MATEMATYKA STOSOWANA 2, 2001

Zbigniew Kotulski (Warszawa)

Generatory liczb losowych: algorytmy, testowanie, zastosowania

1. WSTĘP

1.1. Losowość i liczby losowe. Wiele procesów obserwowanych przez nas w przyrodzie, technice, ekonomii czy życiu społecznym sprawia wrażenie zjawisk losowych, a więc takich, dla których nie potrafimy przewidzieć ich przyszłego przebiegu ani nie potrafimy ustalić przyczyn, które je wywołały. Powodem tego może być brak informacji dotyczących danego zjawiska, nieznany stopień precyzji dostępnych informacji lub też błędy obserwacji uniemożliwiające precyzyjną jego identyfikację. Na przeszkodzie może stać tu brak technicznych możliwości uzyskania dostępnych informacji lub niemożność wykonania jakichś istotnych pomiarów. Przyczyna losowości zjawiska może być też inna: jego specyficzne cechy fizyczne albo niezmierna komplikacja niemożliwa do objęcia żadnym zdeterminowanym modelem. W każdej z przedstawionych sytuacji, niezależnie od tego, czy udaje się nam ustalić, jakie są przyczyny losowości zjawiska (por. np. [9], [24]), możemy spróbować opisać to zjawisko ilościowo, wykorzystując do tego celu pojęcie prawdopodobieństwa rozumianego jako ilościowa miara niepewności (losowości).

Podobnie jak w fizyce i przyrodzie, efekty losowe mogą występować również w świecie liczb. Na przykład, pytanie dotyczące częstotliwości występowania liczb pierwszych wśród liczb naturalnych, w szczególności ich dokładne rozmieszczenie, pozostaje bez odpowiedzi (Próby odpowiedzi na te i podobne pytania można znaleźć w [50]). Więcej można powiedzieć o średniej (w ograniczonym przedziale lub asymptotycznie) częstości występowania tych liczb, zwłaszcza korzystając z obliczeń komputerowych (por. np. [20]). Dla przeciętnego obserwatora pojawienie się liczby pierwszej w ciągu dowolnych (dużych) liczb naturalnych jest zjawiskiem losowym. Bez dokładnego sprawdzenia (poza oczywistymi sytuacjami liczb parzystych, podzielnych przez 5, itp.) nie potrafimy powiedzieć, czy dana liczba jest pierwsza i jaka będzie następna po niej liczba pierwsza. Dla dużych liczb naturalnych

nie są również w sposób oczywisty znane składniki ich rozkładu na czynniki pierwsze (ich znalezienie wymaga dużego nakładu pracy, co jest podstawą bezpieczeństwa pewnych kryptosystemów). Wśród liczb naturalnych mamy zatem do czynienia z ciągami liczb losowych, czyli takich, których pojawienie się nie może być z pewnością przewidziane, a struktura nie może być opisane żadnym określonym wzorcem.

W ogólnej sytuacji, *ciągiem liczb losowych* (ogólniej: losowym ciągiem znaków) nazwiemy taki ciąg, którego nie można zapisać za pomocą algorytmu w postaci krótszej od samego ciągu. Na podstawie takiego ciągu nie można stworzyć żadnych reguł, które pozwalałyby odtworzyć ten ciąg bez znajomości wszystkich jego wyrazów. Nie można również podać żadnego wyrazu tego ciągu na podstawie znajomości innych (wszystkich pozostałych) wyrazów tego ciągu.

Analogicznie jak w przyrodzie, możliwa jest jednak i taka sytuacja, że istnieją reguły (krótkie w porównaniu z długością ciągu) opisujące sposób wypisania wszystkich jego wyrazów, jednak jedynie nasza niewiedza nie pozwala ich zidentyfikować. Ciąg taki nazwiemy *pseudolosowym* i w wielu sytuacjach będziemy go mogli traktować tak jak losowy ciąg liczb.

1.2. Wykorzystanie liczb losowych. Wiemy już, że istnieją losowe ciągi liczb. Powstaje pytanie: jak można je wykorzystać dla celów praktycznych? Liczby losowe przydają się wszędzie tam, gdzie efekt przypadkowy, losowy, jest lepszy niż działanie zdeterminowane przez człowieka, a więc obciążone jego subiektywną oceną sytuacji. Liczby losowe można wykorzystać w reprezentatywnych badaniach statystycznych (przy losowaniu próby z populacji generalnej lub, szerzej, planowaniu schematu losowania), a zatem w zagadnieniach statystycznej kontroli jakości produktów, wszelkich badaniach ekonomicznych, społecznych, marketingowych itp. W naukach eksperymentalnych liczb losowych możemy użyć w zagadnieniach planowania eksperymentu, chociażby dokonując podziału poletek doświadczalnych dla celów porównania różnych czynników w uprawach rolnych lub też wybierając do badań próbkę materiału z większej objętości.

Innym sposobem wykorzystania liczb losowych są wszelkie badania symulacyjne. Są to metody wykorzystujące techniki obliczeniowe, w których przedstawiamy przebieg realnego zjawiska, opisanego odpowiednimi równaniami, uwzględniając wpływające na nie czynniki losowe. Badania takie są często jedynym sposobem ilościowej analizy skomplikowanego procesu technologicznego lub zjawiska przyrodniczego. Podobnie, liczby losowe są źródłem losowości stwarzającej złudzenie realizmu we wszelkich popularnych grach komputerowych, inteligentnych automatach do gier zręcznościowych, trenażerach oraz profesjonalnych grach strategicznych, to znaczy w programach umożliwiających wirtualny udział gracza w operacjach wojennych,

ekonomicznych lub społecznych. Liczby losowe mogą ponadto służyć do budowania modeli skomplikowanych obiektów geometrycznych, na przykład fraktali losowych, wzorów powierzchni itp. Również badania symulacyjne statystyki matematycznej, tak zwane metody bootstrapowe lub sznurowadłowe, wymagają zastosowania liczb losowych.

Liczby losowe są także niezbędnym elementem metod Monte Carlo (por. [13], [68], [41]), czyli wykorzystania metod probabilistycznych w obliczeniach przeprowadzanych zwykle metodami deterministycznymi, na przykład w przybliżonym obliczaniu całek wielowymiarowych, rozwiązywaniu równań różniczkowych i algebraicznych, optymalizacji (często sprowadzającej się do szukania minimum funkcji), algorytmach genetycznych itd. Metody te mogą być, szczególnie dla zadań dotyczących nieregularnych obszarów i funkcji, bardziej efektywne od metod klasycznych.

Zastosowaniem liczb losowych, które w ostatnim czasie zyskało ogromnie na znaczeniu, jest kryptografia, czy też szerzej rozumiana ochrona informacji. Oprócz tradycyjnych zastosowań kryptografii w wojskowości i dyplomacji, w ostatnich latach powstały nowe pola jej zastosowań. Związane są one z nowymi masowymi sposobami przesyłania danych, w których niezbędna jest ochrona informacji, to znaczy z telefonią komórkową i rozległymi sieciami komputerowymi. Dziedziny te wymagają udostępnienia tanich i wydajnych źródeł liczb losowych (służących jako klucze w szyfrach strumieniowych), w dodatku liczb spełniających szczególne wymagania gwarantujące bezpieczeństwo szyfru. W dalszej części tej pracy postaramy się pokazać, jakie są praktyczne sposoby tworzenia ciągów liczb losowych niezbędnych w wymienionych (i niewymienionych) problemach praktycznych.

2. METODY OTRZYMYWANIA LICZB LOSOWYCH (HISTORYCZNE)

Wiemy już, że liczby losowe mają obecnie wiele zastosowań. Także i w przeszłości, w trakcie prowadzenia reprezentatywnych badań statystycznych, odczuwano potrzebę wykorzystania takich liczb. Pierwszymi możliwymi do wykorzystania w praktyce źródłami liczb losowych były tablice liczb losowych. Wymieńmy tutaj, za [48] i [70], kilka takich zestawień, wraz ze sposobami, w jaki zostały sporządzone.

Pierwszą tablicę liczb losowych wydał w roku 1927 L. H. Tippett pod tytułem „Random Sampling Numbers". Zawierała ona 41600 cyfr (od 0 do 9) pobranych z danych ze spisu powszechnego w Wielkiej Brytanii. Cyfry te uzyskano z liczb wyrażających powierzchnie parafii, po odrzuceniu dwóch pierwszych i dwóch ostatnich cyfr z każdej liczby. W 1939 R. A. Fisher i F. Yates podali tablicę 15000 cyfr losowych, uzyskaną przez wypisanie cyfr od 15. do 19. z pewnych 20-cyfrowych tablic logarytmicznych. W tym samym roku Kendall, Babington i Smith przedstawili tablicę 100000 cyfr losowo uzyskanych za pomocą „elektrycznej ruletki", czyli wirującego dysku

z oznaczeniami cyfr $0, 1, \ldots, 9$, obserwując w przypadkowych chwilach wybrany sektor ruletki.

W Polsce w roku 1951 opracowano dla potrzeb GUS tablicę liczb losowych, wykorzystując do tego paski do drukujących maszyn liczących. Z liczb co najmniej czterocyfrowych wydrukowanych na paskach skreślano jedną cyfrę końcową i dwie początkowe, a także wykreślano niektóre kolumny. Uzyskany wynik, przed zapisaniem w tablicy, sprawdzano testami statystycznymi.

W roku 1955 w RAND Corporation uzyskano tablicę 1000000 cyfr losowych w sposób, który i dzisiaj mógłby być wykorzystany w praktyce. Zbudowano źródło wytwarzające 100000 impulsów binarnych na sekundę. Impulsy odczytywano paczkami pięciobitowymi, otrzymując liczby z przedziału $[0, 31]$. Zachowywano liczby z przedziału $[0, 19]$ i zapisywano ich młodszą cyfrę dziesiętną do tablicy. Tablice cyfr losowych oferowano także na kartach perforowanych, co umożliwiało stosowanie ich w obliczeniach komputerowych.

Tablice liczb losowych miały ograniczoną długość i zawierały tylko jeden ciąg takich liczb. W celu przedłużenia ich żywotności (nie można było stale wykorzystywać tych samych liczb, bo to przeczyłoby idei losowości) opracowywano algorytmy wytwarzania ciągów losowych na podstawie tablic. Oto taki przykładowy algorytm (dostosowany do rozmiaru całej tablicy i sposobu zapisu cyfr w kolumnach i wierszach).

Generowanie ciągów liczb losowych na podstawie tablicy cyfr losowych

1. Wybrać losowo liczbę pięciocyfrową z tablicy.
2. Zredukować pierwszą cyfrę tej liczby modulo 2. Tak zmodyfikowana liczba pięciocyfrowa wskaże numer wiersza w tablicy.
3. Zredukować dwucyfrową końcówkę tej liczby modulo 50. Tak otrzymana liczba dwucyfrowa wskaże numer kolumny w tablicy.
4. Rozpocząć ciąg losowy od wskazanej pozycji w tablicy.

3. METODY OTRZYMYWANIA LICZB LOSOWYCH (WSPÓŁCZESNE)

3.1. Generatory fizyczne. Współczesne metody generowania liczb losowych można podzielić na dwie grupy. Jedna z nich to wykorzystanie algorytmów matematycznych (lub ich sprzętowej realizacji), a więc metod, które są powtarzalne (ten sam ciąg liczb losowych, a właściwie pseudolosowych, można otrzymać wielokrotnie, powtarzając przebieg algorytmu z tymi samymi parametrami). Metody algorytmiczne są przedmiotem zainteresowania tej pracy. Zanim jednak do nich przejdziemy, wspomnijmy o drugiej grupie metod: generatorach fizycznych. Proces wytwarzania ciągu liczb losowych polega w nich na zamianie na liczby mierzonych parametrów procesu fizycznego przebiegającego w sposób losowy. W tym wypadku zarówno

powtórzenie trajektorii procesu, jak i powtórne uzyskanie identycznego ciągu liczb losowych nie jest możliwe. Nie wnikając głębiej w technikę uzyskiwania tą drogą ciągów liczb losowych, wymienimy w punktach przykłady takich generatorów. Należą do nich:

- proste generatory fizyczne (mechaniczne), takie jak moneta, urna z kolorowymi kulami, kostka do gry, ruletka itd.;
- licznik impulsów promieniowania jądrowego (licznik Geigera), korzystający z naturalnej losowości tego zjawiska;
- liczniki impulsów elektronicznych pochodzących z dysku komputera, wskaźnika myszy, szumów monitora, karty dźwiękowej i innych podzespołów komputera;
- systemy pobierania losowych bitów z arytmometru komputera lub z klawiatury (w intensywnie eksploatowanych zestawach komputerowych);
- urządzenia elektroniczne konstruowane specjalnie w celu generacji liczb losowych, zwłaszcza w zapisie binarnym. Ich podstawowym elementem są diody szumowe. Dostarczane są jako karty komputerowe lub moduły podłączane do portów zewnętrznych.

