暗号アルゴリズムの詳細評価

報告書

疑似乱数生成

TOYOCRYPT-HR1 **編**

1. 概要

テスト項目の概要とテスト結果の概要を示します。

統計的性質に関する評価(2 章)、及び同等出力系列発生条件の評価(3 章)を合格できないため、評価者は提案方式を不合格と判断します。

1.1. 統計的性質に関する評価について

テストデータを生成し、下記のテスト項目に対するテストプログラムを作成後、評価を行いました。テストデータは特殊なデータの生成とランダムサンプリングを併用しました。

もちろん、当該項目に対するテストに合格したとしても、その項目に対する安全性を保障(証明)するものではありません。

1.1.1. 0/1 等頻度性テスト

提案方式は、本テストに合格したと判断します。詳細は、別冊(疑似乱数生成の評価 0/1 等頻度性テスト TOYOCRYPT-HR1 編)を参照してください。

1.1.2. 連性テスト

提案方式は、本テストに合格したと判断します。詳細は、別冊(疑似乱数生成の評価 連性テスト TOYOCRYPT-HR1 編)を参照してください。

1.1.3. 長周期連性テスト

提案方式は、本テストに合格しませんでした。詳細は、別冊(疑似乱数生成の評価 長周期連性テスト TOYOCRYPT-HR1 編)を参照してください。ある特殊な鍵を与えた場合、FIPS 140 の条件を満たさない長さの gaps(0 bits が連続して出力される)が発生します。

1.1.4. 一様性テスト

提案方式は、本テストに合格したと判断します。詳細は、別冊(疑似乱数生成の評価 一様性テスト TOYOCRYPT-HR1 編)を参照してください。

1.1.5. Avalanche 性テスト

提案方式は、本テストに合格したと判断します。詳細は、別冊(疑似乱数生成の評価 Avalanche 性テスト TOYOCRYPT-HR1 編)を参照してください。

1.1.6. 線形複雑度テスト

提案方式は、本テストに合格しませんでした。詳細は、別冊(疑似乱数生成の評価 線形複雑度テスト TOYOCRYPT-HR1 編)を参照してください。ある特殊な鍵を与えた場合、線形複雑度が理論どおり上昇しない場合があります。

1.2. 用途に対する適合性

次の各項目について、机上もしくはマシンテストにより評価を行いました。

1.2.1. 推測可能性評価について

入力の一部が露呈したとき、その入力に対する出力系列の推定が可能であるかは、 Avalanche テストによりある程度の評価が可能です。更に机上考察の結果を合わせ、提案方式は本テストに合格したと判断します。詳細は 3 章を参照してください。

1.2.2. 周期(出力系列の再現性)の評価について

同一鍵を使用した場合、周期が設計値より小さくなることはないか、机上考察を行い、提案方式は本テストに合格したと判断します。詳細は、3 章を参照してください。

1.2.3. 同等出力系列発生条件の評価について

異なる鍵を使用して、同等（まったく同一ではない）出力系列が発生する条件があるか机上考察を行い、提案方式は、本テストに合格しませんでした。詳細は、3 章を参照してください。ある特殊なストリーム鍵に対しては、同等出力系列を容易に発生させることが可能です。

1.3. 出力系列に対する入力空間の大きさについて

上記「同等出力系列発生条件の評価」で示した理由及び運用上の理由で、入力空間（鍵空間）は設計値よりも小さいと考えます。詳細は 4 章を参照してください。

1.4. ユーザの立場からの評価について

仕様書を用いて実装する状況を想定して、評価を行いました。詳細は 5 章を参照してください。

1.5. 暗号解析の立場からの評価について

統計的性質を利用するのではなく、アルゴリズム構造から、解析を行う状況を想定して、評価を行いました。詳細は 6 章を参照してください。

1.6. ドキュメントについて

ドキュメントの内容について、評価を行いました。詳細は 7 章を参照してください。

1.7. 性能について

ソフトウエア実装した際の性能について、評価を行いました。詳細は 8 章を参照してください。

1.8. その他

最近 NIST が乱数に関する更に詳細な評価基準を公開しました。 http://csrc.nist.gov/rng/ に詳細が記載されていますが、今回行った評価(0/1 等頻度性、一様性、連性など)以外にも、評価項目が増えています。

