4 算法的主要攻击都集中在 RC4 算法两个处理阶段之一的伪随机生成阶段。针对以上问题,本文基于一种图表化的 RC4 算法结构,通过在算法中合理地加入行移位和列移位步骤,提出两种改进算法:第一种改进算法是通过增加行移位和列移位过程改进字节交换步骤;第二种改进算法是通过增加行移位和列移位过程改变伪随机生成阶段中变量的选取方式。两种改进算法都是通过增加攻击者使用故障引入攻击的破解 RC4 算法的难度,使之不如穷举攻击的效率高,从而增加算法安全性。

1 RC4 算法

RC4 算法是典型的基于非线性数组变换的序列密码。实现 RC4 算法需要 2 个处理过程:密钥调度算法(Key-scheduling algorithm,KSA)和伪随机生成算法(Pseudo random-generation algorithm,PRGA)。

RC4 算法如下:

RC4_Run(N,K):

KSA:

(1)$S_0[i]=i(i=0,1,2,\cdots,255)$,$i_0=0$,$j_0=0$;

(2)$i_t=i_{t-1}+1$;

(3)$j_t=j_{t-1}+S_{t-1}[i_{t-1}]+K[i_{t-1}\text{ modl}]$;

(4)$S_t[i_t]=S_{t-1}[j_t]$;$S_t[j_t]=S_{t-1}[i_t]$,$t=1,2,\cdots,255$

PGRA:

(1)$i_0=0$,$j_0=0$;

(2)$i_t=i_{t-1}+1$;$j_t=j_{t-1}+S_{t-1}[i_t]$;

(3)$S_t[i_t]=S_{t-1}[j_t]$;$S_t[j_t]=S_{t-1}[i_t]$;

(4)$Z_t=S_t[S_t[i_t]+S_t[j_t]]$

算法中,$N=2^n=256$,i_t、j_t 为两个参数,K 为种子密钥,l 为其长度,$l=K$ 的比特数/256,Z_t 为 t 时刻的输出值。加密时,将 Z_t 与明文异或;解密时,将 Z_t 与密文异或。

2 故障引入攻击

故障引入攻击假设攻击者控制了密码设备,攻击者可以向密码设备中引入错误,影响加密过程,使密码设备输出错误的加密结果,它的目标是利用错误的加密结果找出秘密信息。衡量这些分析方法的参数是攻击过程中所需的密钥字和引入的错误次数。分析方法步骤如下:

Normal_Run:正常运行 RC4 算法一次,记录足够多的输出密钥字。

Fault_RunT(t):重新运行 RC4 算法,在 i_t 和 j_t 更新为 i_{t+1} 和 j_{t+1},且 $Swap(S_t[i_{t+1}],S_t[j_{t+1}])$ 未执行前,向 S 盒的第 t 个位置引入错误,记录第一个错误的输出时刻 T'。

Fault_RunT(t,n):重新运行 RC4 算法,在 i_{t-1} 和 j_{t-1} 更新为 i_t 和 j_t,且 $Swap(S_{t-1}[i_t],S_{t-1}[j_t])$ 未执行前,向 S 盒的第 t 个位置引入错误,记录引入错误后得到的第 t 个输出密钥字为 Z'_t 和正确的第 t 个输出密钥字为 Z_t。

Judge:通过判断故障引入攻击后产生的错误密钥字的类型,就可以成功地找出初始状态的值。算法描述如下:

(1)Normal_Run;

(2)t=1,j=0;

(3)对每个 $i\in[0,255]$,令 is_det[i]=0;

(4)Fault_RunT(t);

(5)令 n=0;

(6)If(n=t), then n=n+1, goto (7); else goto (7);

(7)Fault_RunT(n);

(8)If($Z'_t=Z_t$), then {if(n<256), then {n=n+1; goto (6)} else goto (9)}; else

①继续运行(7),记录下一个错误的输出时刻 T;