Generatory fizyczne wymagają bezwzględnie testowania każdego wygenerowanego ciagu liczb losowych przed jego użyciem, ponieważ w wyniku awarii urządzenia ciągi te mogą utracić oczekiwane własności. Ponadto, do zastosowań kryptograficznych wygenerowane ciągi muszą być zapisywane na zewnętrznym nośniku w dwóch kopiach.

3.2. Generatory kongruencyjne

3.2.1. *Liczby losowe o rozkładzie równomiernym.* Najważniejszym elementem generowania liczb losowych o dowolnym rozkładzie prawdopodobieństwa jest otrzymywanie liczb losowych o rozkładzie równomiernym. Polega ono na uzyskaniu ciągu liczb całkowitych z ustalonego przedziału $[1, M]$ w taki sposób, by wszystkie z nich występowały z jednakowym prawdopodobieństwem, by częstotliwość występowania liczb z każdego z podprzedziałów tego przedziału była w przybliżeniu jednakowa w czasie i ponadto by można było te liczby uznać za statystycznie niezależne.

Zauważmy, że z posiadanych losowych liczb całkowitych $\{X_i,\, i=1,2,\ldots\}$ o rozkładzie równomiernym w $[1, M]$ zawsze możemy uzyskać liczby losowe $\{R_i,\ i = 1,2,\ldots\}$ o ciągłym rozkładzie równomiernym na $[0,1]$ poprzez przekształcenie

$$R_i = \frac{X_i}{M}, \qquad i = 1,2,\ldots \tag{3.1}$$

3.2.2. *Generator liniowy i afiniczny.* Pierwszym zaproponowanym i do dziś podstawowym w wielu zastosowaniach jest generator liniowy (por. [34]).

Kolejne liczby losowe $X_1, X_2, \ldots$, są obliczane z następującego równania rekurencyjnego:

$$X_{n+1} = g * X_n \bmod M, \tag{3.2}$$

gdzie warunek początkowy (ziarno) $X_0 < M$ musi być zadany, natomiast sam generator jest zdefiniowany przez odpowiedni dobór parametrów M oraz $g < M$.

Pewnym uogólnieniem generatora (3.2) jest generator afiniczny, przedstawiany jako

$$X_{n+1} = a * X_n + b \bmod M, \tag{3.3}$$

gdzie $n = 1, 2, \ldots$ oraz $g, a, b < M$.

Okres generatorów (3.2) i (3.3) zależy od wartości parametrów ich równań kongruencyjnych. Informacje dotyczące długości okresu generatora liniowego zawarte są w następujących twierdzeniach:

TWIERDZENIE. *Jeżeli $M = 2^m$ dla $m \geq 3$, to maksymalny okres generatora liniowego wynosi $N = 2^{m-2}$, gdy*

$$g \equiv 3 \bmod 8 \quad \textit{lub} \quad g \equiv 5 \bmod 8 \tag{3.4}$$

(tj. reszta z dzielenia g przez 8 wynosi 3 lub 5).

TWIERDZENIE. *Jeżeli $M = p$ jest liczbą pierwszą, to generator liniowy posiada okres maksymalny równy p. Okres ten jest osiągany, gdy g jest pierwiastkiem pierwotnym liczby p.*

(Każda liczba m taka, że $m \bmod N$ jest generatorem grupy cyklicznej $G(N)$, jest pierwiastkiem pierwotnym liczby N. $G(N)$ jest zbiorem wszystkich reszt $\bmod N$).

Wadą generatorów liniowych jest ich przewidywalność, to znaczy układanie się punktów o współrzędnych (X_n, X_{n+1}) na linii prostej.

3.2.3. *Uogólniony generator liniowy.* Próbą uniknięcia własności przewidywalności generatora liniowego jest jego uogólnienie polegające na uwzględnieniu kilku poprzednich elementów przy obliczaniu elementu bieżącego, to znaczy wykorzystaniu wzoru

$$X_n = a_1 * X_{n-1} + \ldots + a_k * X_{n-k} + b \bmod M \tag{3.5}$$

dla pewnych stałych $a_1, a_2, \ldots, a_k, b < M$. Przy odpowiednim ich doborze generator (3.5) może osiągnąć maksymalny okres M. Zależność od przeszłości jest bardziej skomplikowana niż w (3.2) i (3.3), jednak ta komplikacja nie jest wystarczająca, by uogólniony generator liniowy mógł służyć do celów kryptograficznych.

Obecnie prowadzone są szeroko prace dotyczące znalezienia takich wartości parametrów uogólnionych generatorów liniowych, by otrzymać tą drogą ciągi liczb losowych o dobrych własnościach statystycznych. Rezultaty uzyskane w tej dziedzinie są omówione szczegółowo w pracy [35].

3.2.4. *Kwadratowy generator kongruencyjny.* W celu uniknięcia liniowej zależności kolejnych wyrazów ciągu zastosowano kwadratową zależność między liczbami w kongruencji definiującej generator:

$$X_{n+1} = a * X_n^2 + b * X_n + c \bmod M \tag{3.6}$$

dla $a, b, c, X_0 < M$. Maksymalny okres tego generatora, dla odpowiednich wartości parametrów a, b i c, może być równy M.

3.2.5. *Wykorzystanie wielomianów permutacyjnych.* Uogólnieniem generatora kwadratowego jest generator wykorzystujący wielomiany permutacyjne postaci

$$g(X) = \sum_{k=0}^{n} a_k X^k, \qquad a_1, \ldots, a_n \in \{0, 1, \ldots, M-1\}, \tag{3.7}$$

jako zależność między wyrazami generowanego ciągu liczb losowych

$$X_{n+1} = g(X_n) \bmod M. \tag{3.8}$$

Maksymalny okres generatora (3.8) jest równy M. W przeciwieństwie do generatora liniowego nie jest on przewidywalny.

3.2.6. *Generator inwersyjny.* Przykładem nieliniowej metody generowania liczb losowych o rozkładzie równomiernym jest również generator inwersyjny (por. [12]). Kolejne liczby losowe uzyskuje się tu z wzoru

$$X_{n+1} = \begin{cases} aX_n^{-1} + b \bmod p & \text{dla } X_n \geq 0, \\ b & \text{dla } X_n = 0, \end{cases} \tag{3.9}$$

gdzie $n = 0, 1, 2, \ldots$ oraz p jest liczbą pierwszą. Maksymalny okres takiego generatora, przy odpowiednim doborze współczynników a i b, może być równy $p - 1$.

Szerokie omówienie nieliniowych metod generacji liczb losowych o rozkładzie równomiernym, zwłaszcza generatorów inwersyjnych, znaleźć można w pracy [42]. Inne metody generowania liczb losowych o rozkładzie równomiernym zawiera też książka [65].

3.3. Liczby losowe o dowolnych rozkładach ciągłych. Ciągłymi rozkładami prawdopodobieństwa nazywamy takie rozkłady, których dziedzina jest zbiorem zawierającym odcinek. Ciągi liczb losowych o rozkładach dowolnych są zazwyczaj otrzymywane z liczb losowych o rozkładach równomiernych w wyniku zastosowania odpowiednich twierdzeń: twierdzenia o odwracaniu dystrybuanty, twierdzeń granicznych lub innych rezultatów specyficznych dla danego rozkładu prawdopodobieństwa. Wiele przykładów generatorów liczb losowych o rozkładach nierównomiernych znaleźć można, na przykład, w książkach [26], [65], [69]. W tej pracy podamy jedynie kilka przykładowych metod generacji takich liczb.

3.3.1. *Metoda odwracania dystrybuanty.* Załóżmy, że $F(\cdot)$ jest dystrybuantą pewnego rozkładu prawdopodobieństwa, a

$$X_i, \quad i = 1, 2, \ldots, \tag{3.10}$$

jest ciągiem niezależnych zmiennych losowych o rozkładzie równomiernym na odcinku $[0, 1]$. Wówczas ciąg zmiennych losowych zdefiniowanych jako

$$Y_i = F^{-1}(X_i), \quad i = 1, 2, \ldots, \tag{3.11}$$

jest ciągiem niezależnych zmiennych losowych o rozkładzie zadanym dystrybuantą F.

Przykład. Rozważmy dystrybuantę rozkładu wykładniczego

$$F(x) = 1 - e^{-x}. \tag{3.12}$$

Funkcją odwrotną do (3.12) jest

$$F^{-1}(y) = -\ln(1 - y) \tag{3.13}$$

i dla niezależnych zmiennych losowych (3.10) ciąg

$$Y_i = -\ln(1 - X_i), \quad i = 1, 2, \ldots, \tag{3.14}$$

jest ciągiem niezależnych zmiennych losowych o rozkładzie wykładniczym.

Istotnym ograniczeniem w zastosowaniu tej metody jest trudność obliczenia dla niektórych rozkładów funkcji odwrotnej do ich dystrybuanty. Ograniczenie to można jednak ominąć, wykorzystując wyrażenia przybliżone dla dystrybuant lub funkcji do nich odwrotnych.

Przykład. Dystrybuanta $\Phi(t)$ rozkładu normalnego $N(0, 1)$ nie może być wyrażona przez funkcje elementarne. Jednak można ją przybliżyć z dużą dokładnością przez takie funkcje. Korzystając z wyrażenia

$$\Phi(t) = \begin{cases} 1/2 - \operatorname{erf}(-t) & \text{dla } t \le 0, \\ 1/2 + \operatorname{erf}(t) & \text{dla } t > 0, \end{cases} \tag{3.15}$$

gdzie jest $\operatorname{erf}(\cdot)$ jest tzw. funkcją błędu, i z przybliżonego wyrażenia dla tej funkcji (por. [1])

$$\operatorname{erf}(t) = 1 - (a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4 + a_5 z^5) \exp[-z^2] + \varepsilon(z),$$
$$z = \frac{1}{1 + pt},$$

gdzie

$$|\varepsilon(z)| \le 1{,}5 \times 10^{-7}$$

oraz

$$p = 0{,}3275911, \quad a_1 = 0{,}2548295921,$$
$$a_2 = -0{,}284496736, \quad a_3 = 1{,}421413741,$$
$$a_4 = -1{,}453152027, \quad a_5 = 1{,}061405429,$$

możemy uzyskać przybliżenie $\Phi(t)$ z dokładnością rzędu 10^{-7} dla wszystkich $t \in (-\infty, \infty)$.

3.3.2. *Metoda eliminacji* (*J. von Neumann*). Metoda eliminacji oparta jest na tej własności zmiennych losowych, że ich histogram z próby dąży do gęstości rozkładu (jeśli ta istnieje – por. [59]). W przypadku jednowymiarowym pozwala ona generować liczby losowe o wartościach z przedziału (a, b) o rozkładzie zadanym gęstością prawdopodobieństwa $f(x)$ spełniającą warunek $f(x) \leq c$ dla $x \in (a, b)$. (W przypadku wielowymiarowym zmienna losowa przyjmuje wartości w kostce $(a, b)^n$).

Algorytm dla rozkładu jednowymiarowego

1. Generujemy dwie liczby losowe R_1 i R_2, odpowiednio,
   - R_1 o rozkładzie $U(a, b)$,
   - R_2 o rozkładzie $U(0, c)$.
2. Sprawdzamy, czy

$$R_2 \leq f(R_1). \tag{3.16}$$

Jeżeli warunek (3.16) jest spełniony, to przyjmujemy

$$X = R_1. \tag{3.17}$$

W przeciwnym wypadku eliminujemy wylosowany punkt i powtarzamy losowanie z kroku 1.

Algorytm dla rozkładu wielowymiarowego

1. Generujemy $n+1$ liczb losowych $R_1, R_2, \ldots, R_n, R_{n+1}$, w tym $R_1, R_2, \ldots, R_n$ o rozkładzie $U(a, b)$, R_{n+1} o rozkładzie $U(0, c)$.
2. Sprawdzamy, czy

$$R_{n+1} \leq f(R_1, R_2, \ldots, R_n). \tag{3.18}$$

Jeżeli warunek (3.18) jest spełniony, to przyjmujemy

$$(X_1, X_2, \ldots, X_n) = (R_1, R_2, \ldots, R_n). \tag{3.19}$$

W przeciwnym wypadku eliminujemy wylosowany punkt i powtarzamy losowanie z kroku 1.

