RC4／arcfour の調査

2002 年 2 月

白石 善明

RC4/arcfourの調査

2002年2月

1 はじめに

RC4/arcfour(以下 RC4 と呼ぶ) は 1987 年に RSA 社の Ron Rivest により開発された可変長な秘密鍵を利用できるストリーム暗号である．RC4 は，SSL(Secure Socket Layer) プロトコルにも暗号化アルゴリズムの一つとして組み込まれ，また WEP(Wired Equevalent Privacy) プロトコルにも組み込まれている．

WEP においてはユーザの秘密鍵と初期化ベクトル (IV) を結合させたものをセッション鍵として利用している．最近，文献 [1] で似通ったセッション鍵を使えば RC4 の出力系列は似通ったものとなる性質を利用して，128 ビットの秘密鍵が推定されるという報告がされている．そして文献 [2] で，WEP を使っている実際の無線 LAN 環境でスニファ (パケットキャプチャプログラム) を動作させて，文献 [1] の攻撃法を利用し，盗聴したパケットから秘密鍵を容易に推定できたと報告されている．RC4 では，状態表 ($n = 8$ の場合 256 個の値を格納する表) を最初に初期化する鍵スケジューリング部が 256 個の値をそれぞれ一度ずつ置換して撹拌するという単純な操作のために，新規の IV が直前に使われた IV と似通っていれば出力系列の最初の部分が似通ってしまう．文献 [1, 2] の結果は，IV の運用方法を留意していなかった WEP プロトコル上の欠陥から導かれたものであると言える．逐次カウントアップされていくような毎回似通った IV を使うのであれば，その IV を MD5 などのハッシュ関数を通したハッシュ値を利用して秘密鍵とあわせてセッション鍵とすれば解決するものであると文献 [1, 2] でも指摘され，RSA 社からも同様のコメントが公表されている [3, 4]．これらの文献で，IV をハッシュ関数を通して利用すればこの攻撃は適用不可能で，SSL ではそのような処理を行っているために，SSL プロトコルには WEP プロトコルのようなプロトコル上の欠陥はないと報告されている．

そこで，本報告ではプロトコル上の欠陥についてではなく，暗号アルゴリズムとしての安全性について述べる．

2 RC4/arcfour

2.1 記号の定義

n ： アルゴリズムで一度に処理する bits 数 (ワード長)．$n = 8$ が使われることが多い．

t ：時間変数．

S_t ：時間 t に置ける状態表．n[bits] のバイナリの値が 2^n 行 (全パターン) あるという表になっている．

$S_t(l)$ ：状態表 S_t の l 行目の値．$S_t = (S_t(l))_{l=0}^{2^n-1}$．

i_t, j_t ：状態表の行を指すポインタ変数.

Z_t ：時刻 t における出力 (n[bits] の疑似乱数).

Z ：RC4 の出力系列．$Z = (Z_t)_{t=1}^{\infty}$.

$X_t^{(k)}$ ：ある系列 X の時間 t の時の k 番目の要素の数.

2.2 暗号化

ここでは RC4 の暗号化手法を示す．初期値 $i_0, j_0 = 0$ とし，初期状態 S_0 は秘密鍵を用いてシャッフル (置換) されているとする．暗号化自体は出力系列 Z と平文を XOR するのみであるため，出力系列のアルゴリズム部を示す.

RC4 の出力は以下の方法で得る．ここでの加算は $\mathrm{mod} 2^n$ とする.

Step1 $i_t = i_{t-1} + 1$

Step2 $j_t = j_{t-1} + S_{t-1}(i_t)$

Step3 $S_t(i_t) = S_{t-1}(j_t),\ S_t(j_t) = S_{t-1}(i_t)$

Step4 $Z_t = S_t(S_t(i_t) + S_t(j_t))$

3 RC4/arcfour に対する攻撃

3.1 統計的な偏りを利用した攻撃 [5]

Golić により，LCSA(Linear sequential circuit approximation) の手法を用いて RC4 の出力系列から統計上の弱点を検出する方法が示されている．本節ではその攻撃法について述べる.

3.1.1 LCSA による近似のための仮定

- S_t の要素は全ての t で一様に分布し，balanced(0 と 1 が同数である) である.
- i_t は決定論的で既知のものである.
- j_t は一様分布に従う ($t > 1$).