今回はスケジュールの都合で間に合いませんでしたが、今後、疑似乱数生成に関する評価は、この基準に従って行うべきと考えます。

2. 統計的性質に関する評価について

テスト項目に合格しなかった「長周期連性テスト」および「線形複雑度」テストについて言及します。詳細は、各別冊を参照してください。

この両項目に合格しなかった理由は、ある特殊な鍵を固定鍵 C およびストリーム鍵 S に選んだ場合、出力系列のある部分に 0 が連続する（gaps が発生する）ことにあります。このため、「長周期連性テスト」に合格しませんし、出力系列に 0 が続く限り線形複雑度も上昇せず、期待値（傾き 1/2 の直線）と、計測した線形複雑度との差が大きくなります。

そのような鍵（固定鍵とストリーム鍵）を使うと gaps が発生する理由は後述（3.3 同等出力系列発生条件の評価）しますが、真の乱数なら長さ 34 の gaps は 2^{34} ビット程度のデータを観測しなくては出現しないはずです。それが、（たとえテスト用の特殊な鍵を使ったにしても）簡単に見つかったのが、テストに不合格と評価した理由です。

次に、固定鍵、ストリーム鍵の生成方法に関する問題点を指摘します。

(1) 暗号技術仕様書には固定鍵 C の作り方は、「例」としてしか記載されていません。しかし、暗号学上危険な固定鍵を使用できないように何らかの制限を設ける（固定鍵作成アルゴリズムを明確に与える）必要があると考えます。固定鍵 C は線形フィードバックシフトレジスタの特性多項式（実際には原始多項式）であるので、ランダムに選択するわけにはいきませんが、ユーザの入力に対して安全な固定鍵を生成するアルゴリズムを明示しなくてはなりません。今回テスト用に使用した(評価者が作成した)固定鍵は、まさに暗号学上危険な固定鍵です。

(2) ストリーム鍵として 0x 00000000 00000000 00000000 0000000 以外のものを認めています。ストリーム鍵については条件を設けない設計としており、アルゴリズムにはストリーム鍵を攪拌する処理があります。しかし、この部分を改良する必要があると考えます。3.3 節で詳細に説明しますが、「暗号技術仕様書」の最終ページに記載されている例のような「 0x 00000000 00000000 00000000 00000001」等を使用した場合は同等出力系列を容易に作ることが可能となります。

なお、評価者のサンプリング数は決して十分とは思えませんが、選んだ鍵の中に、暗号学上危険な固定鍵とストリーム鍵があったのです。ランダムな固定鍵・ストリーム鍵、および特殊な固定鍵・ストリーム鍵を選択しても、乱数としての性質を保つことができるように十分テストする必要があると考えます。

3. 用途に対する適合性について

本章では用途に関する評価を行いました。

3.1. 推測可能性評価

入力の一部が露呈した場合の出力系列の推定可能性について考察しました。TOYOCRYPT-HR1 は、次の 2 つの値を鍵とします。

鍵	サイズ	用途
C (固定鍵)	128bit	線形フィードバックシフトレジスタの特性多項式
S (ストリーム鍵)	128bit	乱数生成に用いる種(seed)

入力の一部が露呈した場合の出力系列の推定可能性を数値で評価するのは困難ですが、Avalanche テストに合格することは、本テストに合格するための必要条件と考えます。

Avalanche テストに関する詳細は別冊 (疑似乱数生成の評価 Avalanche 性テスト TOYOCRYPT-HR1 編) を参照してください。提案方式は、Avalanche テストを合格しました。これは、「入力の一部がわかっても、出力結果に関する情報をえることができない」ことを示唆しています。もちろん十分条件ではないことに注意してください。