3.3.3 *Metoda superpozycji rozkładów.* Metoda ta oparta jest na twierdzeniu o prawdopodobieństwie całkowitym:

$$f(x) = \int_{-\infty}^{\infty} g_y(x) h(y) dy, \tag{3.20}$$

gdzie:

- dla ustalonego y, $g_y(x)$ jest pewną gęstością prawdopodobieństwa zależną od parametru y,
- $h(y)$ jest gęstością prawdopodobieństwa parametru y.

Algorytm generowania liczby losowej o gęstości prawdopodobieństwa postaci (3.20)

1. Losujemy y zgodnie z rozkładem o gęstości prawdopodobieństwa $h(y)$.
2. Podstawiamy wylosowany parametr y do $g_y(x)$ i losujemy x zgodnie z tak otrzymaną gęstością prawdopodobieństwa.

PRZYKŁAD [69]. Generowanie zmiennych losowych o rozkładzie zadanym funkcją gęstości prawdopodobieństwa $f(x)$ postaci

$$f(x) = n \int_1^\infty y^{-n} e^{-xy}\, dy \quad \text{dla } x > 0, \tag{3.21}$$

gdzie $n \geq 1$ jest stałą; wówczas

$$g_y(x) = y e^{-yx}, \quad x > 0, \tag{3.22}$$

ma rozkład wykładniczy, natomiast parametr y ma rozkład o gęstości prawdopodobieństwa

$$h(y) = n y^{-(n+1)}, \quad y > 1. \tag{3.23}$$

Zmienne losowe o rozkładzie (3.23) można generować przez odwracanie dystrybuanty,

$$y = (1 - R)^{-1} = R^{-1}, \tag{3.24}$$

gdzie R jest zmienną losową o rozkładzie równomiernym $U(0,1)$.

3.3.4. *Inne metody.* Dla pewnych typów rozkładów możliwe są specyficzne metody generacji liczb pseudolosowych. Na przykład, liczby losowe o rozkładzie normalnym można generować korzystając z następujących dwóch twierdzeń.

TWIERDZENIE. *Niech R_1, R_2 będą dwiema niezależnymi zmiennymi losowymi o rozkładzie równomiernym $U(0,1)$. Wówczas zmienne losowe X_1, X_2 zdefiniowane jako*

$$\begin{aligned} X_1 &= \sqrt{-2 \ln R_1}\, \cos 2\pi R_2, \\ X_2 &= \sqrt{-2 \ln R_1}\, \sin 2\pi R_2, \end{aligned} \tag{3.25}$$

są niezależne i mają jednakowy rozkład normalny $N(0,1)$.

TWIERDZENIE. *Jeżeli U_1, U_2 są niezależnymi zmiennymi losowymi o rozkładzie równomiernym na przedziale $(-1,1)$, to przy spełnieniu warunku $U_1^2 + U_2^2 \leq 1$ zmienne losowe*

$$X_1 = U_1 \sqrt{\frac{-2 \ln (U_1^2 + U_2^2)}{U_1^2 + U_2^2}}, \quad X_2 = X_1 \frac{U_2}{U_1}, \tag{3.26}$$

są niezależne i mają jednakowy rozkład normalny $N(0,1)$.

Zmienne losowe o rozkładzie normalnym można z dobrym skutkiem uzyskiwać, stosując centralne twierdzenie graniczne dla (praktycznie biorąc kilku) niezależnych zmiennych losowych o rozkładzie równomiernym.

3.4. Rozkłady dyskretne. Rozkłady dyskretne mają takie zmienne losowe, które przyjmują skończoną lub przeliczalną liczbę wartości. Dla tych rozkładów najprostszą metodą generowania zmiennych losowych jest wykorzystanie metody odwracania dystrybuant.

Algorytm (*Metoda odwracania dystrybuanty rozkładu dwupunktowego o jednakowym prawdopodobieństwie każdej z wartości a i b*)

1. Generujemy liczbę losową R o rozkładzie równomiernym $U(0,1)$.

2. Jeżeli $R < 0{,}5$, to przyjmujemy $X = a$; jeżeli $R > 0{,}5$, to przyjmujemy $X = b$.

Dla zmiennych losowych przyjmujących skończoną liczbę wartości można również skorzystać z następującego algorytmu, odpowiadającego modelowi ruletki.

Przykład. Zmienna losowa przyjmuje wartości a, b, c, d z prawdopodobieństwami

$$P(a) = 0{,}50, \qquad P(b) = 0{,}25,$$
$$P(c) = 0{,}10, \qquad P(d) = 0{,}15.$$

Budujemy tablicę 100 elementów T, w której umieszczamy, w dowolny sposób, 50 symboli a, 25 symboli b, 10 symboli c i 15 symboli d. Losujemy liczbę naturalną i z przedziału $[1, 100]$, a następnie wybieramy i-ty element tablicy T.

Przy generowaniu ciągu zmiennych losowych o pewnych rozkładach dyskretnych można niekiedy skorzystać z generatora liczb losowych o rozkładzie nierównomiernym, o ile jest on związany z zadanym rozkładem dyskretnym. Z sytuacją taką mamy do czynienia dla pary: rozkład Poissona – rozkład wykładniczy.

Przykład. Celem jest wygenerowanie ciągu niezależnych zmiennych losowych $\{X_j,\ j = 1, 2, \ldots\}$ o jednakowych rozkładach Poissona z wartością średnią c:

$$P(X_j = k) = \frac{c^k}{k!} e^{-c}, \qquad k = 0, 1, 2, \ldots \tag{3.27}$$

W tym celu dla każdego X_j generujemy niezależne liczby losowe $Y_1^{(j)}, Y_2^{(j)}, Y_3^{(j)}, \ldots$ o rozkładzie wykładniczym z wartością oczekiwaną 1. Wybieramy takie n, dla którego

$$\sum_{i=1}^{n} Y_i^{(j)} \le c < \sum_{i=1}^{n+1} Y_i^{(j)}, \tag{3.28}$$

i przyjmujemy

$$X_j = n. \tag{3.29}$$

To samo można uzyskać inną metodą. Ponieważ $Y_i^{(j)} = -\ln R_i^{(j)}$, gdzie $R_i^{(j)}$ jest zmienną losową o rozkładzie równomiernym $U(0,1)$, więc wartość n w wyrażeniu (3.29) możemy wyznaczyć z warunku

$$\prod_{i=1}^{n+1} R_i^{(j)} < e^{-c} \leq \prod_{i=1}^{n} R_i^{(j)}. \tag{3.30}$$

W (3.28) i (3.30) przyjmujemy, że zawsze

$$\sum_{i=1}^{0} Y_i^{(j)} = 0, \qquad \prod_{i=1}^{0} R_i^{(j)} = 1. \tag{3.31}$$

3.5. Zależne zmienne losowe. Kolejnym ważnym zagadnieniem związanym z generowaniem liczb losowych jest otrzymywanie ciągów zależnych liczb (zmiennych) losowych. Zadanie to jest w istocie zadaniem uzyskiwania realizacji procesów stochastycznych (dokładniej mówiąc – szeregów czasowych) i ściśle wiąże się z problemami symulacji zjawisk fizycznych. Omówienie tego zagadnienia wykracza poza zakres niniejszego opracowania. Zauważmy jedynie, że wstępnym krokiem do uzyskania ciągu zależnych zmiennych losowych jest wygenerowanie ciągu zmiennych niezależnych. W kolejnym kroku dokonywane jest takie ich przekształcenie, że uzyskany ciąg posiada cechę zależności. W szczególności, dla rozkładu gaussowskiego uzyskanie zmiennych zależnych sprowadza się do transformacji liniowej niezależnych zmiennych losowych, przy czym macierz transformacji liniowej jest jednoznacznie wyznaczona przez postulowaną macierz korelacji zależnych zmiennych losowych (por. np. [59]).

W szczególnym przypadku stacjonarnych (w szerszym sensie) ciągów zależnych zmiennych losowych, czyli szeregów czasowych, ich generowanie sprowadza się do wykonania dwóch kroków:

1. wygenerowania ciągu niezależnych zmiennych losowych (liczb losowych),
2. na podstawie znajomości funkcji korelacyjnej lub gęstości widmowej poszukiwanego szeregu czasowego, wygenerowania realizacji tego szeregu.

Stosowane tu metody to (por. np. [21], [39]):

- metoda AR (autoregresji);
- metoda MA (ruchomej średniej);
- metoda ARMA (połączenie metod AR i MA).

Informacje o generowaniu zależnych zmiennych losowych znaleźć można również w [69].

4. GENEROWANIE CIĄGÓW BITÓW

4.1. Uwagi wstępne. Wszelkie liczby w komputerze zapisane są w postaci binarnej, na takich też liczbach wykonywane są operacje arytmetyczne. Wynika z tego, że komputerowa generacja liczb losowych o rozkładzie równomiernym jest w istocie generacją losowych ciągów bitów spełniających określone warunki. Zatem generowanie liczb losowych to generowanie losowych bitów. Operacje arytmetyczne nie muszą tu być jedynym źródłem losowości. Na przykład C. Randhakrishna Rao w swojej książce [48] podał przykłady dwóch oryginalnych metod generacji bitów. Pierwsza polegała na losowaniu ze zwracaniem kul białych i czarnych z worka, w którym znajdowała się ich równa liczba, i odpowiednim przyporządkowaniu zer i jedynek wynikom losowania. W drugiej metodzie obserwowano kolejne urodzenia chłopców i dziewczynek w lokalnym szpitalu (w Indiach może to być całkiem wydajne źródło bitów, ponieważ urodziny dziecka następują tam w każdej sekundzie) i oznaczano je, odpowiednio, przez „0" i „1". Obie metody dały rezultaty spełniające testy statystyczne sprawdzające równomierność rozkładu i niezależność kolejnych bitów, mogłyby zatem być stosowane w praktyce, choćby do tworzenia tablic liczb losowych.

Przedstawionym wyżej metodom generowania bitów nie należy jednak wróżyć wielkiej przyszłości, chociaż losowe ciągi bitów znalazły w ostatnich latach szerokie zastosowania. A wszystko zaczęło się w roku 1917, kiedy to Gilbert S. Vernam, pracownik firmy AT&T, opracował metodę szyfrowania sygnałów telegraficznych zapisanych na taśmie perforowanej. W systemie tym tekst depeszy zapisany był na taśmie papierowej w formie odpowiadających literom sekwencji otworów i nieprzebitych miejsc taśmy (kod Baudota). Dodatkowo przygotowywano taśmę z losowo wygenerowanym ciagiem otworów i zaopatrywano w nią nadawcę i odbiorcę depeszy. Przed wysłaniem informacji łączono zapis na obu taśmach (zgodnie z obowiązującą dziś metodą dodawania bitów modulo 2, przyjmując, że otwór to 1, a nieprzebite miejsce to 0), otrzymując nową taśmę kierowaną do wysłania. U odbiorcy deszyfrowano otrzymaną taśmę wykonując operację odwrotną do tej, którą wykonał nadawca i następnie drukując odszyfrowaną depeszę.

Szyfr Vernama jest do dziś jedynym w pełni bezpiecznym (tzw. udowadnialnie bezpiecznym, czyli takim, dla którego istnieje matematyczny dowód bezpieczeństwa) szyfrem symetrycznym, pod warunkiem, że zastosowany w nim strumień bitów jest idealnym szumem binarnym, czyli takim ciągiem bitów „0" lub „1", o którym możemy powiedzieć, że

- bity są losowane niezależne od siebie,
- bity „0" i „1" są losowane z równym prawdopodobieństwem.

Idealnym wzorcem takiego ciągu jest seria niezależnych rzutów monetą symetryczną, z odpowiednim przyporządkowaniem zer i jedynek obu stro-

nom monety. W praktyce poszukujemy bardziej efektywnych źródeł takich strumieni niż wzorcowe rzuty monetą.

W zastosowaniach profesjonalnych o wymaganym dużym stopniu poufności i zarazem o stosunkowo niewielkiej intensywności (np. korespondencja dyplomatyczna) wykorzystywane są metody podobne do oryginalnego pomysłu Vernama. Na taśmie magnetycznej, dostarczanej następnie w bezpieczny sposób do obu korespondujących stron, zapisuje się ciąg bitów (uzyskany, na przykład, z generatora fizycznego i wszechstronnie statystycznie przetestowany), który po jednokrotnym wykorzystaniu jest niszczony. W telekomunikacji, gdzie potrzebne są dziś źródła bitów o intensywności do 1 GB/sek, wykorzystywane są matematyczne metody generowania bitów. Przedstawimy teraz wybrane algorytmy stosowane w tych generatorach.