②If(T=T′), then {temp1=(n−j); j=n; type_A=1; if(is_det[t]=0), then $S_0[j]=(n-j)$};

③If(T≠T′), then {temp2=n; type_B=1; f(is_det[t]=0), then $S_0[n]=Z_t$};

④If(type_B=1&&type_A=1&&is_det[t]=0), then $S_0[j]=$(temp1−temp2); goto (9);

⑤If(type_B=0&&type_A=1&&is_det[j]=0&&n=256), then {if(temp1=Z_t), then {$S_0[j]=$(j−temp1); goto (9)} else {$S_0[j]=Z_t$; goto (9)}};

⑥If(type_B=1&&type_A=0&&is_det[j]=0&&n=256), then {$S_0[t]=$(t−j); goto (9)};

⑦If(n<256) then {n=n+1; goto (6)};

(9)If(t<256), then {t=t+1; goto (4)} else End.

Hoch 和 Shamir 对 RC4 实施故障引入攻击的结果表明,需要 2^{26} 个密钥字和 2^{16} 次故障引入可以恢复 RC4 的整个初始状态。Biham[8]等人在专门研究了 RC4 的故障引入分析攻击后,得到了

需要 2^{16} 个密钥字和 2^{16} 次故障引入可以恢复 RC4 的整个初始状态的结果。

3　基于 RC4 算法图表化表示方法的改进

3.1　改进算法的初始化

对现有攻击 RC4 方法的研究发现[9-10]，目前针对 RC4 算法的大多数攻击都集中在 PGRA 步骤上，最终的目标都是恢复 PGRA 中的初始状态 S_0，这样就可以恢复出所有密钥字节。在 PGRA 阶段，参与密钥字节输出的只有 2 个字节，并且在字节交换步骤后，前后的状态 S 变化不大，这个特点使 PGRA 阶段成为攻击者的主要攻击对象。针对以上情况，可以对 PGRA 步骤进行改进，通过增加算法的复杂度保护算法不被轻易破解。本文用一种图表化的方法将原算法进行重新描述，并且为后面的改进做初始化工作。

将 0～255 全部转换成由两位 16 进制数组成的字节，例如，0 转换为 00，26 转换成 1A…，将这 256 个字节按从小到大顺序，从左到右、从上到下依次排入一个 16×16 的图表中，如表 1 所示。转换完毕后，原数组中数的位置都可以在表中用第 X 行第 X 列的方式表示。

表 1　初始状态表

Table 1　Initial status table

	0	1	2	3	…	D	E	F
0	00	01	02	03	…	0D	0E	0F
1	10	11	12	13	…	1D	1E	1F
2	20	21	22	23	…	2D	2E	2F
3	30	31	32	33	…	3D	3E	3F
⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮
D	D0	D1	D2	D3	…	DD	DE	DF
E	E0	E1	E2	E3	……	ED	EE	EF
F	F0	F1	F2	F3	…	FD	FE	FF

3.2　改进算法 1 的图表化表示

改进算法 1 通过对原算法 PGRA 步骤的字节交换步骤进行算法复杂化来保护加密信息。

Step1 执行交换 $S[i_t]$、$S[j_t]$ 之前的原算法步骤。

Step2 执行新的交换步骤，模式如下：$S[i_t]=X_{m_t}Y_{m_t}$，$S[i_t]$ 所在的位置字节为 $X_{m_t}Y_{m_t}$（$m_t=1\sim F$），$S[j_t]$ 所在的位置的字节为 $X_{n_t}Y_{n_t}$（$n_t=1\sim F$），将所在行向右进行行移位，移动 $|X_{n_t}-X_{m_t}|$ 位，使 $S[i_t]$ 与 j_t 指定的位置在同一列上，然后将 $S[i_t]$ 所在列向下进行列移位，移动 $|Y_{n_t}-Y_{m_t}|$ 位，使之最终到达原 j_t 所指定的位置。采取这样的交换方式，同时对 j_t 所在位置的值 $S[j_t]$ 进行行移位，向右移动 $(16-|X_{n_t}-X_{m_t}|)$ 位，列移位向下移动 $(16-|Y_{n_t}-Y_{m_t}|)$ 位，使它到达原 $S[i_t]$ 的位置。特殊地，当参与交换的两个字节在同一行时，行移位与列移位的顺序交换，且列移位的位数为 $S[i_t]$ 的第二个 16 进制数；当参与交换的两个字节在同一列时，行移位的位数为 $S[i_t]$ 的第一个 16 进制数。以此代替原有的交换步骤。

