对两个基于混沌的序列密码算法的分析

金晨辉, 高海英
(解放军信息工程大学电子技术学院, 河南郑州 450004)

摘 要: 本文指出/ 混沌非线性反馈密码序列的理论设计和有限精度实现0 和/ 一类新型混沌反馈密码序列的理论设计0 两文基于混沌设计的两个序列密码算法产生的乱数序列的前若干值对密钥的变化并不敏感, 据此在已知混沌变换的条件下, 可以利用已知明文采取先攻击高位密钥再攻击低位密钥的方法对这两个密码算法进行分割攻击. 本文还提出了在正确密钥的分布已知条件下使平均计算复杂性达到最小的穷举攻击算法, 并将它与分割攻击方法结合, 提出了对上述两个密码算法的优化分割攻击方案, 并分析了这两个攻击方案的计算复杂性和成功率.

关键词: 混沌序列; 序列密码; 分割攻击; 穷举攻击; 优化的分割攻击
中图分类号: TN918 文献标识码: A 文章编号: 0372-2112 (2004) 07-1066-05

Analysis of Two Stream Ciphers Based on Chaos

JIN Chen-hui, GAO Hai-ying
(Electronic Technology Institute, PLA Information Engineering University, Zhengzhou, Henan 450004, China)

Abstract: It is pointed out that the two stream ciphers proposed respectively in / Generating nonlinear feedback stream ciphers via chaotic systems0 and in / The theoretical design for a class of new chaotic feedback stream ciphers0 have the property that the frontal values of the output sequences of the two stream ciphers are not sensitive to the least significant bits of the key. So they can be attacked effectively by the divide-and-conquer attack, which attacks the most significant bits of key at first and the least significant bits of key secondly, with the known chaotic transformations and known plaintexts. An optimal exhaustive attack with the minimum computing complexity is proposed under the condition that the distribution of the key is known. Furthermore, an improved divide-and-conquer attack is proposed and its computing complexity and success probability is analyzed.

Key words: chaotic sequence; stream cipher; divide-and-conquer attack; exhaustive attacking; improved divide-and-conquer attack

1 引言

近年来, 混沌序列是个热门的研究课题, 混沌序列在通信领域的应用得到了广泛的研究. 人们还研究了混沌序列在保密通信中的应用问题, 并基于混沌序列提出了多个数据加密方案和数据水印方案. 文献[1]基于分段线性的混沌函数提出了一个序列密码算法(以下简称为 ZLL 加密算法), 文献[2]基于分段非线性函数提出了另一个序列密码算法(以下简称为 SWY 加密算法). 经我们分析发现, 尽管当迭代足够次数后, 混沌序列的值对初值十分敏感, 并呈现混沌现象, 但其前几个输出值却对初始值的低位比特的变化并不敏感. 利用这个特性我们就可利用分割攻击方法, 在仅知道 ZLL 加密算法和 SWY 加密算法的一个不长的输出乱数序列的条件下, 求出该加密算法的密钥.

2 对 ZLL 序列密码算法的已知乱数攻击

本节我们将分析文献[1]基于混沌序列提出的 ZLL 序列密码算法的弱点, 并针对该弱点提出对该加密算法的分割攻击算法, 并分析该分割攻击算法的性能.

2.1 ZLL 序列密码算法简介

ZLL 序列密码算法中有两个密码变换, 其中一个是混沌映射, 另一个是量化函数.

设 p 是自然数, 实数 $a_0, a_1, \cdots, a_{p+1}$ 满足 $0=a_0<a_1<a_2<\cdots<a_p<a_{p+1}=1$, 则 ZLL 序列密码算法中的混沌映射 $f: (-1,1) \to (-1,1)$ 定义为:

$$f(x)=\begin{cases} -1+2(x-a_j)/(a_{j+1}-a_j), & \text{若 } x\in[a_j, a_{j+1}),\ j=0,1,\cdots,p \\ f(-x), & \text{若 } x<0 \end{cases} \tag{1}$$

再设 n 是自然数, $I_0, I_1, \cdots, I_{2^n-1}$ 是 $(-1,1)$ 的 2^n 个连续的左闭右开的等分区间, 即 $I_i=[\frac{i}{2^{n-1}}-1, \frac{i+1}{2^{n-1}}-1]$, 则 ZLL 序列密码算法的量化函数 $T: [-1,1) \to \{0,1\}$ 定义为

$$T(x)=\begin{cases} 0, & \text{若 } x\in \bigcup_{k=0}^{2^{n-1}-1} I_{2k} \\ 1, & \text{若 } x\in \bigcup_{k=0}^{2^{n-1}-1} I_{2k+1} \end{cases} \tag{2}$$

设 $x=\sum_{i=1}^{\infty} x_i/2^i$ 是一个非负小数且诸 $x_i\in\{0,1\}$, 则称 $\sum_{i=1}^{m} x_i/2^i$ 是 x 的 m 精度小数. 显然, m 维二元向量 $(x_1, x_2, \cdots,$

收稿日期: 2002-06-28; 修回日期: 2004-01-20
基金项目: 河南省杰出青年科学基金(No. 0312001800)

x_m) 与 m 精度小数 $\sum_{i=1}^{m} x_i/2^i$ 一一对应. 以下本文总是将 m 维二元向量与 m 精度非负小数之间按上述方式相互转换, 且称 $(x_1, x_2, \cdots, x_n)$ 为 $x=\sum_{i=1}^{1} x_i/2^i$ 的高 n 位比特.