3.2. 周期 (出力系列の再現性) の評価

同一鍵を使用した場合、同一の出力系列が設計周期より小さい周期で発生しないか評価しました。本件と類似の「異なる鍵を使用して同一の出力系列を得ることは可能か」という議論は次の 3.3 節に記載します。

提案方式の設計上の周期は線形フィードバックシフトレジスタの周期であり、これは設計どおりと思われます。線形フィードバックシフトレジスタ以外の部分が、「線形フィードバックシフトレジスタ」の値に悪い影響を与える (常に打ち消しあうなど) ということはないと思えますので、同一鍵を使用して、同一の出力系列が発生するのは、線形フィードバックシフトレジスタ部分の周期に一致すると考えます。

3.3. 同等出力系列発生条件の評価

本節では、異なる鍵を使って同等の出力系列を発生させる方法が可能かどうか議論します。

まったく同じ出力系列を、異なる鍵を用いて発生させるのは困難と思われます。 Avalanche テストの結果は、その状況証拠のひとつとして挙げることができます。

本節では、まったく同じではないが、その類似である“同等の出力系列”を次のように定義します。

定義 1

乱数系列 $S(t)$ と $\overline{S}(t)$ が同等の出力系列とは、ある定数 i があって、次式が成立することとする。

$$S(t) = \overline{S}(t+i)$$

ここで、t はクロック (時間) である。

つまり、ある系列を i ビットずらした系列を得ることが容易であった場合はこの項目に不合格とします。

提案方式は、固定鍵 C とストリーム鍵 S の両方を秘密鍵とします。固定鍵 C を固定した場合、ランダムに選んだストリーム鍵 $S, \overline{S}$ に対して、同等の出力となる i は必ず存在します。周期が有限であるから当然です。この節で評価するのは「そのような i が存在するのはやむをえないとして、求めるのは困難である」ということです。しかし、ある条件のもとに、この i を求めることは可能です。そのためには、次の簡単な性質を使います。

(1) ストリーム鍵 S は（ランダムに選択されない場合を想定して）攪拌されます。この攪拌は線形フィードバックシフトレジスタの部分を使っており、実際には 256 クロックシフトするものです。

(2) ストリーム鍵の最上位ビットが 0 の場合は 1 クロック後の線形フィードバックシフトレジスタの値は、元のレジスタの「左への 1 ビット論理シフト」である。

(3) ストリーム鍵の最下位ビットが 0 の場合は 1 クロック前の線形フィードバックシフトレジスタの値は、元のレジスタの「右への 1 ビット論理シフト」である。

これらの性質のうち、(1)、(2)のみを使って、次のことが示せます。

まず、次の二つのストリーム鍵からは、固定鍵 C の値に関係なく、1 ビットだけずれた乱数系列が生成されます。

S1=0x 00000000 00000000 00000000 00000001

S2=0x 00000000 00000000 00000000 00000002

同様に、S=0x 0000….. 0004, 0000…. 0008 などからは、S1 から各々2,3 ビットだけずれた乱数系列が生成されます。結局、127 種類の同等な系列を得ることが可能となります。なお、この事実は長周期連性テストに不合格(0 が連続する状況、 長い gaps が多発)する原因になっています。テストとして選んだストリーム鍵 S(sa001, sb000-sb126)は、まさに、1 ビットずつの論理シフトなので、ある条件(特殊な固定鍵を使用)の下に発生した gaps は他の(論理シフトだけの)ストリーム鍵を使った場合にも発生するので、長い gaps が多発したのです。

もちろん S をランダムに選択すれば、「多くの同等な出力系列を得る」確率は小さくなりますが、しかし、先頭ビットが 0 である確率は、0.5 と評価できますので、「確率 0.5 で同等の出力系列を得ることができる」ということになります。