4.2. Metody teorioliczbowe. Przedstawione w poprzednim rozdziale generatory oparte na schemacie wielokrotnego obliczania reszt modulo M mogą być z powodzeniem wykorzystane do generowania ciągów bitów. W najprostszej wersji wystarczy każdą z wygenerowanych liczb naturalnych zapisać w formie binarnej, uzyskując każdorazowo blok bitów, czyli znaków „0" lub „1". W celu poprawienia własności statystycznych (zgodności z rozkładem równomiernym i zapewnienia niezależności) można wybrać pojedyncze, na przykład najmłodsze, bity z każdej liczby otrzymanej jako reszta modulo M. Odrębnym problemem pozostaje przewidywalność generatorów arytmetycznych, własność dyskwalifikująca ciągi bitów w zastosowaniach kryptograficznych. Dlatego też, w celu uzyskania ciągu nadającego sie do praktycznego wykorzystania należy sięgnąć do arytmetycznych generatorów liczb losowych nie posiadających cechy przewidywalności. A to wymaga skorzystania z pewnych faktów znanych w teorii liczb.

4.2.1. *Generator reszt kwadratowych BBS.* Generator liczb losowych, który nie ma cechy przewidywalności, został opracowany przez trzech autorów (Blum, Blum, Shub) i od ich nazwisk nazwany BBS, por. [3]. Generowanie liczby losowej następuje w wyniku obliczenia reszty kwadratowej według wzoru

$$X_i = X_{i-1}^2 \bmod M \tag{4.1}$$

dla $i = 1, 2, \ldots$ Siła algorytmu polega na odpowiednim wyborze liczby M oraz punktu startowego generatora (ziarna). Budując generator BBS, w pierwszym kroku znajdujemy dwie liczby pierwsze p i q, odpowiednio duże (najlepiej w sposób tajny), i następnie obliczamy

$$M = pq. \tag{4.2}$$

Następnie ustalamy punkt startowy, wybierając liczbę naturalną s z przedziału $(1, M)$, spełniającą warunek $\mathrm{NWD}(s, M) = 1$, i przyjmujemy

$$X_0 = s^2. \tag{4.3}$$

Jeżeli liczby p i q spełniają warunek

$$p \equiv q \equiv 3 \bmod 4 \tag{4.4}$$

(czyli reszta z dzielenia p oraz q przez 4 jest równa 3), to przekształcenie (4.1) definiuje permutację na $(Z_M)^2$ i okres generatora jest maksymalny.

Jeżeli znany jest rozkład liczby M na czynniki pierwsze (p i q nie są generowane w sposób tajny), to kolejne bity generatora BBS można obliczać mniejszym nakładem obliczeniowym, korzystając z zależności (por. [17])

$$B(x^{2^j} \bmod M) = B(x^\alpha \bmod m), \tag{4.5}$$

gdzie $B(\cdot)$ jest operacją obliczania najmłodszego bitu liczby całkowitej oraz

$$\alpha = 2^j \bmod \phi(M), \quad \phi(M) = (p-1)(q-1). \tag{4.6}$$

Dla legalnego użytkownika generatora własność ta jest ułatwieniem pracy. Niestety, jest to również ułatwienie dla potencjalnego napastnika usiłującego złamać szyfr wykorzystujący generator BBS.

4.2.2. *Generator potęgowy RSA.* Generator bitów losowych, którego bezpieczeństwo oparte jest na systemie podobnym do kryptosystemu klucza publicznego RSA, zaproponowany został w pracy [2]. Metoda generowania ciągu bitów $b_1, b_2, \ldots, b_l$ polega na realizacji następujących kroków.

- Generujemy dwie liczby pierwsze p i q w taki sposób, jak dla szyfru RSA (duże liczby pierwsze zbliżonej wielkości – zasady bezpieczeństwa RSA omówione są np. w [38]).
- Obliczamy

$$M = pq, \quad \phi(M) = (p-1)(q-1). \tag{4.7}$$

- Wybieramy losową liczbę w, $1 < w < \phi$ taką, że $\mathrm{NWD}(w, \phi) = 1$.
- Wybieramy losowo $X_0 \in [1, M-1]$.
- Obliczamy, dla $i = 1, 2, \ldots, l$,

$$X_i = X_{i-1}^w \bmod M, \quad b_i = B(X_i). \tag{4.8}$$

Ewentualna możliwość wykorzystania większej liczby bitów w ciągu z każdej generowanej wartości X_i zależy od wielkości liczby M (por. [38]).

4.2.3. *Generator działający w ciałach skończonych.* Kolejnym źródłem silnych kryptograficznie generatorów bitów są generatory bitów oparte na krzywych eliptycznych nad ciałami skończonymi (por. [16], [27], [28]). Krzywe eliptyczne są to zbiory punktów (x, y) spełniających afiniczne równanie Weierstrassa postaci

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{4.9}$$

z dodatkowym punktem O w nieskończoności, gdzie współczynniki równania (4.9), a_1, a_2, a_3, a_4, a_6, są elementami ciała skończonego K. W zbiorze tych

punktów nie ma ustalonego porządku, co zwiększa bezpieczeństwo systemu, niemożliwe jest bowiem jego systematyczne przeszukiwanie. Zbiór punktów wymiernych krzywej eliptycznej tworzy grupę abelową (z działaniem, które nazwiemy dodawaniem +), która jest skończenie generowana. Wybierając odpowiednio ciało K (tak, by miało dostatecznie dużo elementów) i punkt startowy na krzywej eliptycznej (tak, by zawarty był w licznej podgrupie elementów tej krzywej), można skonstruować generator addytywny działający na elementach tej podgrupy i zwracający losowe punkty (X_i, Y_i). Zamieniając je na bity w rozsądny sposób, możemy uzyskać bezpieczny kryptograficznie generator bitów.

Dokładny opis takiego generatora, a zwłaszcza kluczowej dla jego funkcjonowania procedury wyboru punktu startowego, wymaga uprzedniego wprowadzenia wielu pojęć dotyczących krzywych eliptycznych i wykracza poza zakres tego opracowania.

4.2.4. *Ocena jakości generatorów algebraicznych.* O gwarantowanej jakości tego rodzaju generatorów kryptograficznych mówimy zwykle wtedy, gdy spełniają one tak zwany test następnego bitu (the next bit test). Zagadnienie to formułuje się w następujący sposób.

Załóżmy, że dany jest generator, który wytwarza ciąg n bitów $(b_1, b_2, \ldots, b_n)$. Załóżmy również, że dany jest test statystyczny o złożoności obliczeniowej rzędu wielomianowego, którego statystyka $\widehat{B}(\cdot)$ na podstawie znajomości i pierwszych bitów $b_1, b_2, \ldots, b_i$ pozwala przewidzieć bit b_{i+1}. Powiemy, że generator bitów spełnia *test następnego bitu*, gdy dla dostatecznie dużych n, dla wszystkich wielomianów $\varepsilon(n)$ i dla wszystkich liczb całkowitych i z przedziału $[1, n]$, zachodzi nierówność

$$|P(\widehat{B}(b_1, b_2, \ldots, b_i) = b_{i+1}) - 1/2| < 1/\varepsilon(n), \tag{4.10}$$

gdzie $P(\widehat{B}(b_1, b_2, \ldots, b_i) = b_{i+1})$ jest prawdopodobieństwem zdarzenia, że statystyka $\widehat{B}(\cdot)$ poprawnie odgadnie bit b_{i+1} w wygenerowanym ciągu.

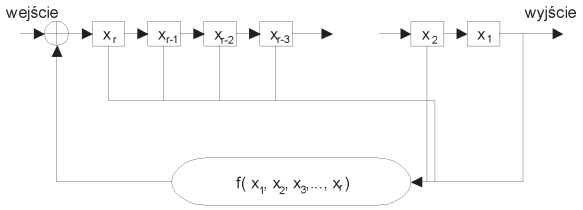
Test następnego bitu jest bardzo mocnym środkiem badania jakości generatorów. Dla generatora bitów losowych zachodzi bowiem (asymptotycznie, gdy n dąży do nieskończoności) równoważność następujących warunków [66]:

- Generator bitów spełnia test następnego bitu.
- Generator bitów spełnia wszystkie testy statystyczne o wielomianowej złożoności obliczeniowej.

4.3. Rejestry przesuwne ze sprzężeniem zwrotnym

4.3.1. *Rejestry nieliniowe ze sprzężeniem zwrotnym.* Bardzo wydajnym źródłem bitów są rejestry przesuwne ze sprzężeniem zwrotnym (FSR – Feedback Shift Register). Są one znane już od dawna; podstawowe wyniki ich do-

tyczące powstały w latach pięćdziesiątych dwudziestego wieku. Działanie rejestrów obejmuje operacje przechowywania i przesuwania bitów w rejestrze, dodawania bitów modulo 2 oraz obliczania wartości funkcji boolowskich. Schemat takiego rejestru przedstawiony jest na rysunku. W najogólniejszej postaci rejestr może być zasilany z zewnątrz ciągiem danych wejściowych (na przykład bitów pochodzących z innego rejestru przesuwnego). Ponadto, stan rejestru (o długości r) w chwili bieżącej zależy od warunku początkowego, czyli od ciągu r bitów znajdujących się w chwili zerowej w komórkach rejestru.



Załóżmy, że dany jest warunek początkowy rejestru,

$$x_1, x_2, x_3, \ldots, x_{r-1}, x_r, \tag{4.11}$$

a w trakcie pracy rejestru jest do niego dostarczany ciąg bitów wejściowych $\{y_{r+1}, y_{r+2}, \ldots\}$. Wyjściem rejestru w kolejnych krokach są wartości bitów początkowych, a po wyczerpaniu ich zasobu, bity uzyskane jako suma modulo 2 bitów wejściowych i wartości funkcji boolowskiej $f(\cdot)$ obliczonej dla argumentów będących bitami pobranymi z wnętrza rejestru. Jak widać na rysunku, bity te są wprowadzane do rejestru przesuwnego kolejno, w miarę jak bity już się tam znajdujące opuszczają rejestr. Po opuszczeniu rejestru przez ostatni bit warunku początkowego stają się one bitami wyjściowymi w krokach $r+1, r+2$ itd. Sprecyzujmy zatem opis działania FSR-u.

Funkcją boolowską $f(x_1, x_2, x_3, \ldots, x_{r-1}, x_r)$ jest odwzorowanie, które każdemu ciągowi bitów $x_1, x_2, x_3, \ldots, x_{r-1}, x_r$, przyjmujących dowolnie wartości „0" lub „1", przyporządkowuje wartość „0" lub „1".W sumie, dla r zmiennych niezależnych jest 2^{2^r} różnych funkcji boolowskich. Tyle zatem możemy utworzyć różnych rejestrów przesuwnych ze sprzężeniem zwrotnym o długości r.

W najogólniejszej postaci rejestru (przedstawionego na rysunku) obliczona wartość funkcji $f(x_1, x_2, x_3, \ldots, x_{r-1}, x_r)$ jest dodawana modulo 2 do ciągu wejściowego $\{y_{r+1}, y_{r+2}, \ldots\}$. Ta operacja dwuargumentowa, nazywana XOR i oznaczana jako $\oplus$, może być zapisana w postaci następującej tabeli:

$$\begin{aligned} 0 \oplus 0 &= 0, \quad 0 \oplus 1 = 1, \\ 1 \oplus 1 &= 0, \quad 1 \oplus 0 = 1. \end{aligned} \tag{4.12}$$

Utworzony ciąg bitów to, kolejno, r bitów warunku początkowego, a następnie bity obliczone z wzoru

$$x_{r+i} = y_{r+i} \oplus f(x_i, x_{i+1}, \ldots, x_{i+r-1}), \qquad i = 1, 2, \ldots \tag{4.13}$$

Rejestry przesuwne są bardzo wydajnym źródłem bitów, jednak w ogólnej postaci równania je opisujące są trudne do analizy i brak jest ich pełnej teorii. Sytuacja znacznie sie upraszcza w przypadku, gdy jako funkcję boolowską $f(\cdot)$ przyjmiemy wyrażenie liniowe.