在选定密码算法的参数 p, n 和 $a_0, a_1, \cdots, a_{p+1}$ 之后, 就可构造出 ZLL 序列密码算法:

设 $k=(k_1, k_2, \cdots, k_m)$ 是 ZLL 密码算法的密钥, 它对应的 m 精度小数记为 x_0, 则可按递归关系

$$x_i = f(x_{i-1}),\ i \geq 1 \tag{3}$$

得到一个混沌序列 $\{x_i\}_{i=1}^{\infty}$. ZLL 密码算法根据 $s_i = T(x_i)$ 产生一条二元序列 $\{s_i\}_{i=1}^{\infty}$, 这条序列, 我们称为乱数序列. 此后, ZLL 算法再将该二元序列与明文序列 $\{m_i\}_{i=1}^{\infty}$ 逐位模 2 加, 得到加密后的密文序列.

因此, 对 ZLL 算法的已知明文攻击就是对该算法的已知乱数攻击.

在 ZLL 算法的具体实现时, 也可采用 m 序列扰动方式实现[1], 即按递归关系 $x_i = f(x_{i-1}) \oplus y_i,\ i \geq 1$ 产生混沌序列 $\{x_i\}_{i=1}^{\infty}$. 其中 $\{y_i\}_{i=1}^{\infty}$ 是高若干位由 m 序列构成, 低位全部为 0 的线性递归序列. 实际上, ZLL 算法的参数 $a_1, a_2, \cdots, a_p$ 也可选为密钥因素使用, 但为简单起见, 在下面的分析中, 我们将在混沌变换的初态是唯一的密钥因素的假设下进行分析, 因而将忽略扰动结构的影响, 仅分析对混沌序列的初态的还原问题.

2.2　ZLL 序列密码算法的信息泄漏规律

以下用 Int(x) 表示不大于实数 x 的最大整数.

首先考查当密钥有较小变化时, 由(1)定义的混沌映射的输出的变化情况.

定理 1　设 f(x) 是 ZLL 算法中由(1)定义的混沌映射, $x, x+\varepsilon \in [a_j, a_{j+1})$, 则有

$$|f(x+\varepsilon) - f(x)| \leq \frac{2|\varepsilon|}{a_{j+1} - a_j}$$

当将密钥 k 的低位比特全设置为 0 时, 密钥 k 将变为 $k-\varepsilon$ 且 $0 \leq \varepsilon < 1$. 定理 1 说明, 只要密钥的这种改变 ε 不大, 能够使 k 和 $k-\varepsilon$ 仍落在同一小区间 $[a_j, a_{j+1})$ 内, 混沌映射 f(x) 的输出的变化也会很小. 接着分析 f(x) 的较小变化会对 ZLL 算法的输出乱数产生什么影响.

定理 2　设 $n > 1$ 是 ZLL 算法选定的参数, 则 ZLL 算法中的量化函数 $T(x) = \mathrm{Int}(2^{n-1}x) \bmod 2$.

推论　设 $n > 1$ 是 ZLL 算法选定的参数, $x \in (-1, 1)$, 如果

$$2^{1-n}\mathrm{Int}(2^{n-1}x) - x \leq \varepsilon < 2^{1-n} + 2^{1-n}\mathrm{Int}(2^{n-1}x) - x \tag{4}$$

则有 $T(x+\varepsilon) = T(x)$.

定理 2 的推论说明, 当量化函数的输入有轻微变化时, ZLL 算法输出的乱数不变. 因此, 结合定理 1 和定理 2 就知, 当密钥的低位比特发生较小的变化 ε 时, 量化函数的输入的变化 $f(k) - f(k-\varepsilon)$ 也会很小, 因而 ZLL 算法输出的第一个乱数一般不变, 这说明 ZLL 算法输出的第一个乱数对密钥的低位比特变化并不敏感.

如果密钥 x_0 的低位比特发生的变化 ε 很小, 则由(3)得到的混沌序列 $\{x_i\}_{i=1}^{\infty}$ 的前若干值的变化也很小, 因而也将导致 ZLL 算法的前若干输出信号并不发生改变, 这就说明, ZLL 算法的开头几个信号只与密钥的高位比特有关, 而与密钥的低位比特关系不大. 试验数据也验证了上述分析结论.

定义 1　设 k 和 k_c 分别是某序列密码算法的正确密钥和试验密钥, $\{s_i\}_{i=1}^{\infty}$ 和 $\{sc_i\}_{i=1}^{\infty}$ 分别是它们产生的乱数序列, 则称 $\max\{n: P\,i:1 \leq i \leq n,\ 有\ sc_i = s_i\}$ 为 k_c 的吻合度.