更に、(乱数などについてあまり知識のないプログラマにとっては)乱数生成のストリーム鍵(種；seed)として、小さい数値(1,2,3,4,…)を順に与えるという実装がありうること考慮しなくてはなりません。乱数生成器とは、「非乱数を与えても、乱数を生成してくれる」ものであると期待するプログラマは多いと予想されます。

同様の問題点(シフト方向を逆にしたもの)が、性質(1)、(3)のみを用いて記述することができます。

この問題は、線形フィードバックシフトレジスタにガロア・コンフィギュレーションを採用したから発生しています。例えば、ストリーム鍵の初期攪拌に、線形フィードバックシフトレジスタ以外の演算を加えることにより、本問題を回避することが可能かも知れませんが、提案方式では、そのような処理を行っていません。

4. 出力系列に対する入力空間の大きさ

4.1. 入力空間評価

入力空間のサイズについては、自己評価書(P.2)に示してある 248bit という値としてよいかどうかは疑問が残ります。前章で示したように、本方式では同等出力系列を比較的容易に生成することができます。ストリーム鍵の最上位ビットが 0 である確率は単純に 0.5 と評価できますので、「同等な出力系列」を同一な出力系列とみなしてよいのなら、極端に単純なストリーム鍵(例えば、上位ビットがほとんど 0)を使うことはできません。つまり、鍵空間は 248bit より小さくなるわけです。

また、「同等な出力系列を同一な出力系列とみなす」という立場では、最上位ビットが 0 のストリーム鍵 S から生成される出力系列と、S の論理シフト(最上位ビットが 1 になるまで)から生成される出力系列とは等価となります。

一方，運用面からも考慮が必要です。 (同等な出力系列の話を考慮しなければ)ストリーム鍵 S は 0x 00000000 00000000 00000000 00000000 を除けば任意に選ぶことができるのに対して，固定鍵 C は，任意に選ぶことができません。固定鍵 C が満たすべき条件は仕様書 4.6 節(P.8)によると

(1) $C(x)$ が F_2 上既約であること

(2) $x^{(2^{128}-1)/u_j} \neq 1 \bmod (C(x))$ であること（u_j は 9 通りの値をとる）

が必要となります。この条件を満たす固定鍵 C を生成するためには，多少複雑なプログラムを作らねばならず，一般ユーザにはかなりの苦労となります(少なくとも，評価者はかなり苦労しました)。

このような状況のとき、しばしば、グループ内で固定鍵を共有する(1 種類の鍵を使い続ける)という運用が予想されます。名前が「固定鍵」ですから、暗号や乱数について詳しくないユーザにとっては、「グループ内でひとつ用意すれば事足りるのか」という錯覚を起こす可能性もあります。

錯覚の問題はともかく、固定鍵Cを通信のセッション毎に生成する(しかも、共有する)のは、計算コストの面から見ても、現実的とは思えません。同一の固定鍵を使い続けるという運用に限っては、鍵空間の大きさは 128bit と評価されます。

5. ユーザサイドからの評価

仕様書を用いて実装する状況を想定して、問題点がないか考察します。

5.1. 固定鍵 C の生成方法について

128 ビットの固定鍵 C は秘密鍵と思われますので、ユーザサイドで生成しなくてはなりません。固定鍵 C の生成方法については、P.8 に「生成アルゴリズムの**例**」という形で紹介されていますが，この部分についても明確に仕様を決めるべきです。ユーザとしては固定鍵 C の生成方法がわからず、困ることが予想されます。また、固定鍵 C の生成手順で、

(1) $C(x)$ が F_2 上既約であること

(2) $x^{(2^{128}-1)/u_j} \neq 1 \bmod (C(x))$ であること（u_j は 9 通りの値をとる）

という記述がありますが、一般ユーザは，上記の判定を行うアルゴリズムを容易には作ることができません。固定鍵 C の生成アルゴリズムを例ではなく、明確な仕様として記述すべきと考えます。すなわち、暗号アルゴリズム(プログラム)の一部として考えるべきです。従って、そのアルゴリズム(プログラム)も当然評価のチェック対象とすべきです。