3.1.2 LCSA による近似

LSCA の手法による出力系列の近似を行う際に次の 2 つの近似を用いる．

[**近似 1(確率が高い近似)**]

S_t は S_{t-1} によって近似ができる．

$$S_t \approx S_{t-1}$$

[**近似 2(線形近似)**]

$S_{t-1}^{(k)}$ はその入力変数の任意の線形関数によって近似できる．しかし，全体的な相関係数を最大にするような線形関数で近似する．この論文では線形近似式は以下ものを使用している．

$$S_{t-1}^{(k)}(X+Y) \approx X^{(k)} + Y^{(k)}$$

まず，RC4 の暗号化部で生成される出力 Z_t は

$$\begin{aligned} Z_t &= S_t(S_t(i_t) + S_t(j_t)) \\ &= S_t(S_{t-1}(j_t) + S_{t-1}(i_t)) \\ &= S_t(S_{t-1}(i_t) + S_{t-1}(j_t)) \\ &= S_t(S_{t-1}(i_{t-1}+1) + S_{t-1}(j_{t-1} + S_{t-1}(i_{t-1}+1))) \end{aligned}$$

と変形できる．つまり S_t を S_{t-1} を使って記述する．近似 1 より，

$$Z_t \approx S_{t-1}(S_{t-1}(i_{t-1}+1) + S_{t-1}(j_{t-1} + S_{t-1}(i_{t-1}+1)))$$

という近似式が得られる．

この関係の k bits 目について考える．

$$Z_t^{(k)} \approx S_{t-1}^{(k)}(S_{t-1}(i_{t-1}+1) + S_{t-1}(j_{t-1} + S_{t-1}(i_{t-1}+1)))$$

まず次のような線形近似を行い，

$$Z_t^{(k)} \approx S_{t-1}^{(k)}(i_{t-1}+1) + S_{t-1}^{(k)}(j_{t-1} + S_{t-1}(i_{t-1}+1))$$

次に，求まった近似に再度線形近似を行う．

$$Z_t^{(k)} \approx S_{t-1}^{(k)}(i_{t-1}+1) + j_{t-1}^{(k)} + S_{t-1}^{(k)}(i_{t-1}+1)$$

ビット単位の加算は XOR であるため，$S_{t-1}^{(k)}(i_{t-1}+1)$ を消去できるので，

$$Z_t^{(k)} \approx j_{t-1}^{(k)}$$

となり，線形近似を 2 回行うことで出力を入力 $j_{t-1}^{(k)}$ で近似できることになる．

j_{t-1} は一様分布に従うポインタ変数であり値を特定できない．ここで，微分を行えば異なる時間の j_t が 2 つ手に入るので i_t に近似できる．そこで，特定できる値すなわち i_t で近似を行うために出力系列の微分を行う．

一階二進微分は，$\dot{Z}_t^{(k)} = Z_t^{(k)} + Z_{t+1}^{(k)}$ と表されるため，これを近似すると先程と同様にして

$$Z_t^{(k)} + Z_{t+1}^{(k)} \approx j_{t-1}^{(k)} + j_t^{(k)}$$

が得られる．これを近似すると

$$j_{t-1}^{(k)} + j_t^{(k)} = j_{t-1}^{(k)} + (j_{t-1}^{(k)} + S_{t-1}^{(k)}(i_t))$$

右辺の $j_{t-1}^{(k)}$ を消去できるので，次のようになる．

$$j_{t-1}^{(k)} + j_t^{(k)} = S_{t-1}^{(k)}(i_t)$$

この式に対して，線形近似を行う．

$$j_{t-1}^{(k)} + j_t^{(k)} \approx i_t^{(k)}$$

すなわち，

$$Z_t^{(k)} + Z_{t+1}^{(k)} \approx j_{t-1}^{(k)} + j_t^{(k)} \approx i_t^{(k)}$$

となり，出力の一階二進微分を $i_t^{(k)}$ で近似できる．ここまでで，線形近似の回数は合計 5 回となっている．このように出力系列を $i_t^{(k)}$ で近似できたが，線形近似回数の合計が奇数回の時は中心積率 (相関係数を求めるときの式の一部分) が 0 になり，相関係数が打ち消されてしまうためにこの式は用いることができない．