4.3.2. *Rejestry liniowe ze sprzężeniem zwrotnym.* Załóżmy teraz, że w naszym modelu rejestru funkcja r argumentów, $f(x_1, x_2, x_3, \ldots, x_{r-1}, x_r)$, jest liniową funkcję bitów, czyli wyrażeniem postaci

$$\begin{aligned} f(x_1, x_2, x_3, \ldots, x_{r-1}, x_r) \\ = c_1 x_1 \oplus c_2 x_2 \oplus c_3 x_3 \oplus \ldots \oplus c_{r-1} x_{r-1} \oplus c_r x_r, \end{aligned} \tag{4.14}$$

gdzie współczynniki c_i spełniają warunek

$$c_1, c_2, c_3, \ldots, c_{r-1}, c_r = 0 \quad \text{lub} \quad 1, \tag{4.15}$$

a rejestr pracuje w cyklu zamkniętym, to znaczy nie jest do niego wprowadzany ciąg danych wejściowych. Zauważmy, że istnieje 2^r różnych rejestrów liniowych ze sprzężeniem zwrotnym, czyli LFSR (Linear Feedback Shift Register).

Powinniśmy teraz odpowiedzieć na dwa podstawowe pytania:

- Kiedy ciąg opuszczający LFSR może być uznany za losowy (pseudolosowy)?
- Jaki jest okres tego ciągu?

Jest oczywiste, że gdy rejestr liniowy ma r komórek, to jego okres p spełnia warunek

$$p \leq 2^r - 1 \tag{4.16}$$

(tyle jest wszystkich stanów, w jakich może się znajdować LFSR). Czy ten maksymalny okres jest osiągany? Odpowiedź na to pytanie możemy znaleźć w monografii Golomba [18]. Autor formułuje tam trzy postulaty, które powinień spełniać ciąg bitów wygenerowanych przez LFSR, aby można było go uznać za losowy (pseudolosowy). Warunki te są sprawdzane dla pełnego cyklu (okresu) p generatora. Oto one:

- Liczba bitów „0" i „1" może się różnić co najwyżej o 1.
- Wśród wszystkich serii (kolejno następujących po sobie jednakowych symboli) połowa powinna mieć długość 1, 1/4 długość 2, 1/8 długość 3 itd.; liczba odpowiednich serii zer i jedynek powinna być równa.

- Funkcja autokorelacji ciągu, $C(\tau)$, powinna spełniać następujący warunek:

$$pC(\tau) = \sum_{n=1}^{p} x_n x_{n+\tau} = \begin{cases} p & \text{dla } \tau = 0, \\ K & \text{dla } 0 < \tau < p, \end{cases} \tag{4.17}$$

  dla $K \ll p$ (jest to warunek analogiczny do własności funkcji auto korelacji białego szumu).

Równanie spełniane przez rejestr liniowy (analogiczne do ogólnego równania rejestru (4.13)) można zapisać w postaci

$$x_n = c_1 x_{n-1} + c_2 x_{n-2} + \ldots + c_r x_{n-r} \bmod 2. \tag{4.18}$$

Równaniu (4.18) odpowiada wielomian charakterystyczny $P(z)$ zdefiniowany jako

$$P(z) = 1 - \sum_{i=1}^{r} c_i z^i. \tag{4.19}$$

Dla LFSR oraz odpowiadającego mu wielomianu charakterystycznego $P(z)$ zachodzą dwa następujące twierdzenia (por. [18]):

TWIERDZENIE. *Jeśli wielomian charakterystyczny $P(z)$ rejestru liniowego ze sprzężeniem zwrotnym jest nierozkładalny, to rejestr ten ma maksymalny możliwy okres $p = 2^r - 1$.*

TWIERDZENIE. *Jeśli rejestr liniowy ma maksymalny okres, to spełnia trzy postulaty Golomba dotyczące rozkładu bitów „0" i „1".*

Otrzymujemy zatem odpowiedzi na dwa postawione uprzednio pytania. Podkreślmy tutaj zalety rejestru liniowego ze sprzężeniem zwrotnym jako generatora bitów losowych:

- Jest szybkim i wydajnym (praktycznie dowolnie długi okres) źródłem bitów.
- Znane są jego własności matematyczne, a więc i odporność na potencjalny atak.
- Wygenerowany w nim ciąg ma dobre własności statystyczne.
- Jest łatwy do implementacji programowej i sprzętowej.

Zauważmy, że rejestr liniowy o wymaganych własnościach nie musi być skomplikowany: w sprzężeniu zwrotnym (funkcji boolowskiej $f(\cdot)$) wystarczy uwzględnić jedynie kilka poprzednich wartości bitów (o indeksach odpowiadających niezerowym współczynnikom wielomianu charakterystycznego (4.19), por. np. [47], [65]).

Podstawową wadą rejestru liniowego w zastosowaniach kryptograficznych jest jego przewidywalność. W celu zidentyfikowania współczynników równania (4.18) wystarczy rozwiązać odpowiedni układ równań liniowych wykorzystujący zaobserwowane bity. Dlatego też obecne badania dotyczące

wykorzystania rejestrów przesuwnych dotyczą układów LFSR sprzężonych za pomocą funkcji nieliniowych, rejestrów nieliniowych, a także rejestrów przesuwnych nad alfabetami wieloznakowymi (uogólnionych rejestrów przesuwnych).

4.4. Metody kryptograficzne

4.4.1. *Podstawy matematyczne.* Współczesne podejście do generowania ciągów bitów do celów kryptograficznych w pełni korzysta z terminologii stosowanej w kryptografii. Podstawowym elementem konstrukcji algorytmu jest funkcja postaci

$$F : K \times D \to O, \tag{4.20}$$

gdzie symbole K, D i O oznaczają odpowiednio:

$K = \{0,1\}^k$ – przestrzeń kluczy (parametry F),
$D = \{0,1\}^l$ – przestrzeń elementów dziedziny (dziedzinę F),
$O = \{0,1\}^L$ – przestrzeń elementów obrazu (przeciwdziedzinę F).

Ponadto, wprowadzamy oznaczenia dla typowych operacji wykonywanych w celu wygenerowania ciągu bitów. Tak więc symbol

$$K \xleftarrow{R} \{0,1\}^k \tag{4.21}$$

oznacza wybór losowego klucza o długości k, a symbol

$$f \xleftarrow{R} F \tag{4.22}$$

oznacza wybór funkcji pseudolosowej (takiej funkcji, której działania nie można odróżnić od działania funkcji losowej)

$$f_R : D \to O. \tag{4.23}$$

Operacja (4.22) jest równoważna złożeniu następujących dwóch operacji:

$$K \xleftarrow{R} \{0,1\}^k \tag{4.24}$$

i

$$f_K(\cdot) = F(K, \cdot), \tag{4.25}$$

czyli uzyskaniu (w sposób losowy i tajny) klucza i podstawieniu wartości tego klucza do odwzorowania (4.20).

Wśród wszystkich funkcji pseudolosowych należy wyróżnić ich klasę zwaną permutacjami. Z permutacją $f(\cdot)$ mamy do czynienia, gdy

$$D = O, \quad \text{czyli} \quad l = L, \tag{4.26}$$

a ponadto

$$f : D \to O \tag{4.27}$$

jest bijekcją.

Uwaga. Permutacją (bijekcją) jest każdy szyfr blokowy (por. [38]).

4.4.2. *Generator bitów pseudolosowych.* Generatorem bitów pseudolosowych będziemy nazywali każde odwzorowanie G postaci

$$G : \{0,1\}^k \to \{0,1\}^m, \qquad m \gg k. \tag{4.28}$$

Uwaga. Dla przestrzeni dziedziny operatora G o wymiarze k (pełniącej rolę ziarna) ciągów pseudolosowych może być co najwyżej 2^k.

Po zdefiniowaniu generatora bitów musimy odpowiedzieć na pytanie: Kiedy generator G może być uznany za pseudolosowy?

W celu zdefiniowania pojęcia pseudolosowości wprowadzamy nowy schemat badania jakości generatora. Niech A oznacza algorytm dowolnego ataku na bezpieczeństwo generatora. Atakiem jest, z założenia, test statystyczny zwracający wartość 0 lub 1. Wprowadzamy oznaczenie

$$\mathrm{Succ}_G^{prg}(A) = \Pr\{A(y) = 1 : s \xleftarrow{R} \{0,1\}^k, y \leftarrow G(s)\} \tag{4.29}$$

dla prawdopodobieństwa sukcesu, czyli uzyskania przez test A odpowiedzi 1, na podstawie ciągu bitów y wytworzonego przez generator G przy założeniu, że jego warunek początkowy (ziarno) był ciągiem losowym, a oznaczenie

$$\mathrm{Succ}_{Rand}^{prg}(A) = \Pr\{A(y) = 1 : y \xleftarrow{R} \{0,1\}^m\} \tag{4.30}$$

dla prawdopodobieństwa sukcesu na podstawie czysto losowego ciągu bitów o takiej samej długości. Wprowadzamy również oznaczenie dla różnicy prawdopodobieństw (4.29) i (4.30),

$$\mathrm{Adv}_G^{prg}(A) = \mathrm{Succ}_G^{prg}(A) - \mathrm{Succ}_{Rand}^{prg}(A), \tag{4.31}$$

oraz

$$\mathrm{Adv}_G^{prg}(t) = \max_A\{\mathrm{Adv}_G^{prg}(A)\} \tag{4.32}$$

dla maksimum tej różnicy po wszystkich algorytmach ataku A, których czas działania jest mniejszy niż t. Wyrażenie $\mathrm{Adv}_G^{prg}(t)$ jest miarą jakości generatora bitów pseudolosowych; wyraża liczbowo maksymalną przewagę, jaką może mieć ciąg losowy (o skończonej długości) nad ciągiem pseudolosowym pochodzącym z generatora G. Mając taką informację, możemy zdefiniować ciąg ciąg pseudolosowy bitów.

Definicja. Ciąg bitów wygenerowanych przez generator G jest *pseudolosowy*, gdy $\mathrm{Adv}_G^{prg}(t)$ zdefiniowane w (4.32) jest małe dla rozsądnych wartości t.

4.4.3. *Przykładowe algorytmy generowania.* Kryptograficzne generatory bitów losowych powstają na ogół w wyniku wielokrotnego powtarzania działania funkcji postaci

$$F : \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l \tag{4.33}$$

gdzie $F(\cdot,\cdot)$ jest elementem kryptograficznym (nazywanym często prymitywem kryptograficznym). Najczęściej jest to jeden z następujących elementów:

- Szyfr blokowy. Najczęściej jest to powszechnie stosowany szyfr blokowy o dużym okresie, na przykład DES [38] lub nowy algorytm Rijndael (do znalezienia w internecie). Może to być również algorytm nietypowy (także tajny), na przykład szyfr znakowy wykorzystujący grafowy zapis klucza [45].
- Dowolna permutacja. Wskazane jest wykorzystywanie permutacji posiadających długie cykle.
- Funkcja pseudolosowa (por. np. [49]).

Generowanie bitów polega na obliczaniu wartości funkcji dla zmieniających się wartości argumentów i sekwencyjnym ustawianiu wyników we wspólnym ciągu. W modelu tym możliwe są różne schematy zmian parametrów w trakcie generacji. Za pracą [17] podajemy cztery takie algorytmy (przyjęto oznaczenie $F(k,s) = F_k(s)$).

Algorytm G1:

$$G_1(s) = F_1(s)\|F_2(s)\|\ldots\|F_n(s), \tag{4.34}$$

gdzie ziarno s ma długość l, a wygenerowany ciąg ma długość $m = nl$.

Algorytm G2:

$$G_2(s) = F_s(1)\|F_s(2)\|\ldots\|F_s(n), \tag{4.35}$$

gdzie ziarno s ma długość k, a wygenerowany ciąg ma długość $m = nl$.

Algorytm G3:

$$G_3(s) = F_s(s)\|F_s(s+1)\|\ldots\|F_s(s+n-1), \tag{4.36}$$

gdzie ziarno s ma długość $l = k$, wygenerowany ciąg ma długość $m = nl$. W wyrażeniu (4.36) argumenty funkcji $F_s(\cdot)$ są dodawane modulo $2^{|s|} = 2^k$.

Algorytm G4:

$$G_4(s) = F_s(1)\|F_{s+1}(2)\|\ldots\|F_{s+n-1}(n), \tag{4.37}$$

gdzie ziarno s ma długość k, wygenerowany ciąg ma długość $m = nl$. W wyrażeniu (4.37) klucze s są dodawane modulo 2^k.

W pracy [17] udowodniono następujące własności powyższych generatorów:

- Algorytm G1 jest asymptotycznie przewidywalny, niezależnie od postaci funkcji F.
- Bezpieczeństwo algorytmów G2, G3 i G4, w ogólnym przypadku, nie jest określone.

- Algorytm G2 jest udowadnialnie bezpieczny (w sensie powyższej definicji), gdy F jest funkcją pseudolosową (funkcja F sama jest bezpiecznym elementem kryptograficznym).

W praktyce kryptograficznej istnieją zalecenia normowe dotyczące wykorzystania odpowiednich elementów kryptograficznych i trybów pracy w pseudolosowych generatorach bitów [15].

