PRGA 的初始状态与 RC4 算法的安全性

王信敏[1],郑世慧[2]
WANG Xin-min[1],ZHENG Shi-hui[2]

1.中国石油大学(华东) 经管学院,山东 东营 257061
2.北京邮电大学 信息安全中心,北京 100876
1.College of Economic Administration,China University of Petroleum,Dongying,Shandong 257061,China
2.Information Security Center,Beijing University of Posts and Telecommunications,Beijing 100876,China
E-mail:wxmwer@163.com

WANG Xin-min,ZHENG Shi-hui.PRGA's initial state and RC4's security.Computer Engineering and Applications,2009,45(8):107-108.

Abstract: PRGA's initial state plays an important role in the security of RC4 algorithm.In this paper,the relationship between PRGA's initial state and RC4's security was analyzed.On the basis of Knudsen's analysis,this paper put forward a new computation method,and solved RC4's complexity problem under the initial state of PRGA.When some values of initial state are known,RC4's complexity can be computed efficiently by this method.
Key words: RC4 algorithm;Pseudo-Random Generation Algorithm(PRGA);initial state;complexity

摘 要:PRGA 初始状态的取值情况对 RC4 算法的安全性具有决定意义,分析了 PRGA 初始状态的取值情况与 RC4 算法安全性的关系。在猜测赋值分析方法的基础上,提出了新的复杂度计算方法,从而解决了 PRGA 初始状态取值情况对应的破译 RC4 算法的复杂度问题。在已知初始状态的部分取值的情况下,该方法能够非常有效地计算出破译 RC4 算法的复杂度。
关键词:RC4 算法;伪随机数生成算法;初始状态;复杂度
DOI:10.3778/j.issn.1002-8331.2009.08.032 **文章编号**:1002-8331(2009)08-0107-02 **文献标识码**:A **中图分类号**:TP309

1 引言

RC4 算法是流密码算法的一种,自 1987 年该算法被设计出来之后,得到了广泛的应用,其安全性分析也成为一个重要的研究课题。RC4 算法中 PRGA 的初始状态对于安全性具有重要的作用,如果初始状态中的部分取值泄漏,攻击者可以利用已知的输出值将初始状态计算出来,并能够预测以后的输出值,而这种攻击方法的复杂度决定于初始状态的取值情况。本文从 PRGA 的初始状态出发,假定已知初始状态中的一部分值,在 Knudsen 等人提出的猜测赋值分析方法的基础上,对破译 RC4 算法的复杂度进行进一步研究。

2 RC4 算法

2.1 符号介绍

首先进行符号说明。n 表示 RC4 算法中使用的一个字节的长度,N 表示长度为 n 的一个字节能够显示的值的总量,即 $N=2^n$,文中使用 n=8 bit 比特的情况,那么 N 为 256。S 表示该算法的内部状态,每一个 S 中有 N 个 n 比特长度的值。t 表示一个参数,t=0,1,2,…。S_t 表示在参数 t 时的内部状态,i_t 和 j_t 表示参数 t 时对应的两个指针,他们指向内部状态 S_t 中的两个值,$S_t[i_t]$ 表示 S_t 中指针指向的值。Z_t 表示每一个 t 对应的伪随机数生成器的输出值。

2.2 RC4 算法

该算法包含两个部分,密钥方案算法 KSA(图 1)和伪随机数生成算法 PRGA(图 2)。

```
KSA(K)
Initialization:
  S←⟨0,1,…,N-1⟩
  j←0
Scrambling:
  For i←0…N-1
    j←j+S[i]+K[i mod ℓ]
    S[i]↔S[j]
```

图 1 KSA 算法

```
PRGA(S)
Initialization:
  i←0
  j←0
Generation loop:
  i←i+1
  j←j+S[i]
  S[i]↔S[j]
  Output z←S[S[i]+S[j]]
```

图 2 PRGA 算法

2.3 RC4 算法的研究背景

目前的分析方法主要针对 KSA 和 PRGA 而进行的。KSA 方案存在一些弱密钥[1-2],同时该算法的 PRGA 也存在一定的线性统计弱点[3-4]。另一个主要的分析方法是通过计算初始状态来攻击 PRGA,Knudsen 等人提出了猜测赋值的分析方法[5],利用这个方法,Mantin 给出复杂度的计算结果[6]。然而,内部状态中有连续的一部分取值已知的情形下,破译 RC4 算法的复杂度