そこで，次の条件を満たす階数の微分により近似を行う．

- 線形近似はあまり高確率で成り立たないためできるだけ線形近似回数は少ないほうが良い．
- 相関係数が打ち消されないように線形近似回数は偶数でなければならない．

ここで h 階微分を考える．h 階微分の場合，$Z_t^{(k)} + Z_{t+h}^{(k)}$ と表され，4 回の線形近似で

$$Z_t^{(k)} + Z_{t+h}^{(k)} \approx j_{t-1}^{(k)} + j_{t+h-1}^{(k)}$$

が得られる．$j_{t-1}^{(k)}$ を消去するために，$j_{t+h-1}^{(k)}$ を次のように式変形する．

$$\begin{aligned} j_{t-1}^{(k)} + j_{t+h-1}^{(k)} = j_{t-1}^{(k)} + (j_{t-1}^{(k)} + S_{t-1}^{(k)}(i_t) + S_t^{(k)}(i_{t+1}) \\ + S_{t+1}^{(k)}(i_{t+2}) \dots S_{t+h-2}^{(k)}(i_{t+h-1})) \end{aligned}$$

このときの線形近似の回数は h 回となるため，h 階微分の線形近似回数は $4+h$ 回であることが分かる．偶数の線形近似回数になる h は 2 であるため，ここでは出力の 2 階微分の近似を用いる．その結果,

$$\ddot{Z}_t^{(k)} = Z_t^{(k)} + Z_{t+2}^{(k)} \approx j_{t-1}^{(k)} + j_{t+1}^{(k)} \approx i_t^{(k)} + i_{t+1}^{(k)}$$

で近似する．i_t と i_{t+1} の和を考えると最下位ビットが常に 1 になることがわかる．

参考までに他の $bits$ について考えると，0 になる確率は下位 $k+1$ ビット目では $1-2^{-k}$ となる．最上位ビットが比較的高い確率で近似できるものの，必ず成り立つ最下位ビットの近似よりは相関係数が小さくなる．したがって，出力の二階二進微分の最下位ビットと 1 との相関を見るようにする．

3.1.3 相関係数の導出

出力系列と二階二進微分をした線形近似式との相関係数 c を求める概要を示す．
相関係数 c を求めるために次のことを行う．

- S_t, S_{t+1}, S_{t+2} のそれぞれについて h 階微分を利用した線形近似が何回行われているかを調べ，全体の相関係数を各々の相関係数の積で表す．
- 各状態で線形近似したときの相関をまとめて考えるために，状態間の近似の相関係数 (S_t と S_{t+1} 間の相関係数) を導入し，S_t に線形近似したときの相関係数を使う．
- 全体の相関係数は，上記でまとめられた相関係数の期待値の積となる．
- 状態間の近似の相関係数の期待値は 2^n が大きいとき “1” となり，考慮しなくても良い．

全体の相関係数 c とは，S_t を線形近似したときの相関係数 c_f とする．文献中の定理 2(balanced で一様にランダムなブール関数と線形ブール関数の相関係数を表したもの) の証明過程で求まった c_f を必要な分だけ累乗して，その期待値 (平均値) を求めると,

$$c = E(c_f^6) = 2^{-6(n-2)}\mu_6$$

と表すことができる．μ_6 は 6 次の中心積率を示し，文献中の $Appendix$ より,

$$\mu_6 = \sum_{k=0}^{2^{n-1}} (k - 2^{n-2})^6 \frac{\binom{2^{n-1}}{k}^2}{\binom{2^n}{2^{n-1}}} \sim 15 \times 2^{3(n-4)}$$

のようになる．したがって,

$$c = 2^{-6(n-2)} \times (15 \times 2^{3(n-4)}) = 15 \times 2^{-3n}$$

となる．ここで $O(c^{-2})$ の長さの出力系列からその統計的偏りを高確率で発見できる [6] ことから

$$L \approx \frac{2^{6n}}{225} \approx 2^{6n-7.814}$$

となり，約 2^{6n-8} の出力系列があれば二階二進微分した出力系列の線形近似式と出力系列の最下位ビットの間に相関係数が $c = 15 \times 2^{-3n}$ の相関がみられることになる．

3.1.4 統計的な偏りを利用した攻撃のまとめ

文献 [5] では，二進微分を利用して線形近似式を導出している．その結果，次のような統計的偏りを利用した攻撃が可能であることを示している．

- 出力系列の二階二進微分をした線形近似式を使うことで，出力系列の最下位ビットが “1” となる相関係数が $c = 15 \times 2^{-3n}$ となる．したがって，出力系列を観測してこのような相関が見られたならば，その乱数は RC4 アルゴリズムであると識別できる．この識別に必要な出力系列の長さは 2^{6n-8} である．
- 相関係数 c を逐次求めていくことで，ワード長 n が推定できる．