なお、多項式の既約判定やべき乗剰余演算のアルゴリズムを仕様書に記載する必要はないかもしれませんが、(1)および(2)の条件を判定するアルゴリズムが記載されている文献の引用は，少なくとも仕様書には必要であると思います。

6. 暗号解析の立場からの評価

本章では、提案アルゴリズムを(アルゴリズムの逆演算を行う立場から)解析する場合の弱点などについてコメントします。

6.1. g 関数について

g 関数は、線形フィードバックシフトレジスタの出力の一部を利用して構成されています。実際には、次の 3 つ値の排他的論理和です。

(1) $x_0 \cdot x_1 \cdots x_{63}$ (64 個の項の論理積)

(2) (17 個の項の論理積)

(3) (4 個の項の論理積)

まず、入力値である線形フィードバックレジスタ(の各レジスタ)は、各々ランダムなビットを生成するものとみなすことができます。従って、①の値が 1 になる確率は(64 個のレジスタが 1 になる確率であるから) $\frac{1}{2^{64}}$、②の値が1になる確率は $\frac{1}{2^{17}}$ 程度であり、ほとんど0といっても差し支えありません。複雑な演算を行う割には、効果が少ない(0を排他的論理和するということは、何も演算しないと同じこと)のではないでしょうか。

P.4 に記載されている「g を 63 次にした理由」を考えますと、効率化のため次数を減らすべきと断定はできませんが、関数 g 単体の出力がもう少しランダムになるように工夫(例えば、4 個の項の論理積を複数求めて、それらの排他的論理和を計算するなど)すべきだと思います。

実際、アルゴリズム全体に打撃を与える攻撃を明示できませんが、関数 g の出力結果の偏りを利用した攻撃が成立しかねません。まず、(3)の出力結果、つまり 4 個の項の論理積が1になる確率は $\frac{1}{2^4} \approx 0.06$ 程度です。すると、関数 g の出力が 1 となる確率は、(③に支配されるので) 0.06 と評価されます。つまり、h 関数の出力に g 関数の影響は、ほとんど影響しません。つまり、攻撃者サイドからは g 関数は存在しないものとみなす攻撃が有効となりかねません。(付録を参照してください)

6.2. 固定鍵 C の選び方について

固定鍵の選び方については制限を与えていませんが、0 と 1 が均一になるように選択したほうが安全です。極端な場合として、ほとんどのビットが 0 のものを考えます。例えば、テストで使用した C4001 のように、ビットが ON となるのは 4 箇所であり、しかも上位 64 ビットは全て 0 となるものを考えます。すると、ストリーム鍵の攪拌処理(線形フィードバックシフトレジスタのみを利用)は、あまり効果がありません。

例えば、このような固定鍵鍵 C4001 を利用した場合上に、ストリーム鍵 S として暗号仕様書の例にあるような 0x 00000000 00000000 00000000 00000001 などを使った場合は、S に対する攪拌処理を行っても、「ほとんどのビット(特に高位ビットのほとんど)が OFF」という性質が残ります。

S の攪拌直後のレジスタの値(特に上位 64 ビット)のほとんどが 0 となる場合は、大雑把に言うと次のような現象が起こります。

(1) x_{127} は 0。

(2) π 関数の出力結果は 0(内積をとるべきレジスタの上位が 0 であるから)

(3) g 関数が 0 を出力する確率は高い(4 つ以上の項の論理積だから、0 の分布に支配されるから)