Step3 执行原算法的输出密钥字节步骤。

由于 Step1 和 Step3 与原算法一致，就不在此赘述，只列出 Step2 的具体算法：

(1)$i_t=a[X_{m_t}][Y_{m_t}]$，$m_t=0\sim15$；

(2)For (k=0 to 15)

{$a[X_{m_t}][Y_k]=a[X_{m_t}][Y_{k+|X_{n_t}-X_{m_t}|}]$}；

{$a[X_{n_t}][Y_k]=a[X_{n_t}][Y_{k+|X_{n_t}-X_{m_t}|}]$}；then

$a[X_k][Y_{m_t}]=a[X_{k+(16-|Y_{n_t}-Y_{m_t}|)}][Y_{m_t}]$；

$a[X_k][Y_{n_t}]=a[X_{k+(16-|Y_{n_t}-Y_{m_t}|)}][Y_{n_t}]$

3.3　改进算法 2 的图表化表示

改进算法 2 利用行移位和列移位，通过改变变量 i_t 和 j_t 的选取方式来抵御攻击。

假设初始状态如表 1 所示。

Step1 例如，当 $i_0=F0$，$j_0=0+S[i_0]=F0$，则在初始状态表中搜索 F0 位置的值，如表 1 所示为 22。然后，将第 F 行所有字节向右移动 3 位，将第 0 列向上移动 3 位，得到新的 j_0，以这种方法表示原算法的变量 j，因此得到的新图表如表 2 所示。

表 2　改变后的状态表

Table 2　Changed table

	0	1	2	3	…	D	E	F
0	56	92	A2	09	…	5D	77	45
1	FF	24	E7	56	…	33	EF	C4
2	23	8D	63	25	…	AB	C0	42
3	48	36	DE	DA	…	41	00	21
⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮
D	AC	05	86	2A	…	0E	14	3C
E	3E	0F	55	B3	…	D4	D9	F1
F	8A	02	22	49	…	17	37	CF

表示成一般模式为：确定 i_t，此时 $S[i_t]=X_{m_t}Y_{m_t}$，$j_t=j_{t-1}+S_{t-1}[i_t]$；$S[j_t]=X_{n_t}Y_{n_t}$，则 $X_{n_t}Y_{n_t}$ 所在行首先向右移 $X_{n_t}+1$ 位，然后 $X_{n_t}Y_{n_t}$ 所在列向上移 $Y_{n_t}+1$ 位。移位后新的 $S[j_t]=X'_{n_t}Y'_{n_t}$。

Step2 $Swap(S'[i_t],S[j_t])$。

Step3 $S[(S'[i_t]+S[j_t])\ \mathrm{mod}\ FF]=$

$X'_{m_t}Y'_{m_t}$,然后$X'_{m_t}Y'_{m_t}$所在行向右移动$X'_{m_t}+1$位,其所在列向上移动$Y'_{m_t}+1$位,得到新的$S[(S'[i_t]+S[j_t])\bmod FF]=X''_{m_t}Y''_{m_t}$,即为输出字节。

Step1 具体算法如下:

(1)$S[i_t]=X_{m_t}Y_{m_t}$,$j_t=j_{t-1}+S_{t-1}[i_t]$,$S[j_t]=a[X_{m_t}][Y_{m_t}]=X_{n_t}Y_{n_t}$;

(2)For(k=0 to 15)

{$a[X_{m_t}][Y_k]=a[X_{m_t}][Y_{k+n_t+1}]$;

$a[X_k][Y_{m_t}]=a[X_{k+n_t+1}][Y_{m_t}]$}.