作者简介:王信敏(1979-),男,讲师,主要研究领域为电子商务安全与身份认证;郑世慧(1979-),女,博士后,主要研究领域为对称密码算法分析与设计。
收稿日期:2008-01-22 **修回日期**:2008-04-02

会比已知相同数量的分散取值状态下要低很多,Knudsen 等人只是给出了部分实验结果,并未进行理论分析,本文针对这个问题,对这种情形进行了理论分析,提出了有效的计算方法。

2.4 猜测赋值分析方法

使用 a_t 表示 t 时的内部状态中被赋过的值的总量,a 表示初始内部状态中已被赋过的值的总量。分析方法如下:

步骤 1 检查 $S_{t-1}[i_t]$是否已被赋值过:

(1)如果有值,跳到步骤 2。

(2)否则,用 2^n-a_t 个未出现的值逐一对 $S_{t-1}[i_t]$赋值,更新 a_t,然后跳到步骤 2。

步骤 2 检查 Z_t 是否等于 a_t 中某个值,即是否在内部状态已知的值中:

(1)如果等于,可以计算出 $S[j_t]$。如果此时不出现矛盾,t+1,跳到步骤 1。

(2)如果不等于,跳到步骤 3。

步骤 3 检查 $S_{t-1}[j_t]$是否已经被赋值过:

如果没有,逐一对 $S_{t-1}[j_t]$赋值,更新 a_t。接着检查是否出现矛盾。如果没有矛盾,t+1,然后跳到步骤 1。

假设用 $c_1(\cdot)$、$c_2(\cdot)$、$c_3(\cdot)$来表示这 3 步的复杂度,那么复杂度公式如下:

$$c_1(a)=\frac{a}{2^n}c_2(a)+(1-\frac{a}{2^n})(2^n-a)c_2(a+1) \quad (1)$$

$$c_2(a)=\frac{a}{2^n}\left((1-\frac{a}{2^n})^2(1+c_1(a+1))+\frac{1}{2^n}c_1(a)\right)+(1-\frac{a}{2^n})c_3(a) \quad (2)$$

$$c_3(a)=(1-\frac{a}{2^n})\left(f(a)+\frac{2a+1}{2^n}c_1(a+1)+(2^n-a)e(a)c_1(a+2)\right) \quad (3)$$

其中,

$$e(a)=(1-\frac{a+1}{2^n})(1-\frac{1}{2^n-a}) \quad f(a)=(2^n-a)(1+e(a))+\frac{a}{2^n}$$

这些复杂度是在随机状态(a 个已知的值分布随机)下的平均复杂度。

3 PRGA 的初始状态及复杂度分析

3.1 PRGA 的一个典型状态

定义 1 如果初始状态中已知连续 a 个位置的值是已知的,并且指针 i_t 正好指向这 a 个位置中的某一个,把这种内部状态定义为典型状态,为了便于下面的描述,指针 i_t 指向第一个赋值的位置,如图 3 所示。

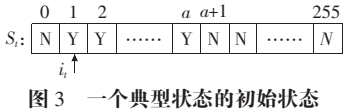

图 3 一个典型状态的初始状态

其中 N 表示该位置值未知,Y 表示值已知

3.2 复杂度分析

不妨假设初始指针 i_t 刚好落在第一个位置,此时 t=1,j_t 已知,指针 i_t 从 t=1 开始,到碰到第一个未赋值的位置为止(如果当 i_t 到达 t=a 时,t=a+1 位置是一个未赋值的位置,那么就到 t=a 结束这个过程,否则继续进行下去,直到碰到未赋值的位置),在这个阶段,每个 i_t 位置都是已赋值的,并且每个值都是正确值,然后当由 $S_{t-1}[i_t]$、$S_{t-1}[j_t]$和 Z_t 可以推出某一个未赋值位置的正确值时,对新位置赋值,依次分析下去。这样,当到达第一个未赋值位置时,可以得到一些新的赋值,并且都是正确值。然后再把整个过程返回到 t=1。