以下 $k^{(m)}$ 均表示将正确密钥 k 的高 m 比特保持不变, 其它比特全部置 0 所得的实验密钥, 且将 $k^{(m)}$ 的吻合度记为 n_m. 要想从理论上推出吻合度的分布规律是非常困难的, 因此, 下面利用模拟试验的方法, 考查 ZLL 算法的 n_m 的分布规律. 尽管我们只是随便选取一组参数进行模拟试验, 但无论哪组参数, 其吻合度都有相近的统计规律, 因而都具有一定的信息泄漏.

取密钥为 64 比特, 参数 $n=6$, $p=2$, $a_1 = 0x0f0f\ 0f0f\ 0f0f\ 0f0f$, $a_2 = 0xf1f1\ f1f1\ f1f1\ f1f1$. 再随机选取 1 万个 64 比特密钥作为一组, 统计结果表明, n_{32}, n_{48}, n_{56} 的分布规律为

n_{32}	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
个数	7	28	158	337	946	1307	2169	1713	1743	815	376	201	99	47	28	14

n_{48}	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	\	26
个数	5	18	60	294	417	1064	1099	1723	1584	1444	998	628	323	179	92	32	21		19

n_{56}	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	\	28
个数	6	14	49	111	433	556	1203	1152	1902	1296	1369	826	495	255	139	71	38		85

n_{32}, n_{48}, n_{56} 的期望分别为 10.17, 16.18 和 19.15, 且

$p(n_{32} \geq 6) = 0.9965$, $p(n_{48} \geq 13) = 0.9623$, $p(n_{56} \geq 16) = 0.9387$

上述试验结果(简记为(*))为我们设计分割攻击算法以及分析该分割攻击算法的效率提供了依据.

2.3　对 ZLL 序列密码算法的分割攻击算法

定理 3　设 k 是序列密码算法的密钥, 则在除高 m 比特外全为 0 的所有试验密钥中, 吻合度 $\geq t$ 的试验密钥的个数的期望为 2^{m-t}, 且吻合度 $\geq t$ 的试验密钥中包含 $k^{(m)}$ 的概率为 $p(n_m \geq t)$.

当 $k^{(m)}$ 产生的乱数与正确密钥产生的乱数相互独立时, 近似有 $p(n_m \geq t) = 2^{-t}$, 此时不能从 n_m 的分布率中得到 $k^{(m)}$ 的信息; 但当 $p(n_m \geq t)$ 显著大于 2^{-t} 时, 就可据此将 $k^{(m)}$ 与吻合度 $\geq t$ 且除高 m 比特外全为 0 的其它试验密钥区分开来. 这就是对混沌密码进行分割攻击的基本思想和依据.

由于我们已利用模拟试验的方法得到了 n_m 的分布规律, 下面研究如何充分利用这个有利条件, 设计出效率更高的穷举攻击方案.

引理 1　设 $p_1 \geq p_2 \geq \cdots \geq p_m > 0$ 且 $q_1, q_2, \cdots, q_m$ 是 $p_1, p_2, \cdots, p_m$ 的一个排列, 则当且仅当 $q_i = p_i$ 对诸 i 都成立时, $\sum_{i=1}^{m} iq_i$ 达到最小值 $\sum_{i=1}^{m} ip_i$.

引理 1 说明在正确密钥的分布率已知的条件下, 先检验正确率较大的那些试验密钥是合理的.

定理 4　设密钥空间可被划分为 m 个不交的集合 $A_1, A_2, \cdots, A_m$, 且正确密钥落入集合 A_r 中的概率为 p_r, 如果加密算法

没有等效密钥, 且 A_r 中各点是正确密钥的概率都相同, 则依次检验集合 $A_1, A_2, \cdots, A_m$ 中点的穷举攻击找到正确密钥时检验过的密钥的个数平均为

$$\frac{1}{2}+\sum_{r=1}^{m}p_r\left(\sum_{j=1}^{r-1}|A_j|+\frac{1}{2}|A_r|\right)$$

命题1 设 n_{32} 的概率分布符合(*), 记 A_i 是使 $n_{32}=i$ 的候选密钥全体, 则 A_i 中的各点构成正确密钥的概率 p 满足

i	4	5	6	7	8	9	10	11	\ 12
$2^{26}p$	0l 00035	0l 00028	0l 00316	0l 01348	0l 07568	2l 09	6l 94	10l 96	26l 6

证明 由于随机密钥的吻合度为 4 的概率为 2^{-5}, 故在 2^{32} 个试验密钥中, 吻合度为 4 的个数 $|A_4|$ 近似为 $2^{32}/2^5=2^{27}$. 再由试验结果(*)知, $k^{(32)}$ 的吻合度为 4 的概率为 0l 0007, 故集合 A_4 中各点是正确密钥的概率平均为 $0l\,0007/2^{27}=2^{-26}\approx 0l\,00035$. 类似可验证其它数据.