新的$a[X_{m_t}][Y_{m_t}]=X'_{n_t}Y'_{n_t}$

Step3 的具体算法与 Step1 相近,不在此赘述。

4 正确性分析

下面证明改进算法1和改进算法2能正确运行。在$t=p_t$时,明文字节为$M_{p_t}=X_{p_t}Y_{p_t}$。运行改进算法1:

Run_Step1:$j_{p_t}=S[i_{p_t}]=X_{n_t}Y_{n_t}$。

Run_Step2:得到新的状态S_{p_t+1}。

Run_Step3:$Z_t=S[S[i_{p_t}]+S[j_{p_t}]]=X_{q_t}Y_{q_t}$。

加密过程:$M_{p_t}\oplus Z_t=X_{p_t}Y_{p_t}\oplus X_{q_t}Y_{q_t}=E_{p_t}$(⊕为异或运算)。

解密过程:$E_{p_t}\oplus Z_t=X_{p_t}Y_{p_t}\oplus X_{q_t}Y_{q_t}\oplus X_{q_t}Y_{q_t}=M_{p_t}$。

证明成立。继续则可推得改进算法1可以正确运行加解密操作。同理可证改进算法2可以正确运行加解密操作。

5 安全性分析

5.1 改进算法1的安全性

改进算法1在字节交换步骤加入行移位和列移位步骤,使得在字节交换时有更多的字节参与进来。虽然与输出字节产生直接相关的只有2个字节,但是其交换步骤不仅是2个字节参与,而是有2行字节进行了行移位操作,2列字节进行了列移位操作。i_t是自增加的,不依赖于j_t和S_t,当i_t取遍$0\sim255$时,恰好跑遍S_0的每个位置的值,由于i_t不依赖于别的变量,因此i_t可以认为是公开的。j_t不是自增加的,它是加了$S[i_t]$之后增加的,由于$S[i_t]$的不可预测性,可以考虑j_t是一个伪随机函数,j_t就是一个以不可预测方式指向S_t每个位置的伪随机变量,且j_t的增加依赖于其他变量,它可以看成是不公开的。同时进行行移位的其他位置,其位移增加依赖于i_t和j_t位置的列距离差,这也是不可预测的,也可以看成是一个伪随机变量。列移位的情况同理。

用故障引入攻击分析新的改进算法,由于行移位和列移位的影响,会产生4种错误:①在第n位引入的错误恰好是第$S_t[i_t]+S_t[j_t]$位,由于引入错误在字节交换之前,所以该位置的字节可能被行移位和列移位移动到其他位置,而第t个输出字节不会出现错误,但是,S_1已经被引入了错误的字节,最终也会输出错误的字节,输出错误字节的时刻与T'不同,符合B类错误的判断。②由于输出字节在交换字节之后,被行移位和列移位的字节存在恰好被移动到$S_t[i_t]+S_t[j_t]$位的可能,最终运行后的结论也符合B类错误的判断。③引入错误的位置恰好是第$S_t[i_t]+S_t[j_t]$位,而该字节不会受到行移位和列移位的影响,符合B类错误的判断。④引入错误的位置恰好是第j_t位,符合A类错误的判断。行移位和列移位改变了字节交换步骤的模式,这样由于错误的判断类型增加且难以分辨,错误分析的工作量也会随之大幅增加,文献[11]得到的至少需要2^{16}次故障引入在改进后,会变成至少需要$2^{2^{16}}$次故障引入,严重降低了故障引入攻击的效率,还不如使用穷举攻击,这使得算法的安全性大幅增加。