3.2 乱数系列から状態表の初期状態を推定する攻撃 [7]

3.2.1 swap の無い RC4 への攻撃

この攻撃は，あらかじめ状態 S 中のいくつかの値を部分的に推測しておいて，鍵ストリーム Z_t の観測により残りの行に値を割り当てて行く方法である．

RC4 では次の 4 つの変数 $j_t, S(i_t), S(j_t), S^{-1}(Z_t)$ が未知変数となる．この中の 3 つの値が既知となれば 4 つ目の値が決まり，4 つの値が既知であれば矛盾が生じていないかを確認できる．つまり，矛盾があればその探索を終了すればよい．各 4 つの変数についてそれぞれ 3 つの値から推測するには，次のようにすればよい．

1. $S^{-1}(Z_t), j_t, S(i_t)$ が既知のとき

   $$S(j_t) = S^{-1}(Z_t) - S(i_t)$$

   により $S(j_t)$ が求まる．

2. $S(i_t), S(j_t), j_t$ が既知のとき

   $$S^{-1}(Z_t) = S(i_t) + S(j_t)$$

   により $S^{-1}(Z_t)$ が求まる．

3. $S(i_t), S(j_t), S^{-1}(Z_t)$ が既知のとき

$$j_t = S^{-1}(S^{-1}(Z_t) - S(i_t))$$

により j_t が求まる.

4. $j_t, S(j_t), S^{-1}(Z_t)$ が既知のとき

$$S(i_t) = S^{-1}(Z_t) - S(j_t)$$

により $S(i_t)$ が求まる.

[**攻撃法**]

1. 状態 S の初めの v 個の値を推測する.

2. i_t, j_t の初期値 i_0, j_0 は 0 であり既知である.

3. 初めの v step
初めの v step の間は $S(i_t)$ がいつも既知であるので，$S(i_t)$ から計算できる j_t もいつも既知になる．したがって残りの 2 つの未知変数のうち 1 つが既知となればもう 1 つの値を決定でき，状態表に値を割り当てられる.

   - Z_t が既に割り当てられているとき ($= S^{-1}(Z_t)$ が既知のとき)
   $\rightarrow S(j_t)$ を決定できる.
   - j_t の指す行に既に値が割り当てられているとき ($= S(j_t)$ が既知のとき)
   $\rightarrow S^{-1}(Z_t)$ を決定でき，Z_t を割り当てる.
   - $S^{-1}(Z_t)$ が既知 であり，$S(j_t)$ も既知であったとき (4 つの未知変数が既知のとき)
   $\rightarrow$ 矛盾がないか調べる．矛盾があれば探索を終了し，異なる推測値をセットして実行し直す.

4. $v+1$ step 目以降
$v+1$ step 目以降になると，あらかじめ推測していた値以外を i_t が指すようになる．そのとき $S(i_t)$ は未知になる．$S(i_t)$ が未知になると，当然 $S(i_t)$ から計算する j_t も未知になる．さらに j_t は j_{t-1} の値から計算していたので，j_t を他の 3 つの変数から導き出せるまでは他の値の推定ができない.

   [j_t **の値が不明なとき**]

   i_t の指しているところに値があるとき ($S(i_t)$ が既知)，Z_t の値が割り当てられているかを調べる．もし割り当てられていたら ($S^{-1}(Z_t)$ が既知),

   $$S^{-1}(Z_t) - S(i_t) = S(j_t)$$

   を計算し，それが表の中に割り当てられていないかを調べる．割り当てられていたなら j_t の値がわかる (j_t が既知になる).