设正确密钥 $k=(k_{32}, k_{48}, k_{56}, k_{64})$ 的左边是高位比特, 且 k_{32} 和 k_{48} 分别是 32 和 16 比特块, k_{56}, k_{64} 都是 8 比特块. 再设正确密钥和试验密钥产生的乱数序列分别为 $\{s_i\}_{i=1}^{\infty}$ 和 $\{sc_i\}_{i=1}^{\infty}$, 则可设计对 ZLL 算法的优化的分割攻击算法为:

算法 1

Stepl 取 $d_1=13$, $d_2=22$. 赋初值 $k_{32}=k_{48}=k_{56}=k_{64}=0$.

Step2 攻击 $k^{(32)}$: 以 k_{32} 为初值产生乱数序列 $\{sc_i\}_{i=1}^{32}$. 如果 $sc_1=s_1, \cdots, sc_{12}=s_{12}$, 则将该 k_{32} 写入 $File_0$; 如果存在 $6<t<12$, 使 $sc_1=s_1, \cdots, sc_t=s_t$ 但 $sc_{t+1}\neq s_{t+1}$, 则将该 k_{32} 写入文件 $File_{12-t}$. 如果 $sc_1=s_1, \cdots, sc_6=s_6$ 不全成立, 则将 k_{32} 增 1, 并在 $k_{32}<2^{32}$ 时返回 Step2, 在 $k_{32}=2^{32}$ 时执行 Step3.

Step3 依次读出 $File_0, File_1, \cdots, File_6$ 中的 k_{32}, 并执行 Step4. 如果其中的 k_{32} 已全部读出, 则将 d_2 减 1, 并在 $d_2=15$ 时算法终止, 在 $d_2>15$ 时返回 Step3, 同时将此后的执行 Step5 都改为执行 Step5c.

Step4 攻击 $k^{(48)}$. 如果 $k_{48}=2^{16}$, 则令 $k_{48}=0$ 并返回 Step3 读出下个 k_{32}. 否则以 (k_{32}, k_{48}) 为初值产生乱数序列 $\{sc_i\}_{i=1}^{d_1}$. 如果 $\{sc_i\}_{i=1}^{d_1}$ 与 $\{s_i\}_{i=1}^{d_1}$ 全部吻合, 则执行 Step5. 否则将 k_{48} 增 1 后返回 Step4.

Step5 攻击 $k^{(56)}$. 如果 $k_{56}=2^8$, 则令 $k_{56}=0$ 并将 k_{48} 增 1 后返回 Step4. 否则, 以 (k_{32}, k_{48}, k_{56}) 为初值产生乱数序列 $\{sc_i\}_{i=1}^{d_2}$, 如果 $\{sc_i\}_{i=1}^{d_2}$ 与 $\{s_i\}_{i=1}^{d_2}$ 全部吻合, 执行 Step6. 否则将 k_{56} 增 1 后返回 Step5.

Step6 攻击 $k^{(64)}$. 如果 $k_{64}=2^8$, 则令 $k_{64}=0$ 并将 k_{56} 增 1 后返回 Step5. 否则以 $(k_{32}, k_{48}, k_{56}, k_{64})$ 为初值产生乱数序列 $\{sc_i\}_{i=1}^{80}$. 如果 $\{sc_i\}_{i=1}^{80}$ 与 $\{s_i\}_{i=1}^{80}$ 全部吻合, 则输出候选密钥 $(k_{32}, k_{48}, k_{56}, k_{64})$, 算法终止. 否则将 k_{64} 增 1 后返回 Step6.

Step5c 攻击 $k^{(56)}$. 如果 $k_{56}=2^8$, 则令 $k_{56}=0$ 并将 k_{48} 增 1 后返回 Step4. 否则, 以 (k_{32}, k_{48}, k_{56}) 为初值产生乱数序列 $\{sc_i\}_{i=1}^{d_2+1}$, 如果 $\{sc_i\}_{i=1}^{d_2}$ 与 $\{s_i\}_{i=1}^{d_2}$ 全部吻合且 $sc_{d_2+1}\neq s_{d_2+1}$, 执行 Step6. 否则将 k_{56} 增 1 后返回 Step5c.

备注:

(1)借助 $k^{(m)}$ 的分布率已知这个条件进行算法 1 的优化设计主要体现在使用了两个技巧: 一个是在 Step2 中对 $k^{(32)}$ 的候选值依其正确率的大小进行排序, 另一个是在 Step6 中对 $k^{(56)}$ 的候选值依其正确率的大小进行检验, 但其平均计算复杂性的降低主要是通过第二个技巧完成的. 第一个技巧只是使得在 $k^{(56)}$ 的吻合度 <22 时, 我们只需读出已攻击出的 $k^{(32)}$ 而不必再重新对 $k^{(32)}$ 进行攻击.

(2) Step2 中的文件 $File_0, \cdots, File_6$ 的规模依次约为 $2^{20}, 2^{20}, 2^{21}, \cdots, 2^{25}$ 个 32 位字, 故该算法的存储量约为 2^{26} 个 32 位字. 将 Step2 和 Step5 均改为选取固定值 d, 且仅对吻合度为 d 的那些可能密钥做进一步检验, 就是文献[3]采取的分割攻击策略. 由引理 1 知, 这种方式没有算法 1 的效率高.