5.2 改进算法2的安全性

改进算法2在伪随机变量的选取上引入了行移位思想,改变了它的选取方式。在伪随机变量确定前,先进行行移位和列移位,行移位和列移位的位数仅与$X_{n_t}Y_{n_t}$有关;输出字节与$X'_{n_t}Y'_{n_t}$有关。这样最终的输出字节$X''_{n_t}Y''_{n_t}$与$X_{m_t}Y_{m_t}$、$X_{n_t}Y_{n_t}$、$X'_{m_t}Y'_{m_t}$、$X'_{n_t}Y'_{n_t}$都相关。这几个变量中,$X_{m_t}Y_{m_t}$由i_t确定,其余3个变量的确定都依赖于其他变量,可以看成是不公开的,是伪随机变量,因此,改进算法2在确定密钥输出字节的伪随机变量数量上要多出2个。

用故障引入攻击分析改进算法2,会产生4种错误:①在$X_{m_t}Y_{m_t}$选定后,$X_{n_t}Y_{n_t}$出现错误,这会导致$X'_{n_t}Y'_{n_t}$出现错误,出现错误的是j_t,符合A类错误判断。② 行移位和列移位之后,被引入错误的字节恰好成为$X'_{n_t}Y'_{n_t}$,出现错误的是j_t,符合A类错误判断。③$X'_{m_t}Y'_{m_t}$、$X'_{n_t}Y'_{n_t}$两个字节都可能是被引入错误的字节,而产生的错误类型都符合B类错误判断,与前两种错误情况类似,攻

击者也同样要考虑到两种错误。④被引入错误的字节恰好被第一次行移位和列移位步骤移动到其他位置，与改进算法 1 中第①种错误情况类似，这种错误情况符合 B 类错误判断。因此，改进算法 2 同样使得错误类型增加，错误类型难以判断，故障引入分析的工作量大幅度增加，至少需要 $2^{2^{16}}$ 次故障引入，使得算法的安全性大幅增加。

5.3　算法改进后的安全隐患情况

无论是改进算法 1 还是改进算法 2，本质上都没有改变 RC4 算法具备的非线性特点。正是由于 RC4 算法的非线性特点，使得 RC4 算法成为故障引入攻击这种特殊的差分分析攻击的对象。但是相对于普通的对流密码的差分攻击，RC4 算法所具备的非常大的内部状态集和密钥空间，使得在故障引入攻击出现之前，大多数研究都只能局限于对 RC4 算法特定字节弱密钥性的概率分析[10-11]。而两种改进算法并未改变这一点，根据 S_t、i_t、j_t 和 N 的取值，内部状态的取值有 $2^{2n}(N!)$ 种，与原 RC4 算法相同，即不考虑故障引入攻击的情况下，两种改进算法与原算法具有同样的安全性，不存在因算法改进带来的其他隐患。

6　效率分析

RC4 算法目前主要被应用于加密无线网的数据传输，由于该算法运算简单，因此，它更多地被集成到芯片中。改进算法虽然增加了行移位和列移位，但是这两种算法在运算本质上与原算法的字节交换步骤是相同的，所以，改进算法也可以被集成到芯片中。下面具体对比改进前和改进后算法的效率。

当 $n=8$ 时，对于 n 字节的明文，原算法为生成加密明文的密钥输出序列，进行了 $5n$ 次模加运算，$2n$ 次字节交换运算。改进算法 1 进行了 $5n$ 次模加运算，n 次字节交换运算，n 次行移位运算和 n 次列移位运算。改进算法 2 进行了 $5n$ 次模加运算，n 次字节交换运算，$2n$ 次行移位运算和 $2n$ 次列移位运算。即平均每加密一个明文字节，原算法会进行 5 次模加运算和 2 次字节交换运算；改进算法 1 每生成一个字节，运行 5 次模加运算、1 次字节交换运算、1 次行移位运算和 1 次列移位运算；改进算法 2 运行 5 次模加运算、1 次字节交换运算、2 次行移位运算和 2 次列移位运算。图 1 给出了两种改进算法分别相对于原算法的效率比。