[j_t **が既知のとき**]

残りの 3 つの未知変数のうち 2 つが既知であれば，残りの未知変数を決定できる．v step 目までとの違いは $S(i_t)$ も推定する変数に入っており場合分けが一つ増えることである．

- i_t が指すところに値があれば ($S(i_t)$ が既知ならば)，v step 目までに行った方法で $S(j_t), S^{-1}(Z_t)$ を推定できる．
- $S(i_t)$ が未知であるとき (推定に失敗すると j_t が不明となるとき) 表に Z_t の値があり，かつ j_t に値が割り当てられていれば $S(i_t)$ を推定できる．
- 3 つの未知変数が全て既知になれば，それらの関係を使って矛盾が無いかを調べる．もし矛盾があれば異なる推測値で攻撃処理をやり直し，矛盾がなければそのまま処理を続ける．

5. 4. の操作を繰り返し，全ての行に矛盾なく値が割り当てられたなら初期状態 S を推定できる．

なお，どの程度の出力系列があればこの攻撃が成功するのかは文献中に記述されていなかった．

3.2.2 swap 頻度が少ない RC4 への攻撃

前節の攻撃をもとにした swap 頻度の少ない RC4 への攻撃について説明する．

[**攻撃の概略**]
s 回毎に swap が起こるとする．最初の s ステップは swap なしの攻撃と同じアルゴリズムを実行するが，s ステップを越えた後，j_t の値が未知になると $S(i_t)$ の swap 先が分からなくなる．この際 $S(i_t)$ の値を未知のものにして処理を続けるのだが，j_t が既知の行を指していて状態表が誤ってしまうこともありえる．これを防ぐために $S_t(i_t), S_t(j_t), S_t^{-1}(Z_t)$ の中のいくらかを状態表から除く．このときに正しい値を除くことを出来るだけ少なくするように工夫をする (工夫の仕方は記述されていない)．

[**結果**]
128 回に 1 回だけ swap するような swap 頻度が少ないときは，あらかじめ割り当てておく正しい値は 256 個中 40 個で成功確率は 50%となる．しかし，2 回に 1 回 swap が起こるような場合では，256 個中 240 個の値をあらかじめ割り当てても成功確率は 50%程度となり，この攻撃法は現実的には使えない．

3.2.3 フルバージョン RC4 への攻撃

前節までの攻撃では，あらかじめ正しい値を事前にいくつか推測しておく必要があったが，本節ではその推測が不要な攻撃方法について述べる．

[**概略**]
攻撃アルゴリズムは，値を割り当てていない初期状態 S_0 から始める．

時刻 $t = 1, 2, 3, \ldots, m$ において，$S_{t-1}(i_t)$ または $S_{t-1}(j_t)$ に値が割り当てられていないとき，

1. $S_{t-1}(i_t)$ に $(0 \leq v \leq (2^n - 1))$ の範囲の値 v を選ぶ．
2. j_t を $S_{t-1}(i_t)$ から計算し，その後 $S_{t-1}(j_t)$ に $(0 \leq v \leq (2^n - 1))$ の範囲の値 v を選ぶ

このように順番に値を状態表に割り当てていく．つまり初期状態の全数探索を行うのであるが，次に述べる値を割り当てる方法や，ある条件での矛盾を考慮しながら探索するようにして探索回数を抑える．

1)：値の割り当てかた

状態表 S の中の値は $0 \leq v \leq (2 - n - 1)$ が一度ずつ入っており，それが置換されているだけである．したがって，新しく選ぶ $S_{t-1}(i_t), S_{t-1}(j_t)$ はそれぞれ，状態表中に既に割り当てられている値を割り当てないようにする．

2)：Z_t の値がまだ S_t 内に割り当てられていないとき

$i_s = S_t(i_t) + S_t(j_t)$ を計算し，i_s が指す行に何らかの値が割り当てられているという矛盾が生じていないかを確かめる．

- 割り当てられている (矛盾している) 場合
  探索に矛盾が生じていることがわかるため，これ以上の探索を行わない．以前の処理まで戻り，$S_t(i_t)$ や $S_t(j_t)$ に異なる値を割り当てて試す．
- 割り当てられていない場合
  矛盾が生じていないため，$S_t(i_s) = Z_t$ と値を割り当てて次の処理に進む．

3)：Z_t の値が既に S_t 内に割り当てられているとき

$i_s = S_t(i_t) + S_t(j_t)$ を計算し，i_s が指す行と $S^{-1}(Z_t)$ の行が異なるという矛盾が生じていないかを確かめる．

- i_s と $S^{-1}(Z_t)$ が等しい場合
  矛盾が生じていないことが分かる．このとき $S_t(j_t)$ が一意に定まる．
- i_s と $S^{-1}(Z_t)$ が等しくない (矛盾している) 場合
  探索に矛盾が生じている．ここでは全ての $S_t(j_t)$ を試し，矛盾が生じていないものがないかを確かめる．それでも矛盾が生じてしまうなら，$S_t(i_t)$ に異なる値を割り当てる．