(3)我们也可在攻击 $k^{(32)}$ 之前先攻击 $k^{(16)}$. 但先攻击 $k^{(16)}$ 只是降低了 Step2 的计算复杂性, 由后面的定理 5 的分析知道, 算法 1 的计算复杂性主要集中在 Step5 和 Step6, 与之相比 Step2 的计算复杂性可忽略不计, 因而我们没有采取先攻击 $k^{(16)}$ 再攻击 $k^{(32)}$ 的策略.

定理 5 算法 1 的成功率为 0l 9, 平均计算复杂性为 2^{45}.

证明 该攻击算法不漏掉正确密钥等价于对 $k^{(32)}, k^{(48)}, k^{(56)}$ 的攻击都不会漏掉正确的值. 由(*)知, 该算法不漏掉正确的 $k^{(32)}, k^{(48)}, k^{(56)}$ 的概率分别为 0l 9965, 0l 9623, 0l 9387, 故该攻击算法不漏掉正确密钥的概率为

$$p=0l\,9965\times 0l\,9623\times 0l\,9387>0l\,9$$

先分析 Step2 的执行次数. 由于 Step2 对 2^{32} 个可能的 k_{32} 都进行检验, 因而 Step2 的计算复杂性为 2^{32}.

再分析 Step4 的平均计算复杂性.

由于吻合度 ≥ 13 的 (k_{32}, k_{48}) 的个数约为 $2^{48-13}=2^{35}$, 故对固定的 d_2, 至多需检验 2^{35} 个 (k_{32}, k_{48}). 又由试验结果(*)知, $k^{(56)}$ 的吻合度 ≥ 22 的概率为 0l 1909, $k^{(56)}$ 的吻合度为 21, 20, 19, $\cdots$, 16 的概率分别为 0l 1369, 0l 1296, 0l 1902, 0l 1152, 0l 1203, 0l 0556, 故由定理 4 知算法 1 找到 $k^{(48)}$ 时检验过的 (k_{32}, k_{48}) 的个数平均为

$$0l\,5+0l\,1909\times 0l\,5\times 2^{35}+0l\,1369\times 1l\,5\times 2^{35}+0l\,1296\times 2l\,5\times 2^{35}+0l\,1902\times 3l\,5\times 2^{35}+0l\,1152\times 4l\,5\times 2^{35}+0l\,1203\times 5l\,5\times 2^{35}+0l\,0556\times 6l\,5\times 2^{35}>1l\,416\times 2^{36}$$

即 Step4 的平均执行次数为 $1l\,416\times 2^{36}$.

最后分析 Step5 和 Step6 的计算复杂性.

由试验结果(*)知, $k^{(56)}$ 的吻合度 ≥ 22 的概率为 0l 1909, $k^{(56)}$ 的吻合度为 21, 20, 19, $\cdots$, 16 的概率分别为 0l 1369, 0l 1296, 0l 1902, 0l 1152, 0l 1203, 0l 0556, 且 (k_{32}, k_{48}, k_{56}) 的吻合度 ≥ 22 以及为 21, 20, 19, $\cdots$, 16 的个数分别为 $2^{56-22}, 2^{56-22}, 2^{56-21}, 2^{56-20}, \cdots, 2^{56-16-1}$, 故由定理 4 知, 算法 1 找到 $k^{(56)}$ 时检验过的 (k_{32}, k_{48}, k_{56}) 的平均个数为

$$1/2+0l\,1909\times 2^{34}/2+0l\,1369\times(2^{34}+2^{34}/2)+0l\,1296\times(2^{35}+2^{35}/2)+0l\,1902\times(2^{36}+2^{36}/2)+0l\,1152\times(2^{37}+2^{37}/2)+0l\,1203\times(2^{38}+2^{38}/2)+0l\,0556\times(2^{39}+2^{39}/2)>1l\,1\times 2^{37}$$

即 Step5 和 Step5c 的执行次数平均为 $1l\,1\times 2^{37}$. 又因对于检验过的每个 (k_{32}, k_{48}, k_{56}), 都需对 k_{64} 的每个可能进行检验, 故 Step6 的执行次数平均为 $1l\,1\times 2^{37}\times 2^8=1l\,1\times 2^{45}$. 这说明算法

1 的平均计算复杂性近似为 2^{45}. 证毕.

算法 1 将攻击 64 比特的 ZZL 算法的平均计算复杂性由设计的 2^{63} 降为 2^{45}, 减少了 18 比特的密钥熵, 从而导致该算法在现有的计算能力下是可破的.