この 3 条件が重なると出力系列に 0 が非常に多くなることがわかります。「長周期連性テスト」に不合格になった理由は、この 3 条件が揃ったからです。

7. ドキュメントについて

7.1. 仕様書の内容の確認

(1) 「暗号技術仕様書 TOYOCRYPT-HR1」の下記の記述は正しいでしょうか。ご確認願います。

#	内容
1	4.1 CLK 関数の記述について $x_{i+1} = c_i x_{127}(t), (0 \le i \le 126)$ には、排他的論理和に関する項がありません。図 6(P.9)と違います。誤植ではありませんか。

(2) テストデータ(P.15)の内容について

TOYOCRYPT-HR1(128,C,S)の出力は 0x f3ab6619 …(以下略) と記載されていますが，誤植ではありませんか。テストプログラムを実行すると， 0x f3ad6619 … (以下略) となります。

8. 性能について

ソフトウェアとして実装した際の性能評価を行いました。測定条件は下記の通りです。

#	項目	条件	
1	プログラム	提案方式の開発者により提供	
2	記述言語	C 言語	
3	測定装置の性能	Pentium III 600MHz，RAM128MB	
4	OS	Windows98	RedHat Linux 7.0J
5	コンパイラ	Visual C++ 6.0 Win32 Release 用オプション(/O2 等)	GNU C コンパイラ gcc 2.96 オプション –O2

実験では、10000000 バイトの乱数を生成するのに要する時間を測定しました。提案方式には、Serial/Parallel Implementation という二つの方式が記載されています。その両方に対して、10000000 バイトの乱数を生成するのに要した時間を_ftime(Visual C++)/gettimeofday(gcc)を用いて 5 回測定しました。測定結果を以下に示します。

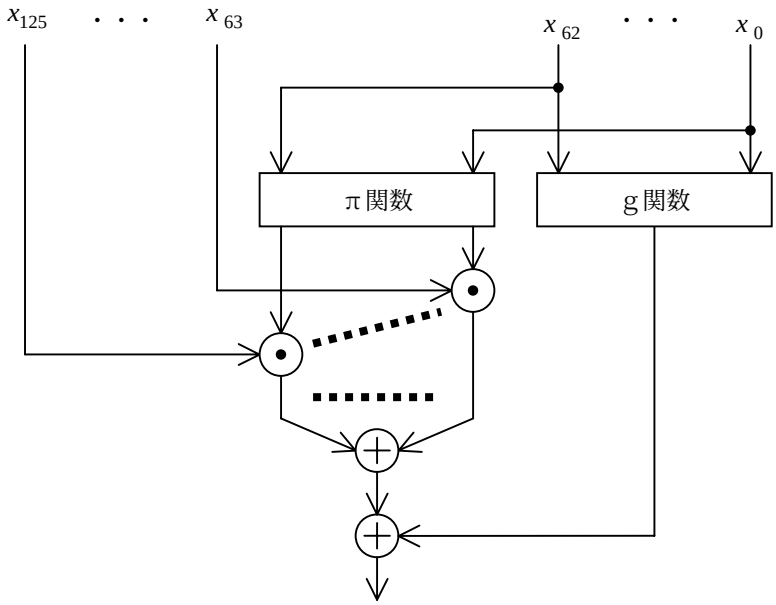
#	Windows98(Visual C++)		RedHat(gcc)	
	Serial	Parallel	Serial	Parallel
1 回目(秒)	64.300	5.050	83.375	4.653
2 回目(秒)	64.420	4.940	83.039	4.604
3 回目(秒)	64.320	5.000	83.026	4.598
4 回目(秒)	64.310	4.990	83.031	4.601
5 回目(秒)	64.260	5.110	82.998	4.597
平均(秒)	**64.322**	**5.018**	**83.0938**	**4.6106**
レート(Mbps)	**1.186**	**15.204**	**0.918**	**16.547**

Parallel Implementation を使用すると 15 倍程度の性能を得ることができました。

付録 1

***h* 関数のリダクション**

(1)**オリジナルの *h* 関数**



(2) ***g* 関数を常に** 0 **を出力する(実際には、**0 **を出力する確率は** 0.06 **程度)関数とみなした場合**
オリジナルのものと比較して、*h* 関数が簡単になります。