Z_t を観測するたびに 2) と 3) の条件のどちらかを実行できる．よってこの条件により矛盾を検出し無駄な探索を減らすことができる．

筆者らはこれらの原理を使って RC4 の暗号化を行う 4 ステップを順番に実行しながら，状態表に値を当てはめていく方法を与えている．

[**実験結果**]
この攻撃によって出力系列からの状態表の復元は，状態表の全数探索の平方根程度の探索回数で済む．これは数値実験でも確認されていた．しかし，表 1 のように，$n > 5$ では計算量が大きすぎて実行することができないため，$n = 8$ で使われる通常の RC4 に対する現実的な攻撃法とは言えないと記述されている．

表 1: n-bit RC4 の攻撃の計算量

n	complexity
3	2^8
4	2^{21}
5	2^{53}
6	2^{132}
7	2^{324}
8	2^{779}

n が既知であり，かつ $n \leq 4$ であるときにはこの攻撃は有効であると言える．

3.2.4 乱数系列から状態表の初期状態を推定する攻撃のまとめ

文献 [7] では，4 つの未知変数 $j_t, S(i_t), S(j_t), S^{-1}(Z_t)$ を逐次推定していくことで，出力系列から状態表の初期状態を復元する攻撃法が示されていた．その結果，次のような攻撃が可能であることを示している．

- swap 頻度が少ないバージョンの RC4(n=8) を考えたとき，(1)swap 頻度が 128 回に 1 回のときは初期状態の 256 個中 40 個の値が既知のときに成功確率が 50%程度，(2)swap 頻度が 2 回に 1 回のように高頻度になると 256 個中 240 個の値が既知でも成功確率は 50%程度，となる．

- フルバージョンの n ビット RC4 の出力系列から未知変数に制限を付けながら初期状態の表に割り当てていく方法では，計算量が状態数 2^n の全数探索の平方根程度，つまり約 $\sqrt{2^{2^n}}$ 程度で初期状態が推定できる．

2 点目からフルバージョン RC4 の初期状態を推定する攻撃は，$n \geq 5$ では計算量的に攻撃は不可能となる．

4 まとめ

本報告書では，RC4 に対する暗号化アルゴリズムの安全性について説明した．統計的偏りを利用した RC4 アルゴリズムと特定する識別攻撃は可能であるが，出力系列から初期状態を推定する攻撃はいずれも $n \geq 5$ では不可能なために，現在利用されている $n = 8$ の RC4 に対して実用的な攻撃法は存在しないと言える．

WEP では IV の不注意な取扱いによるプロトコル上の脆弱性が明らかにされたが，RC4 の弱点が発見されたのではない．SSL では WEP プロトコルのようなプロトコル上の問題点は現在のところ発見されていない．

参考文献

[1] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," Eighth Annual Workshop on Selected Areas in Cryptography, Aug. 2001.

[2] A. Stubblefield, J. Ioannidis, and A.D. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP," AT&T Labs Technical Report TD-4ZCPZZ, available at *http://www.cs.rice.edu/ astubble/wep/wep_attack.html*, Aug. 2001.

[3] RSA Laboratories Tech Notes, "RSA Security Response to Weaknesses in Key Scheduling Algorithm of RC4," available at *http://www.rsasecurity.com/rsalabs/technotes/wep.html*, Sep. 2001.

[4] RSA Laboratories Tech Notes, "WEP Fix using RC4 Fast Packet Keying," available at *http://www.rsasecurity.com/rsalabs/technotes/wep-fix.html*, Dec. 2001.

[5] Jovan Dj. Golić, "Linear Statistical Weakness of Alleged RC4 Keystream Generator," Proceedings of EUROCRYPT'97, *Lecture Notes in Computer Science*, vol.1233, W. Fumy ed., pp.226–238, 1997.

[6] Jovan Dj. Golić, "Linear models for keystream generators," *IEEE Trans. Computers*, vol.C-45, pp.41–49, Jan. 1996.

[7] L. Kundsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege, "Analysis methods for (alleged) RC4," Advances in Cryptology - ASIACRYPT'98, *Lecture Notes in Computer Science*, vol. 1514, K. Ohta and D. Pei eds., Springer-Verlag, pp. 327-341, 1998.