如果在对 $k^{(56)}$ 的攻击中, 不采取算法 1 中的优化的攻击方法, 而是采取文献[3]中的方法(没有 Step6 和 Step8), 选择一个固定的 d_2 进行攻击. 则当选取 d_2 = 18 时, 由于吻合度 E 18 的(k_{32}, k_{48}, k_{56})的个数约为 2^{56-18}, 因而平均的计算复杂性为 $2^{64-18}/2 = 2^{45}$, 但其不漏掉 $k^{(56)}$ 的概率只有 $p(n_{56}$ E 18) = 0.5719; 当选取 d_2 = 16, 虽然不漏掉 $k^{(56)}$ 的概率仍为 $p(n_{56}$ E 16) = 0.9287, 但其平均计算复杂性却升至 $2^{64-16}/2 = 2^{47} > 1.1 @ 2^{45}$, 因而本文提出的优化的分割攻击方法可以显著地提高攻击算法的效率.

3　对 SWY 序列密码算法的已知乱数攻击

3.1　SWY 序列密码算法简介

SWY 序列密码算法的结构与 ZLL 算法相同, 也分为混沌映射和量化函数两步, 二者的区别仅在于它们所使用的混沌映射不同. SWY 算法使用的混沌变换是所谓的/ 逐段二次方根0 映射 $g: (-1, 1) \to (-1, 1)$:

$$g(x) = \begin{cases} \frac{1}{a_i}\left[\sqrt{\frac{4a_i(x - c_i)}{c_{i+1} - c_i} + (1 - a_i)^2} - 1\right], & \text{若 } x \in [c_i, c_{i+1}),\ i = 0, 1, \ldots, N-1 \\ 1, & \text{若 } x = 1 \\ g(-x), & \text{其它} \end{cases}$$

其中自然数 N E 2, 实数 $a_0, a_1, \ldots, a_{N-1}$ 和实数 $c_0, c_1, \ldots, c_N$ 都是 SWY 算法设定的参数, 它们满足:

(1) $0 = c_0 < c_1 < c_2 < \cdots < c_{N-1} < c_N = 1$;

(2) P i: 0F i< N, $a_i \in (-1, 1) \setminus \{0\}$, 都有 $\sum_{i=0}^{N-1} (c_{i+1} - c_i) a_i = 0$.

SWY 算法同样可采取 m 序列扰动的方法实现, 其参数 $a_1, a_2, \ldots, a_p$ 也可选为密钥因素使用, 但为简单起见, 在下面的分析中, 我们仍在混沌变换的初态是唯一的密钥因素的假设下进行分析, 因而将忽略扰动结构的影响, 仅分析对混沌序列的初态的还原问题.

3.2　SWY 序列密码算法的信息泄漏规律分析

尽管 SWY 算法将 ZLL 算法中的逐段线性函数改为逐段非线性函数, 但其混沌变换的输出的高位比特对输入的低位比特的不敏感性仍然存在且比 ZLL 算法更加突出, 因而仍然可以对 SWY 算法实施分割攻击.

定理 6　设 $g(x)$ 是 SWY 算法中的混沌变换, x, x+ E $\in [c_i, c_{i+1})$, 则有

$$|g(x + E) - g(x)| < \frac{2|E|}{(c_{i+1} - c_i)(1 - |a_i|)}$$

由定理 6 可以看出, SWY 算法使用的混沌变换 $g(x)$ 仍然具有输出的高位比特对输入的低位比特的变化不敏感这个特性, 因而可以利用分割攻击方法对 SWY 算法实施攻击. 下面是对具有 40 比特密钥的 SWY 算法进行的一些试验数据.

取密钥为 40 比特密钥, 即 m= 40. 再取 n= 5, N= 2, a_1 = 0x64fa 9638 06, a_2= - 0xe90e 13218e, c_0= 0, c_1= 0xb29c 622d 08, 随机选取 1 万个 40 比特密钥作为一组, 统计结果表明, n_{16}, n_{32} 的分布规律为

n_{16}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
个数	3	16	82	250	841	1897	2638	2012	1097	597	275	151	85	25	16	9	5	1

n_{32}	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	\ 25
个数	2	10	45	174	431	973	1790	2193	1900	1162	646	336	149	77	31	17	9	55

且 n_{16}, n_{32} 的期望值分别是 6.41 和 15.48, 同时

$p(n_{16}$ E 2) = 0.9981, $p(n_{16}$ E 3) = 0.9899, $p(n_{16}$ E 4) = 0.9649, $p(n_{32}$ E 11) = 0.9943

从上述试验数据看, SWY 算法的信息泄漏比 ZZY 算法的信息泄漏更大, 因而更容易被攻破. 但由于涉及多精度整数的开方运算的编程, 我们仅对 40 比特精度的 SWY 算法进行了试验.

上述试验结果为我们对 SWY 算法进行分割攻击以及对分割攻击算法的分析提供了依据.

设正确密钥 k= (k_{16}, k_{32}, k_{40}), 再设正确密钥和试验密钥产生的乱数序列分别为 $\{s_i\}_{i=1}^{\infty}$ 和 $\{sc_i\}_{i=1}^{\infty}$, 则可设计对 SWY 算法的分割攻击算法为:

算法 2

Step1　取 d= 18. 赋初值 $k_{32} = k_{40} = 0$.

