Robust and Cryptographically Secure Pseudo-Random Bit Generation
頑強で暗号学的に安全な疑似乱数ビット生成

Mpho Tjabane(110060)

北陸先端科学技術大学院大学 情報科学研究科

2003 年 2 月 14 日

キーワード: 乱数性、ビット生成、並行性、スレッド.

1 始めに

暗号システムの安全性は、しばしば予測不可能な数系列の生成に依存する。そのような系列に対する明確な必要条件（十分ではない）は、**乱数性**（ランダム性）である。ここで、ある系列がランダムであることを実証するために用いられる二つの基準は、**一様分布**と**統計的独立性**である。一様分布とは、その系列におけるそれぞれの数の生起頻度がほぼ同一であることを意味する。また、統計的独立性とは、その系列における様々な値に相関がないということである。ある系列が特定の分布に従うかどうかを決定するために、様々な統計的検査を行うことができる。しかしながら、独立性を判定するためのそのような統計的検査は存在しない。すなわち、ある系列における統計的独立性は、その系列の要素の予測不可能性を意味する。

ソフトウェアにおいては、決定性アルゴリズムによって乱数系列を生成したい場合がよくある。ここで決定性とは、同一の入力を与えられたときは**いつでも**同一の出力を生成することを意味する。コンピュータは決定性機械なので、何らかの手段によって乱数データを計算しなければ、それを生成することは**できない**。このような状況は、自然現象によって生じる乱数性とは非常に異なっている。コンピュータは「真の」乱数を生成することはできないので、それが生成する乱数データは疑似乱数と呼ばれる。

この研究の主眼は、多くの統計的検査による疑似乱数を生成することができる決定性アルゴリズムを設計することにある。そのような検査の統計は、データ項目における構造（あるいはそのような構造がないこと）を明確にする。

2 研究の目的

何年にも渡って、多くの異なった疑似乱数生成器が設計されてきた。それらのアルゴリズムのほとんどは（すべてではないにせよ)、単一プロセッサ計算機における実装を想定して設計されていた。今日では、マルチプロセッサ計算機やLAN（local area networks）によって接続された複数の計算機が存在する。本研究の主な目的は、現在の技術をうまく利用するためにアルゴリズム設計がどのように発展しなければならないかを示すことである。特に、本研究は以下の点を明らかにする：

- スループットの高い、計算量的に予測困難なランダムビット生成アルゴリズムの構築、
- 64 ビットプラットフォームのような、現在利用可能なハードウェア仕様を十分に利用した、上記アルゴリズムの効率的な実装、
- マルチプロセッサやネットワークに接続されたワークステーションにおける実装を容易にするための、本質的に「並列な」アルゴリズムの構築、
- 単一プロセッサにおける並行性を得るための、上記の本質的な「アルゴリズム的並列性」の使用法、
- 並列アルゴリズムが並列生産者-消費者問題に帰着される場合のマルチスレッドのスケジューリングを複雑にする、POSIX スレッド標準の実装における問題点の指摘。

3 TM3w ビット生成器

以後の参照を容易にするため、提案アルゴリズムを TM3w [2]と呼ぶことにする。このビット生成器は、以下のようなフィボナッチ再帰式に基づいている。

$$X_n = X_{n-r_i} + X_{n-s_i} \bmod 2^w \tag{1}$$

ここで、$\{X_0, ..., X_{r_i}\}$ は初期値であり、w は計算機のワード長である。また、r_i と s_i は、$x^{r_i} + x^{s_i} + 1 \bmod 2$ が原始多項式であり、$\forall i \neq j \neq k$ について $\gcd(r_i, r_j) = \gcd(r_j, r_k) = 1$ で、$r_i > r_j > r_k$ となるように選ぶ。このような順序付けは、初期化の時点で必要となる。(1) 式によって定義される生成器は、$p = 2^{(w-1)}(2^{r_i} - 1)$ なる周期を持つ。

本研究で提案されるランダムワード生成器は、以下の 3 つの加算生成器によって導出される。

$$A_n = (A_{n-47} + A_{n-5}) \bmod 2^{32}, n \geq 47 \tag{2}$$

ここで、

[2]著者の頭文字である。3w は、ワード長が w であるようなコンピュータにおいて、アルゴリズムの出力が $3 \times w$ ビットであるという事実による。

$$x^{47} + x^5 + 1 \pmod 2 \tag{3}$$

は原始多項式である。二つ目の生成器は以下で与えられる：

$$B_n = (B_{n-41} + B_{n-3}) \bmod 2^{32}, n \geq 41 \tag{4}$$

ここで、

$$x^{41} + x^3 + 1 \pmod 2 \tag{5}$$

は原始多項式である。三つ目の生成器は以下で与えられる：

$$C_n = (C_{n-35} + C_{n-2}) \bmod 2^{32}, n \geq 35 \tag{6}$$

ここで、

$$x^{35} + x^2 + 1 \pmod 2 \tag{7}$$

は原始多項式である。

この 3 つの生成器の初期状態は、32 ビットワードを含む配列である。これらの配列は、ユーザから与えられた初期鍵によって導出され、ビットローテートや配列要素の省略により、それぞれのユーザ毎に異なる値を持つ。この初期鍵は、原始多項式の最高次の次数の生成器の初期状態と同じ長さである。たとえば、式 (4), (6), (8) における原始多項式のべき乗の指数は、それぞれ {47, 5}, {41, 3}, {35, 2} であり、そのバイナリ演算は 2^{32} を法とした加算である。そして、初期状態は以下のようになる：

$$key = \{k_0, k_1, ..., k_{46}\} \tag{8}$$

ここで、$\forall i = 0 : 46$ において、k_i は 32 ビットの配列要素である。最初の生成器の初期状態は、以下のようにして生成される：

$$\{k'_0, k'_1, ..., k'_{46}\} = \{k_0, k_1, ..., k_{46}\} <<< 20 \tag{9}$$

上式において、20 は $(47+5)-32$ として求められる。またここで、$<<<$ は、ビットローテートを表す。式 (9) は、配列全体のビットローテートを示している。このローテートの後、生成器 A の初期状態は、以下の代入によって求められる：

$$\{A_0, A_1, ..., A_{46}\} = \{k'_0, k'_0 \uplus k'_1, ..., \uplus_{i=0}^{46} k'_i\} \tag{10}$$