此时,已赋值位置的数量为刚才所计算出的新值的数量 $\Delta_1 a$ 与原来已知值的数量 a 之和,前 a 个位置仍为原来已知的值。重复上面的过程,这样又可以得到一部分新值 $\Delta_2 a$,而且都是正确值,最后,可以得到由这 a 个连续值所产生的所有新值,假设总共得到了 Δa 个新值。

当这个阶段结束时,内部状态中仍有部分未赋值位置,再实施猜测赋值的方法,对未知位置逐一赋值,这样,整个过程的复杂度等价于初始内部状态中有 a+Δa 个值时的复杂度。由此得出:

推论 1 典型状态下的复杂度⇔某个内部状态的复杂度,即等价于已知 a+Δa 个值时初始状态对应的复杂度。

引理 1 i_t 从 t=1 开始,到第一个未赋值位置这个阶段,在经过的每一处,可以推出一个新的正确的值的概率为 $\Pr=\frac{a_t}{2^{n-1}}\cdot(1-\frac{a_t}{2^n})$,其中 a_t 表示参数 t 时内部状态中已知值的总数。

证明 指针 i_t 指向的每一个位置,能够推出一个正确的新值,必须使得 Z_t 处在 a_t 个值之中,j_t 出现在这 a_t 个值之外,或者 Z_t 不在 a_t 个值之中,j_t 出现在 a_t 个值之中。Z_t 处在 a_t 个值之中的概率为 $a_t/2^n$,不在其中的概率为 $1-a_t/2^n$,j_t 出现在这 a_t 个值之外的概率为 $1-a_t/2^n$,在其中的概率为 $a_t/2^n$。这样

$$\Pr=\frac{a_t}{2^n}\cdot(1-\frac{a_t}{2^n})+\frac{a_t}{2^n}\cdot(1-\frac{a_t}{2^n})=\frac{a_t}{2^{n-1}}\cdot(1-\frac{a_t}{2^n})$$

假设第一阶段获得的新值为 $\Delta_1 a$ 个。重复这个过程,又可以得到新值 $\Delta_2 a$ 个,在这个过程中每个位置 t 可以得到一个新值的概率为 $\frac{\Delta_1 a}{2^{n-1}}(1-\frac{a_t}{2^n})$(求法同引理 1)。依次进行下去,直到能够得到所有的新值,此时 $\Delta a=\Delta_1 a+\Delta_2 a+\cdots$。这样可以求出 a+Δa 的平均值,部分数据见表 1。

表 1 典型状态下的部分结果

N=256					
a	a+Δa	复杂度	a	a+Δa	复杂度
20	22.881	2^{620}	70	110.200	2^{200}
25	**29.406**	2^{581}	75	**121.189**	2^{163}
30	37.456	2^{533}	80	**131.183**	2^{132}
35	**45.139**	2^{487}	85	**141.033**	2^{105}
40	53.192	2^{445}	90	**150.663**	2^{79}
45	**61.584**	2^{399}	95	**160.006**	2^{57}
50	71.573	2^{352}	100	**169.006**	2^{37}
55	80.924	2^{312}	105	**177.618**	2^{23}
60	90.523	2^{271}	100	185.807	2^{11}
65	**100.306**	2^{236}			

得到了典型状态下的等价状态之后,就可以来分析其复杂度了。重新从 t=1 开始,当指针 i_t 到达第一个未赋值状态之前,公式(1)~(3)变为:

$$c_1(a)=(1-\frac{a}{2^n})^2\cdot\left((1-\frac{a+1}{2^n})\cdot\frac{1}{2^n-a}\cdot c_1(a+1)+(2^n-a)e(a)(1+c_1(a+2))\right)$$

因为所有的 $S_{t-1}[i_t]$都是已知的,只有 $S_{t-1}[j_t]$和 Z_t 同时出现在未被赋值位置时才有可能出现逐一赋值的情况。然后 i_t 继续往下进行,当 $S_{t-1}[i_t]$未知时,假定此时内部状态中已知的值是随机分布的(显然这种假定是合理的),然后使用猜测赋值的分析方法中(1)~(3)进行复杂度计算,所以此时的平均复杂度应该分两段来求,并非已知内部状态中 a+Δa 个随机分布的值所对应的平均复杂度。值得注意的是,此处求得的复杂度是一个平均

(下转 128 页)