Step2　攻击 $k^{(16)}$: 对每个可能的 k_{16}, 以 k_{16} 为初值产生乱数序列 $\{sc_i\}_{i=1}^{7}$. 如果 $sc_1 = s_1, \ldots, sc_7 = s_7$, 则将 k_{16} 标识为 0; 否则, 如果存在 2F t< 7, 使 $sc_1 = s_1, \ldots, sc_t = s_t$ 但 $sc_{t+1} \neq s_{t+1}$, 则将该 k_{16} 标识为 7- t.

Step3　对所有已标识的 k_{16} 依标识的大小排序并存储, 小者在前, 大者在后. 并按该次序, 调出 k_{16} 执行 Step4. 如果所有标识过的 k_{16} 都检验完毕仍未找到正确密钥, 则将 d 减 1, 并在 d= 10 时, 算法终止, 在 d> 10 时依次读出标识 0, 1, 2, 3, 4, 5 的 k_{16} 并执行 Step4c.

Step4　攻击 $k^{(32)}$. 如果 $k_{32} = 2^{16}$, 则令 k_{32}= 0 并返回 Step3 读出下个 k_{16}. 否则, 对每个可能的 k_{32}, 以(k_{16}, k_{32})为初值产生乱数序列 $\{sc_i\}_{i=1}^{d}$, 如果 $\{sc_i\}_{i=1}^{d}$ 与 $\{s_i\}_{i=1}^{d}$ 全部吻合, 则执行 Step5, 否则将 k_{32} 增 1 后返回 Step4.

Step5　攻击 $k^{(40)}$. 如果 $k_{40} = 2^8$, 则令 k_{40}= 0 并将 k_{32} 增 1 后返回 Step4. 否则, 以(k_{16}, k_{32}, k_{40})为初值产生乱数序列 $\{sc_i\}_{i=1}^{60}$. 如果 $\{sc_i\}_{i=1}^{60}$ 与 $\{s_i\}_{i=1}^{60}$ 全部吻合, 则输出候选密钥(k_{16}, k_{32}, k_{40}), 算法终止, 否则将 k_{40} 增 1 后返回 Step5.

Step4c　攻击 $k^{(32)}$. 如果 $k_{32} = 2^{16}$, 则令 k_{32}= 0 并返回 Step3 读出下个 k_{16}. 否则, 以(k_{16}, k_{32})为初值产生乱数序列 $\{sc_i\}_{i=1}^{d+1}$, 当 $\{sc_i\}_{i=1}^{d} = \{s_i\}_{i=1}^{d}$ 全部吻合且 $sc_{d+1} \neq s_{d+1}$ 时, 执行 Step5, 否则将 k_{32} 增 1 后返回 Step4c.

定理 7　算法 2 的成功率为 0.9924, 平均计算复杂性为 $1.295 @ 2^{25}$.

证明　由于算法 2 不漏掉正确密钥等价于对 $k^{(16)}$ 和

$k^{(32)}$ 的攻击都不会漏掉正确的值. 由试验结果知, $k^{(16)}$ 和 $k^{(32)}$ 不被漏掉的概率分别为 0.9981 和 0.9943, 因此该攻击算法不漏掉正确密钥的概率为

$$p = 0.9981 \times 0.9943 = 0.9924$$

由于 $k^{(32)}$ 的吻合度 E 18 的概率为 0.132, 吻合度为 17, 16, 15, 14, 13, 12, 11 的概率依次为

0.1162, 0.19, 0.2193, 0.179, 0.0973, 0.0431, 0.0174

且 (k_{16}, k_{32}) 的吻合度 E 18 以及为 17, 16, 15, 14, 13, 12, 11 的个数近似为 2^{32-18}, 2^{32-18}, 2^{32-17}, , , 2^{32-10}, 故由定理 4 知, 算法 2 找到 $k^{(32)}$ 时检验过的 (k_{16}, k_{32}) 的个数平均为

$$\frac{1}{2} + 0.132 \times \frac{2^{14}}{2} + 0.1162 \times (2^{14} + \frac{2^{14}}{2}) + 0.19 \times (2^{15} + \frac{2^{15}}{2}) + 0.2193 \times (2^{16} + \frac{2^{16}}{2}) + 0.179 \times (2^{17} + \frac{2^{17}}{2}) + 0.0973 \times (2^{18} + \frac{2^{18}}{2}) + 0.0431 \times (2^{19} + \frac{2^{19}}{2}) + 0.0174 \times (2^{20} + \frac{2^{20}}{2}) \cup 1.1295 \times 2^{17}$$

即 Step4 和 Step4c 的执行次数平均为 1.1295×2^{17}. 又因对于检验过的每个 (k_{16}, k_{32}), 都需对 k_{40} 的每个可能进行检验, 故 Step5 的执行次数平均为 $1.1295 \times 2^{17} \times 2^{8} = 1.1295 \times 2^{25}$. 显然算法 2 攻击 $k^{(16)}$ 的计算复杂性 $< 2^{16}$, 因而算法 2 的平均计算复杂度近似为 1.1295×2^{25}.