Obecne i przyszłe badania dotyczące kryptograficznych metod generacji bitów koncentrują się, z jednej strony, na trybach pracy generatorów i wykorzystaniu w nich znanych szyfrów blokowych, z drugiej zaś, na otrzymywaniu nowych konstrukcji funkcji pseudolosowych i skonstruowaniu generatora udowadnialnie bezpiecznego.

4.5. Wykorzystanie układów dynamicznych do generacji bitów. Nową metodą generowania ciągów bitów jest wykorzystanie dyskretnych układów dynamicznych. Metoda ta ma wszelkie zalety generatora algorytmicznego, to znaczy pozwala tworzyć powtarzalne ciągi bitów oraz poddaje się badaniu za pomocą metod matematycznych (stwarza zatem szansę opracowania generatora udowadnialnie bezpiecznego). Z drugiej strony, poprzez naturalne związki matematycznych układów dynamicznych i odpowiednich procesów fizycznych urzeczywistnia możliwość realizacji sprzętowej takiego generatora (pozwala zaprojektować generator fizyczny o powtarzalnych trajektoriach).

Obiektem niezbędnym do skonstruowania chaotycznego generatora bitów losowych jest układ dynamiczny. *Układem dynamicznym* będziemy nazywali parę (S, F), gdzie S jest przestrzenią stanów (zazwyczaj przestrzenią metryczną), natomiast $F : S \to S$ jest odwzorowaniem mierzalnym będącym generatorem półgrupy iteracji. *Trajektorią* startującą ze stanu początkowego s_0 nazwiemy ciąg $\{s_n\}_{n=0}^{\infty}$ elementów przestrzeni stanów S uzyskany przez następujące iteracje:

$$s_{n+1} = F(s_n), \quad n = 0, 1, 2, \ldots \tag{4.38}$$

Dla celów konstrukcji generatora będziemy wymagali, żeby układ dynamiczny (S, F) był chaotyczny, a zatem miał cechę silnej wrażliwości trajektorii na zmianę warunków początkowych [54]. W literaturze stosowanych jest wiele precyzyjnych matematycznie definicji chaosu [8]. Najbardziej rozpowszechniona korzysta z pojęcia *wykładników Lapunowa*, zdefiniowanych jako

$$\lambda_{s,v} \equiv \lim_{n\to\infty} \frac{1}{n} \ln \|DF^n(s)(v)\|, \tag{4.39}$$

gdzie $\| \ \|$ jest normą w przestrzeni stycznej w punkcie $s \in S$, v jest elementem tej przestrzeni, natomiast $DF^n(s)(v)$ oznacza pochodną Frécheta n-tej iteracji F w punkcie s w kierunku v. Układ dynamiczny (S, F) jest

chaotyczny w pewnym obszarze, gdy dla prawie wszystkich punktów tego obszaru ma on co najmniej jeden dodatni wykładnik Lapunowa (gdy ma ich więcej niż jeden, nazywamy go hiperchaotycznym). „Prawie wszędzie" jest tu rozumiane w sensie pewnej miary niezmienniczej równoważnej mierze Lebesgue'a, to znaczy takiej miary skończonej μ na $\sigma(S)$ (σ-algebrze podzbiorów mierzalnych przestrzeni S), która spełnia warunek F-niezmienniczości:

$$\forall A \in \sigma(S), \quad \mu(A) = \mu(F^{-1}(A)), \tag{4.40}$$

oraz dla której istnieje dodatnia ograniczona funkcja rzeczywista g na S taka, że

$$\forall A \in \sigma(S), \quad \mu(A) = \int_A g(s)\, ds.$$

Drugą cechą układu niezbędną do konstrukcji generatora bitów losowych jest mieszanie. Powiemy, że układ dynamiczny (S, F) jest *mieszający*, gdy dla każdego $A, B \in \sigma(S)$,

$$\lim_{n\to\infty} \mu(F^{-n}(A) \cap B) = \mu(A)\mu(B). \tag{4.41}$$

W równaniu (4.41), $F^{-n}(A)$ oznacza przeciwobraz zbioru A pod działaniem n-tej iteracji odwzorowania F. Oznacza to, że wielokrotne działanie operatora F^{-1} uniezależnia (w sensie probabilistycznym) zbiory (zdarzenia) A i B (por. [51]). Jeżeli dodatkowo założymy, że $\mu(S) = 1$ (tj. miara stacjonarna μ jest probabilistyczna), to warunek (4.41) jest równoważny warunkowi

$$\lim_{n\to\infty} \frac{\mu(F^{-n}(A) \cap B)}{\mu(B)} = \frac{\mu(A)}{\mu(S)}, \tag{4.42}$$

co wskazuje, że wielokrotne stosowanie odwzorowania F^{-1} do dowolnego zbioru A równomiernie rozprzestrzenia ten zbiór po całej przestrzeni stanów S.

Rozważmy zatem chaotyczny, mieszający układ dynamiczny (S, F) posiadający unormowaną miarę niezmienniczą μ. Podzielmy przestrzeń stanów S na dwie rozłączne części S_0, S_1, w taki sposób, że $\mu(S_0) = \mu(S_1) = \frac{1}{2}$. Jako ziarno generatora przyjmujemy warunek początkowy $s \in S' \subseteq S$, gdzie S' jest pewnym zbiorem dopuszczalnych warunków początkowych (zazwyczaj $\mu(S') = 1$), a ciąg pseudolosowy otrzymujemy, obserwując trajektorię otrzymaną przez iterację F, to znaczy ciąg $s_n := F^n(s)$. Jako n-ty bit b_n przyjmujemy „0", gdy $F^n(s) \in S_0$, lub „1" w przeciwnym wypadku. W ten sposób otrzymujemy nieskończony ciąg bitów $G(s)$, czyli odwzorowanie

$$G : S' \to \prod_{i=1}^{\infty} \{0, 1\}, \tag{4.43}$$

zdefiniowane jako

$$G(s) = \{b_i(s)\}_{i=1,2,\ldots} = \{b_1(s), b_2(s), \ldots\}, \tag{4.44}$$

gdzie $\prod_{i=1}^{\infty}\{0,1\}$ jest iloczynem kartezjańskim przeliczalnej liczby zbiorów dwuelementowych $\{0,1\}$. Można teraz pokazać, że tak zdefiniowany generator bitów ma podstawowe własności wymagane od generatora: jednoznaczną zależność ciągu od warunku początkowego, równe prawdopodobieństwo wystąpienia „0" i „1" oraz (asymptotycznie) statystyczną niezależność bitów (por. [57], [58]). Fakty te można zapisać w postaci następujących trzech twierdzeń.

TWIERDZENIE. *Dla każdego $s \in S'$ spełniony jest warunek*

$$\mu(G^{-1}(\{b_i(s)\})) = 0. \tag{4.45}$$

Twierdzenie to mówi, że jeżeli weźmiemy dwa różne ziarna (warunki początkowe układu dynamicznego), to w wyniku otrzymamy (z prawdopodobieństwem 1) dwa różne ciągi bitów. W rzeczywistości, w wyniku chaotyczności układu dynamicznego, nawet dla dwóch bardzo bliskich warunków początkowych otrzymane ciągi bitów będą się znacznie różniły.

Ponieważ układ dynamiczny jest mieszający, jest on również ergodyczny. Twierdzenie ergodyczne Birkhoffa–Chinczyna ma dla naszego układu postać

TWIERDZENIE.

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n-1}\chi_{S_0}(F^i(s)) = \int_S \chi_{S_0}\,d\mu = \mu(S_0), \tag{4.46}$$

gdzie χ_{S_0} jest funkcją indykatorową zbioru S_0.

A zatem, ponieważ przyjęliśmy, że $\mu(S_0) = 1/2$, wnioskujemy, że w wygenerowanym ciągu bitów $G(s) = \{b_i(s)\}_{i=1,2,\ldots}$, dla każdego warunku początkowego s, średnia liczba bitów „0" dąży do $1/2$. Ten fakt ma również miejsce dla każdego (losowo wybranego) podciągu $(b_{k_n})_{n=1,2,\ldots}$.

Innym efektem warunku mieszania jest asymptotyczna niezależność bitów, którą można ująć w formie następującego twierdzenia.

TWIERDZENIE. *Jeżeli układ dynamiczny (S,F) spełnia warunek mieszania, to istnieje taka liczba naturalna k, że dla każdego $s \in S'$, bity b_i oraz b_{i+k} są (asymptotycznie przy rosnącym k) niezależne dla $i = 1,2,\ldots$*

Zatem, biorąc jako generator losowych bitów zmodyfikowany układ dynamiczny $(S', H_k^1) := (S', F^k)$, dla dostatecznie dużego k uzyskujemy źródło bitów spełniających nasze podstawowe wymagania, to znaczy niezależnych i o rozkładzie równomiernym. Spełnienie dodatkowych wymagań statystycznych można zagwarantować przez wybór odpowiedniego odwzorowania F i odpowiedniego podziału przestrzeni stanów na S_0, S_1 (por. [55]).

Chaotyczne generatory bitów, posiadające, w teorii, nieskończony okres i wszelkie własności idealnych generatorów ciągów losowych, w praktyce komputerowej napotykają na trudności wynikające z dyskretnej struktury arytmetyki komputerowej i ze skończonej (lecz dowolnie dużej) długości słowa

komputerowego. Niedogodności te można zredukować poprzez właściwą dyskretyzację układu dynamicznego, wprowadzenie „strefy zakazanej" w pobliżu granicy podziału przestrzeni stanów S na części S_0 i S_1 (por. [7]) lub też wykorzystanie rozwiązywalnych (konstruowalnych) chaotycznych układów dynamicznych, to znaczy takich, dla których rozwiązanie może być przedstawione w jawnej postaci (uzyskując w ten sposób możliwość dodatkowej weryfikacji iterowanej wartości punktu trajektorii układu dynamicznego).

Do rozwiązywalnych chaotycznych układów dynamicznych należą najczęściej prezentowane w literaturze odwzorowanie logistyczne (kwadratowe) i odwzorowanie trójkątne (kawałkami liniowe). Odwzorowanie logistyczne postaci

$$X_{n+1} = 4X_n(1 - X_n) \tag{4.47}$$

generuje ciąg chaotyczny i mieszający. Wyrazy tego ciągu można przedstawić w postaci ($n = 1, 2, \ldots$)

$$X_n = \sin^2(2^n \arcsin \sqrt{X_0}). \tag{4.48}$$

Podobnie, chaotyczny i mieszający układ dynamiczny generowany przez odwzorowanie trójkątne

$$X_{n+1} = \begin{cases} 2X_n & \text{dla } 0 \le X_n < 1/2, \\ 2(1 - X_n) & \text{dla } 1/2 \le X_n \le 1, \end{cases} \tag{4.49}$$

ma rozwiązanie analityczne postaci

$$X_n = \frac{1}{\pi} \arccos(\cos 2^n \pi X_0). \tag{4.50}$$

Układów mających powyższą cechę można znaleźć w literaturze więcej (por. [8], [25], [32]). Rozwiązania wszystkich tych równań można wyrazić przez złożenia funkcji elementarnych. Można je przedstawić w ogólnej postaci ($n = 0, 1, 2, \ldots$)

$$X_n = \Psi(\theta T \kappa^n), \tag{4.51}$$

gdzie:

- $\Psi(t)$ jest pewną funkcją okresową (np. trygonometryczną, eliptyczną, hipereliptyczną, Weierstrassa itd.),
- κ jest liczbą naturalną,
- T jest okresem funkcji $\Psi(t)$.

$X_0 = \Psi(\theta T)$ jest warunkiem początkowym układu chaotycznego (θ jest liczbą rzeczywistą, parametrem definiującym warunek początkowy).

Wykładnik Lapunowa takiego układu jest równy

$$\lambda = \ln \kappa. \tag{4.52}$$

Rozwiązywalne układy dynamiczne pozwalają nie tylko zwiększyć praktyczną powtarzalność (na różnych komputerach) procedury generacji bitów poprzez porównanie wyniku uzyskanego dwiema metodami, lecz również zwiększyć prędkość obliczeń, na przykład przez obliczanie co k-tego stanu i unikanie $k-1$ iteracji odwzorowania.