ここで、$\uplus$ は、2^{32} を法とした整数加算を表す。生成器 B の初期状態は、最初の生成器の初期状態から下記のような過程によって得ることができる。まず、

$$\{A'_0, A'_1, ..., A'_{46}\} = \{A_0, A_1, ..., A_{46}\} <<< 12 \tag{11}$$

12 は、$(41 + 3) - 32$ として得られる。式 (9) における配列は、A'_{41} から A'_{46} までの要素を取り除き、以下の代入によって求められる：

$$\{B_0, B_1, ..., B_{40}\} = \{A'_0, A'_0 \uplus A'_1, ..., \uplus_{j=0}^{40} A'_j\} \tag{12}$$

式 (8), (10) と以下の式では、ビットローテートにより、異なるビットパターンとなる。生成器 C では、この過程は以下のようになる。

$$\{B'_0, B'_1, ..., B'_{40}\} = \{B_0, B_1, ..., B_{40}\} <<< 5 \tag{13}$$

ここで、5 は $(35 + 2) - 32$ として求められる。B'_{35} から B'_{40} までの要素を取り除き、以下の初期状態を得る。

$$\{C_0, C_1, ..., C_{34}\} = \{B'_0, B'_0 \uplus B'_1, ..., \uplus_{k=0}^{34} B'_k\} \tag{14}$$

モジュロ加算、ビットローテート、ビット削除をインクリメンタルに使用が、この生成器の生成する系列の予測をより困難にしている。

3.1 ランダムワード生成

ランダムワード生成の過程は、式 (1), (3), (5), および初期状態 (9), (11), (13) を使った繰り返しからなる。さらに、撹乱を得るために、上記の式からの出力は、以下の式 (14) から (16) へと渡される。ここで、$\langle A_k \rangle$, $\langle B_l \rangle$, $\langle C_m \rangle$ を、それぞれ生成器 A、B、C によるワード系列とする。そして、三つの 32 ビットワード $W_{j,1}$, $W_{j,2}$, $W_{j,3}$ が以下の式を用いて生成される。

$$W_{j,1} = ((A_k \odot B_l) \uplus A_{k+1}) \oplus C_m \tag{15}$$
$$W_{j,2} = ((B_l \odot C_m) \uplus B_{l+1}) \oplus A_k \tag{16}$$
$$W_{j,3} = ((C_m \odot A_k) \uplus C_{m+1}) \oplus B_l \tag{17}$$

ここで、$\odot$, $\uplus$, $\oplus$ は、それぞれ、法 2^{32} の乗算、法 2^{32} の加算、法 2^{32} の排他的論理和を表す。インデックス変数 j, k, l, m の下限は、それぞれ $j \geq 0$, $k \geq 46$, $l \geq 40$, $m \geq 34$ である。また、以下の順序つきペアが、上記組合わせ生成器を初期化するために必要である：$\{A_{46}, A_{47}\}$, $\{B_{40}, B_{41}\}$, $\{C_{34}, C_{35}\}$。全数探索評価では、中間および最終的生成器の予測のために必要となる計算量は、現実的でないほどに巨大となる。

4　並行性

TM3w の実装においては、式 (2), (4), (6) と式 (14), (15), (16) は、三人の生産者および三人の消費者からなる生産者-消費者問題を定義する。それぞれの生産者は、3 人の生産者のそれぞれの出力を入力として受け取る。これを単一プロセッサにおいて実装するとき、すべての生産者-消費者は二つの計算機資源を共有しなければならない。すなわち、CPU 時間とメモリである。これにより、すべてのプロセスが実行されることを保証するために同期問題が生じる。それぞれの生産者あるいは消費者は、スレッド[3]として実現され、それらのスレッドはある制御機構によりリソースにアクセスする。POSIX 標準によって提供される制御機構は、mutex （mutual exclusions）と条件変数である。これらの機構は、他のスレッドからの干渉を受けることなくスレッドによる共有データの操作を可能にする。

5　結論

本研究により、提案されたランダムビット生成器は高いスループットを持つという意味で効率的であり、出力系列を予測するときに必要となる計算量の観点から、暗号学的に安全であると言える。

また、単一プロセッサにおける並行性を求めると、結果として移植可能性の低い実装になる。これは、並行性を得るための手段を提供するアプリケーション・プログラミング・インターフェースが、特定のオペレーティングシステムに束縛されるからである。現在 4 つのスレッド API が存在するものの、多くの共通点を持つのはその中のたった二つ（Solaris と POSIX）しかない。また、たとえそうであっても、一つの実装を他に移植するのは双方の深い知識を必要とする。C 言語における 64 ビットデータ型の欠落は、64 ビットプロセッサの単一精度計算を阻害する。また、32 ビット長の unsigned long 型は、96 ビットの出力になる。もし 64 ビットデータ型が定義されたとすると、出力は 192 ビットになるだろう。大きな違いである。

[3] スレッドは、プロセスの中における単一の制御の流れとして定義される。