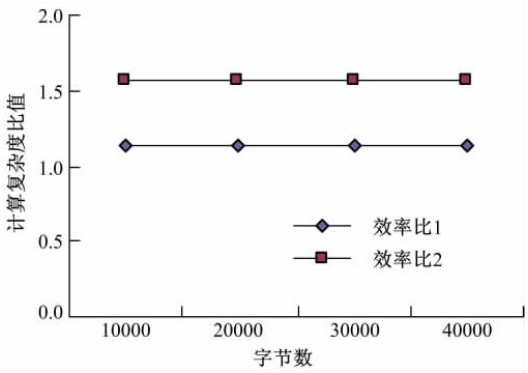

图 1　效率分析

Fig. 1　Analysis of efficiency

根据以上分析，相对原算法，两种算法都降低了效率。如果加密方不希望降低太多效率，可以使用改进算法 1；如果加密方希望密文更安全，可以使用改进算法 2。

7　结束语

针对由 Adi Shamir 提出的故障引入攻击，并综合了其他攻击方法的基础上，本文基于一种图表化的 RC4 算法结构，通过在算法中合理地加入行移位和列移位步骤，提出了两种改进方法。通过增加非线性变换的复杂度解决了 RC4 算法两个处理过程之一的伪随机生成阶段的安全问题。同时，两种算法并未改变原 RC4 算法具备的非线性特点，使它们在不考虑故障引入攻击时与原算法具备同样的安全性，不存在因算法改进带来的其他安全隐患。改进算法 1 在字节交换步骤中引入行移位和列移位，改进算法 2 在变量选取中引入行移位和列移位，两种算法通过增加故障引入攻击对引入错误情况的判断难度，以增加故障引入攻击实施的时间复杂度，保证算法的安全性。在经过缜密的正确性、安全性分析后，本文证明了两种改进算法对故障引入攻击的有效防御性，但是相对原算法，两种算法都降低了效率。如果加密方不希望降低太多效率，可以使用改进算法 1；如果加密方希望密文更安全，可以使用改进算法 2。

参考文献：

[1] Rivest R L. The RC4 encryption algorithm[Z]. RSA Data Security, Inc, 992.

[2] Forouzan Behrouz A. Cryptography and Network Security[M]. NewYork: McFraw-Hill, 2008.

[3] 谷利泽,郑世慧,杨义先. 现代密码学教程[M]. 北京:北京邮电大学出版社,2009.

[4] Hoch Jonathan J, Shamir Adi. Fault analysis of stream ciphers[C]// CHES 2004. Berlin: Springer-Velag, 2004.

[5] 杜育松,沈静. 对RC4算法的故障引入攻击研究[J]. 电子科技大学学报,2009,38(2):253-257.
Du Yu-song, Shen Jing. Research on fault induction attack on RC4 algorithm[J]. Journal of University of Electronic Science and Technology of China, 2009, 38(2): 253-257.

[6] Miyaji A, Sukegawa M. New analysis based on correlations of RC4 PRGA with nonzero-bit differences [J]. IEICE Transactions on Fundamentals of Electronic Communications and Computer Sciences, 2010, E93A(6): 1066-1077.

[7] Sichani M H, Movaghar A. A new analysis of RC4 a data mining approach(J48)[C]// Proceesdings of the International Conference on Security and Cryptography, Portugal 2009.

[8] Biham E, Granboulan L, Nguyen P Q. Impossible fault analysis of RC4 and differential fault analysis of RC4[C]// FSE 2005, Berlin, Germany: Springer-Verlag, 2005.

[9] Chen J G, Miyaji A. Generalized RC4 key collisions and hash collisions[J]. Lecture Notes in Computer Science, 2010, 6280:73-87.

[10] Chen J G, Miyaji A. New class of RC4 colliding key pairs with greater hamming distance[J]. Lecture Notes in Computer Science, 2010, 6047: 30-44.

[11] Matsui M. Key collisions of the RC4 stream cipher [J]. Lecture Notes in Computer Science, 2009, 5665: 38-50.