Układy dynamiczne będące iteracją pewnych odwzorowań mają, z punktu widzenia kryptografii, tę samą wadę co generatory kongruencyjne, to znaczy są przewidywalne. Jak łatwo zauważyć, zbiór możliwych położeń punktów (X_n, X_{n+1}) pokrywa się z wykresem funkcji rzeczywistej $y = F(x)$, gdzie $F(x)$ jest odwzorowaniem generującym układ dynamiczny. Cecha ta jest mniej istotna dla generatora bitów, ponieważ zależność funkcyjna X_{n+1} od X_n jest ukryta przez transformację stanu na bit, niemniej jednak użytkownik takiego generatora byłby spokojniejszy, gdyby używany układ dynamiczny nie miał tej cechy. Remedium może tu być wykorzystanie tzw. konstruowalnych układów dynamicznych.

Zacznijmy od rozwiązywalnego układu dynamicznego, którego odwzorowanie generujące ma postać

$$X_{n+1} = \sin^2(z \arcsin \sqrt{X_n}), \tag{4.53}$$

gdzie parametr z jest liczbą naturalną. Rozwiązanie równania (4.53) ma postać

$$X_n = \sin^2(\pi\theta z^n); \tag{4.54}$$

jego wykładnik Lapunowa jest równy $\lambda = \ln z$, a układ jest chaotyczny dla $z > 1$. Jak widać, odwzorowanie (4.53) jest przewidywalne (w jednym kroku).

Rozważmy teraz dowolny ciąg postaci (4.54). Jeżeli parametr z w (4.54) jest ułamkiem, to ciąg ten jest chaotyczny, jednak odwzorowanie wiążące ze sobą punkty X_n i X_{n+1} jest wielowartościowe, to znaczy nie można przedstawić go w postaci

$$X_{n+1} = F(X_n) \tag{4.55}$$

(por. [19]). W szczególności, gdy z w (4.54) jest ułamkiem postaci

$$z = \frac{p}{q}, \tag{4.56}$$

gdzie p i q są liczbami względnie pierwszymi, to wykres (X_n, X_{n+1}) jest krzywą Lissajous taką, że każdej wartości X_n odpowiada q wartości X_{n+1}, a każdej wartości X_{n+1} odpowiada p wartości X_n. Takie odwzorowanie nie jest przewidywalne. Sytuację jeszcze lepszą pod tym względem otrzymamy dla niewymiernego parametru z w równaniu (4.54). Wówczas liczba wzajemnie odpowiadających sobie punktów jest niemożliwa do określenia.

Bogatym źródłem nieprzewidywalnych chaotycznych układów dynamicznych do celów kryptograficznych są układy dwuwymiarowe (por. np. [4]) lub

o jeszcze większym wymiarze przestrzeni stanów. W tej bardziej złożonej rachunkowo sytuacji zarówno sama idea generacji bitów, jak i uzyskane wyniki teoretyczne pozostają niezmienione.

Zastosowanie układu chaotycznego do generacji bitów losowych urzeczywistnia ideę pewnej „tożsamości" układów chaotycznych i stochastycznych: niemożności odróżnienia takich układów przy dyskretnej przestrzeni obserwacji (por. [60], [61]).

Zauważmy na zakończenie, że na procedurze generowania bitów (por. [29]) nie kończą się możliwości wykorzystania chaosu w kryptografii. Stosowane jest szyfrowanie wiadomości przez ciągłe układy dynamiczne metodami synchronizacji trajektorii i sterowania chaosem [44], [46]. Również dyskretne układy dynamiczne mogą pełnić rolę szyfrów blokowych (por. np. [22], [30], [31]), mogą być zatem także stosowane jako elementy kryptograficzne w procesie generacji bitów metodą kryptograficzną.

5. METODY TESTOWANIA GENERATORÓW

Przedstawiając poszczególne metody generowania ciągów bitów losowych, wspominaliśmy również o kryteriach oceny ich jakości i sugerowanych metodach jej weryfikacji. Metody te były albo dostosowane do szczególnej postaci generatora (np. postulaty Golomba), albo stanowiły trudny do sprawdzenia wzorzec oczekiwań (np. test następnego bitu). W praktyce stosowane są odpowiednio dobrane zestawy testów statystycznych pozwalające zaakceptować lub odrzucić wygenerowany ciąg bitów lub sam generator. Testy powinny wykazać:

- zgodność rozkładu ciągu bitów z postulowanym (równomierność rozkładu);
- losowość rozkładu (brak wzorca);
- wzajemną niezależność bitów (nieprzewidywalność kolejnych bitów).

W literaturze możemy znaleźć ogromną liczbę testów służących do badania generatorów liczb losowych (na przykład w klasycznej książce [26] rozdział dotyczący testowania generatorów jest wielokrotnie większy od opisu metod generacji; por. też [23], [65], [69]). Rozróżnić tu należy metody ogólne, dotyczące badania rozkładów liczb całkowitych w zapisie dziesiętnym, i specyficzne metody służące do badania ciągów binarnych. Przygotowując zestaw testów, należy zwrócić uwagę na to, by pozwalały one wszechstronnie zweryfikować własności statystyczne generatora, a równocześnie ograniczyć ich liczbę tak, by mogły one służyć do weryfikowania ciągów binarnych bezpośrednio przed ich użyciem (czyli w trakcie eksploatacji generatora). Można też przygotować dwa zestawy testów: jeden dla etapu projektowania czy wyszukiwania parametrów generatora i drugi dla okresu normalnej eksploatacji.

Najszerzej stosowanym w świecie narzędziem weryfikacji generatorów (z racji dominowania standardów amerykańskich w dziedzinie ochrony informacji) jest zestaw testów podany przez normę amerykańską FIPS-140-2 [14], dotyczącą bezpieczeństwa modułów kryptograficznych. W badaniu generatorów ciągów bitów losowych norma przewiduje cztery testy istotności. Każdy z nich przeprowadzany jest dla ciągu długości 20000 bitów (w przypadku wykorzystywania dłuższych ciągów przebadane muszą być kolejne podciągi o tej długości). Poziom istotności tych testów, czyli prawdopodobieństwo odrzucenia ciągu mogącego pochodzić z prawidłowego źródła bitów, jest równy 0,0001. A oto owe cztery testy, których spełnienie jest wymagane przez normę.

1. *Test monobitowy.* Jest to test oparty na statystyce chi-kwadrat. Sprawdza on, czy liczba bitów „1" leży w pewnych granicach, zależnych od ogólnej liczby wygenerowanych bitów. Przeprowadzany jest w kolejnych krokach:

- Obliczamy

$$X = \text{liczba jedynek w ciągu 20000 bitów.} \tag{5.1}$$

- Uznajemy, że nie ma podstaw do odrzucenia ciągu, gdy

$$9725 < X < 10275. \tag{5.2}$$

2. *Test pokerowy.* Jest to test losowości wykorzystujący statystykę chi-kwadrat, który sprawdza wymaganie równomierności rozkładu segmentów czterobitowych (alfabet $2^4 = 16$ znaków). Jego kolejne kroki to:

- Dzielimy 20000 bitów na 5000 czterobitowych segmentów.
- Liczymy ilość pojawień się liczb $i = 0, \ldots, 15$ (zapisanych binarnie) wśród tych segmentów; wynik wpisujemy do tablicy $f(i)$.
- Obliczamy statystykę testową

$$X = \frac{16}{5000}\Big(\sum_{i=0}^{15} [f(i)]^2\Big) - 5000. \tag{5.3}$$

- Uznajemy, że nie ma podstaw do odrzucenia ciągu, gdy

$$2{,}16 < X < 46{,}17. \tag{5.4}$$

3. *Test serii.* Jest to element tak zwanej analizy sekwencyjnej ciągu bitów. Serią nazywamy ciąg jednakowych znaków – zer lub jedynek. Sprawdzając wygenerowany ciąg, wymagamy, żeby liczba serii o długości $1, 2, 3, 4, 5$ oraz 6 i większej spełniała określone ograniczenia (takie jak idealny ciąg binarny). Wykonujemy więc kolejno następujące kroki:

- Liczymy liczbę serii o wskazanych długościach.

- Uznajemy, że nie ma podstaw do odrzucenia badanego ciągu, gdy uzyskane liczby serii spełniają warunki podane w tabeli

Długość serii	Liczba wystąpień
1	2343 . . . 2657
2	1135 . . . 1365
3	542 . . . 708
4	251 . . . 373
5	111 . . . 201
≥ 6	111 . . . 201

4. *Test długich serii.* Jest to również element analizy sekwencyjnej. Polega na sprawdzeniu warunku:

- W ciągu 20000 bitów nie powinno być serii o długości 26 i większej.

Zastosowanie powyższego zestawy testów FIPS stanowi pewne minimum przy weryfikacji ciągu bitów. Do dokładniejszej analizy generatorów można przystosować (budując odpowiednie statystyki) każdy z występujących w literaturze rodzajów testów, przy czym badać można zarówno zmienne losowe jednowymiarowe, jak i wektorowe (por. np. [11]). Wśród tradycyjnych testów statystycznych można wyróżnić następujące grupy testów, sprawdzających różne cechy badanej populacji generalnej, a także warunki, w jakich przeprowadzano eksperyment losowy:

- Testy losowości, które pozwalają stwierdzić, czy ciąg wyników pomiarów może być traktowany jako ciąg realizacji zmiennych losowych. Pomagają one ustalić, czy badana próba pomiarów jest reprezentatywna.
- Testy zgodności, pozwalające stwierdzić, jaki rozkład prawdopodobieństwa ma ciąg obserwacji, czy dwie serie pomiarów mają taki sam rozkład itp.
- Testy normalności, pozwalające stwierdzić, czy można uznać, że ciąg obserwacji pochodzi z populacji generalnej mającej rozkład normalny; jest to szczególny rodzaj testów zgodności, wyróżniony ze względu na swoją ważność oraz specyficzne własności obliczeniowe.
- Testy dotyczące parametrów rozkładu, które pozwalają potwierdzić wiarogodność wyestymowanych na podstawie serii pomiarów parametrów rozkładu, takich jak momenty, statystyki pozycyjne itp.
- Testy niezależności, pozwalające stwierdzić na podstawie pomiarów, czy próbkowane (obserwowane) zmienne losowe można uznać za niezależne.

Oprócz ogólnie stosowanych testów statystycznych, do weryfikacji poprawności generatorów bitów losowych służą specjalizowane testy, które możemy nazwać kryptograficznymi. Są to w szczególności:

- Testy widmowe, które badają własności widma Fouriera i Walsha wygenerowanego ciągu (por. np. [5], [64], [67]). Widmo ciągu idealnego, jako „białego szumu", powinno być w przybliżeniu stałe dla wszystkich wartości argumentów.
- Testy złożoności liniowej. Ich celem jest oszacowanie, jaki jest najmniejszy rząd (długość r) rejestru liniowego, z którego mógłby pochodzić badany ciąg bitów (por. [10]).
- Testy złożoności sekwencyjnej (maksymalnej złożoności, profilu złożoności itp.). Ich celem jest stwierdzenie, czy rozkład serii w wygenerowanym ciągu nie odbiega od rozkładu serii pochodzących z idealnego źródła bitów (por. np. [33], [38], [40]).
- Testy entropijne. Badają one informacyjne własności ciagu bitów, na przykład efektywność kodową tych ciągów lub informację wzajemną oddalonych bitów. Najbardziej popularny jest test Maurera (por. [37]), z dużym prawdopodobieństwem wykrywający każde odchylenie statystyki obliczonej dla badanego ciągu od statystyki w pełni losowego ciągu bitów, a zatem, w odróżnieniu od testów klasycznych, wszelkie rodzaje defektów statystycznych. Algorytm testu związany jest z metodami kompresji danych. Polega na podzieleniu długiego ciągu danych na bloki o skończonej długości N (zwykle od 8 do 16 bitów) i zbadaniu ich entropii. Entropia może przyjąć wartość od zera (dla ciągów deterministycznych) do maksymalnej wartości $\log N$ dla ciągów w pełni losowych. Test pozwala wyznaczyć w przybliżeniu entropię oraz efektywną długość klucza (czyli liczbę bitów pochodzących z ciągu w pełni losowego, która ma taką samą entropię jak N bitów pochodzących z badanego generatora), jeżeli generator ma posłużyć jako źródło kluczy kryptograficznych.
- Inne metody testowania. Tutaj należałoby umieścić metody badania, które nie są testami statystycznymi, ale pozwalają ocenić pewne własności generatorów. Na przykład falki (wavelets) są uogólnieniem widma ciągu pozwalającym wykryć jego niestacjonarność (por. [36], [62]). Można również potraktować ciąg bitów jako łańcuch Markowa i estymować odpowiednie prawdopodobieństwa przejścia, poszukując zależności między bitami oddalonymi o stały odstęp.
- Badanie generatorów przez zadania testowe. Dotyczy to generatorów liczb losowych o wszelkich rozkładach. Pozwala sprawdzić praktyczną szybkość działania generatora i przydatność uzyskanego ciągu w konkretnych zastosowaniach. Pozwala też dokonać ocen względnych różnych generatorów (por. np. [63]).