利用定理 4 和实验结果 (*) 容易算出在应用算法 2 攻击 40 比特密钥的 ZLL 算法的 $k^{(32)}$ 时, 平均计算复杂性为 $1.12 \times 2^{30} > 1.1295 \times 2^{25}$, 因而 SWY 算法比 ZLL 算法更容易攻破.

4 进一步的说明

前面我们指出 ZZL 算法和 SWY 算法所使用的混沌变换具有输出对输入的微小变化不并敏感这个弱点, 并据此提出了对 64 比特的 ZZL 算法和 40 比特的 SWY 算法的一种分割攻击方法. 该攻击方法利用了这两个密码算法的输出序列的前若干值对初态的低位的变化并不敏感这个信息泄漏. 一个自然的想法是丢弃输出序列的初始若干信号. 对于丢弃前 t 个信号的修改方法, 我们可采取先攻击出混沌变换的第 t 时刻的状态. 此时已相当于求出了该加密算法的一个等效密钥. 如果非要求出初态不可, 则可由该状态利用类似的分割攻击方法求出第 t - 1 的状态, 再依次倒推出其初始状态. 鉴于篇幅有限, 我们将在另文中具体研究这种方法.

当将混沌变换的某些参数也设置为密钥时, 由于混沌变换仍是由四则运算和幂方运算复合而成的函数, 仍具有输入低位的变化对输出高位的变化影响很小的弱点, 因而该文的方法仍可实施. 此时, 我们可采取先攻击所有密钥的高位, 再攻击密钥的低位的策略进行分割攻击. 对于该攻击方法的效果, 我们也将在另文具体分析.

我们的攻击算法是在忽略 m 序列扰动结构条件下进行的. 对于具有 m 序列扰动结构的混沌密码算法, 如果扰动结构由密钥决定, 我们可采取先假设出决定其扰动结构的那部分密钥, 再对扰动结构的每个假设, 采取本文的方法进行攻击. 显然, 在扰动结构的密钥与混沌变换的密钥相互独立时, 该攻击方法所降低的密钥熵与攻击不具有扰动结构的混沌密码算法所降低的密钥熵相同.

对于设计好的序列密码, 诸 $p(k^{(m)} = t)$ 应近似为 2^{-t-1}. $p(k^{(m)} = t)$ 越接近 2^{-t-1}, 越不易求出 $k^{(m)}$. 显然, 将混沌变换连续迭代若干次后再应用量化函数输出乱数的修改方案将有助于改善吻合度的统计规律, 但这种修改同时也将成倍地增加数据加密时的计算量.

5 结论

在本文中, 针对正确密钥的分布已知的穷举攻击模型, 我们提出了使平均计算复杂性达到最小的穷举攻击算法; 给出了文献[1]和文献[2]基于混沌变换设计的两个序列密码算法的信息泄漏规律, 即这两个序列密码的前若干输出信号主要由密钥的高位比特决定; 应用这个信息泄漏, 并结合正确密钥的分布已知的穷举攻击模型, 设计了对这两个序列密码算法的优化的分割攻击算法, 并分析了攻击算法的计算复杂性. 这种优化的分割攻击算法比文献[3]提出的分割攻击算法具有更高的效率. 结论表明, 当混沌变换的初态是唯一的密钥因素时, 攻击 64 比特的 ZLL 算法的计算复杂性约为 1.11×2^{45}, 成功率为 0.9, 攻击 40 比特的 SWY 算法的计算复杂性约为 1.1295×2^{25}, 成功率为 0.9924, 因而此时的 ZLL 算法和 SWY 算法都是不安全的. 本文的结果说明, 尽管混沌序列具有遍历性和多次迭代后的信号对初值的敏感性, 但其初值的高位输出比特对低位输入比特的变换并不十分敏感, 如果设计不当, 基于混沌原理设计的加密算法仍有被破译的危险.

参考文献:

[1] 周红, 罗杰, 凌燮亭. 混沌非线性反馈密码序列的理论设计和有限精度实现[J]. 电子学报, 1997, 25(10): 57-60.

[2] 桑涛, 王汝笠, 严义埙. 一类新型混沌反馈密码序列的理论设计[J]. 电子学报, 1999, 27(7): 47-50.

[3] 金晨辉. 一个基于混沌的分组密码算法的分析[J]. 中国工程科学, 2001, 3(6): 75-80.

作者简介:

金晨辉 男, 1965 年 3 月出生于河南扶沟, 1988 年 7 月在北京师范学院获硕士学位, 2000 年 12 月在解放军信息工程大学获博士学位, 现为解放军信息工程大学电子技术学院教授、博士生导师, 主要研究方向为密码学和信息安全. E2mail: Jinchenhui@126.com

高海英 女, 1978 年 7 月出生于河南沈丘, 2003 年 7 月在解放军信息工程大学获硕士学位, 现为解放军信息工程大学电子技术学院助教、北京邮电大学博士生, 主要研究方向为密码学.