Kończąc tę pracę, przytoczmy przykład praktycznej weryfikacji generatora bitów losowych. Analizując działanie implementacji komputerowej generatora bitów opartego na pewnym konstruowalnym układzie dynamicznym [32], wykorzystaliśmy następującą sekwencję testów dla ciągów o długości 100 MB:

- testy FIPS;
- test entropijny Maurera;
- test złożoności liniowej;
- test widmowy Walsha;
- testy zgodności chi-kwadrat i Kołmogorowa–Smirnowa dla słów różnej długości.

Testy FIPS zaakceptowały wszystkie wygenerowane ciągi bitów, pozostałe testy sporadycznie odrzucały wygenerowaną próbę, przy czym najbardziej restrykcyjne były testy zgodności przeprowadzane dla długich ciągów. Takie wyniki wskazują, że zarówno zestaw testów, jak i ich parametry muszą być dobrane odpowiednio do przeznaczenia badanego ciągu (tj. długości wykorzystywanego ciągu, poziomu bezpieczeństwa, trybu pracy układu kryptograficznego, postulowanego poziomu istotności itp.). Dokładniejsza analiza zagadnienia badania jakości generatorów liczb losowych dla celów kryptograficznych zawarta jest w opracowaniu [21] (powstałym w trakcie realizacji grantu KBN 8T11 D01112 „Zastosowania metod stochastycznych w kryptografii") i w przygotowywanej na jego podstawie monografii.

REFERENCES

[1] M. Abramowitz, I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, National Bureau of Standards, Apllied Mathematics Series 55, 1964.

[2] W. B. Alexi, B. Chor, O. Goldreich, C. P. Schnorr, *RSA and Rabin functions: certain parts are as harg as the whole*, SIAM J. Comput. 17 (1988), 194–208.

[3] L. Blum, M. Blum, M. Shub, *A simple unpredictable pseudo-random number generator*, SIAM J. Comput. 15 (1986), 364–383.

[4] A. Beardon, *Iteration of Rational Functions*, Springer, New York, 1991.

[5] K. G. Beauchamp, *Walsh Functions and their Applications*, Academic Press, London, 1975.

[6] H. Beker, F. Piper, *Cipher Systems: the Protection of Communication*, Wiley, New York, 1982.

[7] E. Bollt, Y.-C. Lai, C. Grebogi, *Coding, channel capacity, and noise resistance in communicating with chaos*, Phys. Rev. Lett. 79 (1997), 3787–3790.

[8] R. Brown, L. O. Chua, *Clarifying chaos: examples and counterexamples*, Internat. J. Bifurcation Chaos 6 (1996), 219–249.

[9] A. Compagner, *Definitions of randomness*, Amer. J. Phys. 59 (1991), 700–705.

[10] C. Ding, G. Xiao, W. Shan, *The Stability Theory of Stream Ciphers*, Springer, Berlin, 1991.

[11] Cz. Domański, *Testy statystyczne*, PWE, Warszawa, 1990.
[12] J. Eichenauer, J. Lehn, *A non-linear congruential pseudo-random number generator*, Statist. Pap. 27 (1986), 315–326.
[13] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer, New York, 1996.
[14] FIPS 140-2, *Security Requirements for Cryptographic Modules*, NIST 1999.
[15] FIPS 186-2, *Digital Signature Standard*, NIST 2000.
[16] J. Gawinecki, J. Szmidt, *Zastosowanie ciał skończonych i krzywych eliptycznych w kryptografii*, Wydawnictwo WAT, Warszawa, 1999.
[17] S. Goldwasser, M. Bellare, *Lecture Notes on Cryptography*, preprint, August 1999 (w internecie).
[18] S. W. Golomb, *Shift Register Sequences*, Holden-Day, San Francisco, 1967.
[19] J. A. Gonzalez, R. Pino, *Chaotic and stochastic functions*, Physica 276A (2000), 425–440.
[20] K. Górski, A. Paszkiewicz, A. Zugaj, Z. Kotulski, J. Szczepański, *Własności ciągów generowanych przez najmniejsze pierwiastki pierwotne liczb pierwszych*, w: Krajowe Sympozjum Telekomunikacji (Bydgoszcz, 1998), Wyd. Inst. Telekomunikacji PW, tom B, 1998, 143–151.
[21] K. Górski, Z. Kotulski, A. Paszkiewicz, J. Szczepański, A. Zugaj, *Generatory losowych ciągów binarnych w kryptografii*, opracowanie, 250 stron, Warszawa, 1999.
[22] T. Habutsu, Y. Nishio, I. Sasase, S. Mori, *A secret key cryptosystem by iterating a chaotic map*, w: Eurocrypt'91, 1991, 127–140.
[23] P. Hellekalek, *Good random number generators are (not so) easy to find*, Math. Comput. Simulation 46 (1998), 485–505.
[24] M. Kac, *What is random?*, Amer. Sci. 71 (1983), 405–406.
[25] S. Katsura, W. Fukuda, *Exactly solvable models showing chaotic bahavior*, Physica 130A (1985), 597–605.
[26] D. E. Knuth, *The Art of Computer Programming—Seminumerical Algorithms*, Vol. 2, 3rd ed., Addison-Wesley, Reading, 1997.
[27] N. Koblitz, *Wykład z teorii liczb i kryptografii*, WNT, Warszawa, 1995.
[28] N. Koblitz, *Algebraiczne aspekty kryptografii*, WNT, Warszawa 2000.
[29] T. Kohda, A. Tsuneda, *Statistic of chaotic binary sequences*, IEEE Transactions on Information Theory 43, (1997), 104–112.
[30] Z. Kotulski, J. Szczepański, *Discrete chaotic cryptography*, Ann. Phys. 6 (1997), 381–394.
[31] Z. Kotulski, J. Szczepański, K. Górski, A. Paszkiewicz, A. Zugaj, *The application of discrete chaotic dynamical systems in cryptography—DCC Method*, Internat. J. Bifurcation Chaos 9 (1999), 1121–1135.
[32] Z. Kotulski, J. Szczepański, K. Górski, A. Górska, A. Paszkiewicz, *On constructive approach to chaotic pseudorandom number generators*, w: Proc. Regional Conference on Military Communication and Information Systems. CIS Solutions for an Enlarged NATO, RCMIS 2000 (Zegrze, 2000), tom 1, 191–203.
[33] L. Kuipers, H. Niederreiter, *Uniform Distribution of Sequences*, Wiley, New York, 1974.
[34] D. H. Lehmer, *Mathematical methods in large-scale computing units*, w: Proc. 2nd Sympos. on Large-Scale Digital Calculating Machinery (Cambridge, MA, 1949), Ann. Comp. Lab. Harvard University 26 (1951), 141–146.
[35] P. L'Ecuyer, *Random Number Generation*, w: J. Banks (ed.), Handbook of Simulation, rozdział 4, Wiley, New York, 1998.

[36] Y. Li, Z. Xie, *The wavelet detection of hidden periodicities in time series*, Statist. Probab. Lett. 35 (1997), 9–23.

[37] U. M. Maurer, *A universal statistical test for random bit generator*, J. Cryptology 5 (1992), 89–105.

[38] A. Menezes, P. van Oorschot, C. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1996.

[39] P. S. Naidu, *Modern Spectrum Analysis of Time Series*, CRC Press, Boca Raton, 1995.

[40] H. Niederreiter, *The linear complexity profile and the jump complexity of keystream sequences*, w: Advances in Cryptology (Aarhus, 1990), Lecture Notes in Comput. Sci. 473, Springer, Berlin, 1991, 174–188.

[41] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, 1992.

[42] H. Niederreiter, *New developments in uniform pseudorandom number and vector generation*, w: H. Niederreiter, P. J.-S. Shiue (red.), Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Lecture Notes in Statist. 106, Springer, Heidelberg, 1995.

[43] E. Ott, C. Grebogi, J. A. Yorke, *Controlling chaos*, Phys. Rev. Lett. 64 (1990), 1196–1199.

[44] U. Parlitz, L. O. Chua, Lj. Kocarev, K. S. Halle, A. Shang, *Transmission of digital signals by chaotic synchronization*, Internat. J. Bifurcation Chaos 2 (1992), 973–977.

[45] A. Paszkiewicz, K. Górski, A. Górska, Z. Kotulski, K. Kulesza, J. Szczepań-ski, *Proposals of graph-based ciphers, theory and implementations*, w: Proc. Regional Conf. on Military Communication and Information Systems. CIS Solutions for an Enlarged NATO, RCMIS 2001 (Zegrze, 2001) (w druku).

[46] L. M. Pecora, T. L. Caroll, *Synchronization in chaotic systems*, Phys. Rev. Lett. 64 (1990), 821–824.

[47] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C. The Art of Scientific Computing*, 2nd ed., Cambridge Univ. Press, Cambridge, 1992.

[48] C. Radhakrishna Rao, *Statystyka i prawda*, PWN, Warszawa, 1994.

[49] O. Reingold, *Pseudo-Random Synthesizers, Functions and Permutations*, The Weizmann Institute of Science, 1998 (praca doktorska).

[50] P. Ribenboim, *Mała księga wielkich liczb pierwszych*, WNT, Warszawa, 1997.

[51] M. Rosenblatt, *Procesy stochastyczne*, PWN, Warszawa, 1967.

[52] R. Rueppel, *Analysis and Design of Stream Ciphers*, Springer, Berlin, 1986.

[53] B. Schneier, *Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C*, WNT, Warszawa, 1995.

[54] H. G. Schuster, *Chaos deterministyczny*, PWN, Warszawa, 1995.

[55] J. Szczepański, Z. Kotulski, *On topologically equivalent ergodic and chaotic reflection laws leading to different types of particle's motion*, Arch. Mech. 50 (1998), 865–875.

[56] J. Szczepański, K. Górski, Z. Kotulski, A. Paszkiewicz, A. Zugaj, *Some models of chaotic motion of particles and their application to cryptography*, Arch. Mech. 51 (1999), 509–528.

[57] J. Szczepański, Z. Kotulski, K. Górski, A. Paszkiewicz, A. Zugaj, *On some models of pseudorandom number generators based on chaotic dynamical systems*, Proc. RCMCIS'99 (Zegrze, 1999), vol. 3, 213–220.

[58] J. Szczepański, Z. Kotulski, *Pseudorandom number generators based on chaotic dynamical systems*, Open Systems Information Dynamics 8 (2001) (w druku).

[59] W. Szczepiński, Z. Kotulski, *Error Analysis with Applications in Engineering*, Lastran Corporation, Rochester, 2000.
[60] T. J. Taylor, *On stochastic and chaotic motion*, Stochastics Stochastics Rep. 43 (1993), 179–197.
[61] T. J. Taylor, *Time series, stochastic and chaotic*, w: W. A. Barnett *et al.* (eds.), Nonlinear Dynamics and Economics (Florence, 1992), Cambridge Univ. Press, Cambridge, 1996.
[62] Y. Wang, *Detection of jumps and curps by wavelets*, Biometrics 82 (1995), 385–397.
[63] S. Wegenkittl, *On empirical testing of pseudorandom number generators*, w: G. De Pietro *et al.* (eds.), Proceedings of the international workshop Parallel Numerics'95, CEI-PACT Project, WP5.1.2.1.2., 1995.
[64] P. D. Welch, *The use of fast Fourier transform for estimation of power spectra: A method based on time averaging over short, modified periodograms*, IEEE Trans. Automatic Control AU-15 (1973), 70–73.
[65] R. Wieczorkowski, R. Zieliński, *Komputerowe metody generacji liczb losowych*, WNT, Warszawa, 1997.
[66] A. C. Yao, *Theory and applications of trapdoor functions*, w: Proc. 23rd IEEE Symposium on Foundations of Computer Science, IEEE, Chicago, 1982, 80–91.
[67] C. K. Yuen, *Testing random number generators by Walsh transform*, IEEE Trans. Computers C-26 (1977), 329.
[68] R. Zieliński, *Metody Monte Carlo*, WNT, Warszawa, 1970.
[69] R. Zieliński, *Generatory liczb losowych*, WNT, Warszawa, 1972.
[70] R. Zieliński, *Wytwarzanie losowości*, Wiad. Mat. 29 (1992), 189–203.

Instytut Podstawowych Problemów Techniki
Polska Akademia Nauk
00-049 Warszawa, ul. Świętokrzyska 21
E-mail: zkotulsk@ippt.gov.pl
URL: http:/www.ippt.gov.pl