Zaawansowane metody kryptografii i ochrony informacji
(MKOI) - Projekt - Etap 2
Prowadzący - mgr inż. Marcin Tunia
Semestr 14Z

Dominik Chmiel
Marcin Łojewski

Temat: Implementacja i porównanie dwóch wybranych generatorów liczb pseudolosowych (Micali-Schnorr, Blum-Blum-Shub, RSA pseudorandom bit generator, Blum-Micali).

Spis treści

Wprowadzenie

Generatorem liczb pseudolosowych nazywamy program, który na podstawie niewielkiej ilości informacji generuje deterministycznie losowy ciąg bitów.

Jednym z elementów jego zastosowania są obliczenia probabilistyczne (np. całkowanie metodą Monte-Carlo, Deska Galtona) [1], które to jednak nie wymagają dużej „siły" generatorów.

Głównym zastosowaniem generatorów licz pseudolosowych jest kryptografia a przede wszystkim synchroniczne szyfrowanie strumieniowe, które wymaga „silnych" generatorów, tak aby informacja przesyłana za jego pomocą pozostała znana tylko dla jej określonych użytkowników [2].

W zaimplementowanym przez nas programie użytkownik powinien móc wprowadzić dowolne parametry początkowe generatorów (zgodne z założeniem danego algorytmu), a następnie przetestować ich wydajność oraz skuteczność.

W naszym projekcie zaimplementujemy dwa generatory liczb pseudolosowych: Generator Blum-Micali oraz Generator RSA. Następnie przeprowadzimy na nich testy statystyczne, test chi-kwadrat, użyjemy testu z pakietu Diehard Runs test oraz za pomocą metody Monte Carlo spróbujemy wyznaczyć wartość liczby pi, którą następnie porównamy z jej prawdziwą wartością (z dokładnością do odpowiedniej liczby miejsc po przecinku).

Opracowanie teoretyczne

Generatory liczb pseudolosowych powinny spełniać następujące warunki:

- Trudne do ustalenia ziarno oraz kolejno generowane bity, przy znanym dotychczasowym wygenerowanym ciągu bitów.
- Jednorodność – W każdym punkcie generowanego ciągu bitów prawdopodobieństwo wystąpienia jedynki lub zera jest takie samo i wynosi 1/2, oczekiwana liczba zer w całym ciągu wynosi około n/2 dla ciągu n bitów.
- Skalowalność – Każdy podciąg ciągu bitów, który uzyskał pozytywny wynik testu jakości, poddany temu samemu testowi powinien również uzyskać wynik pozytywny.
- Zgodność – zachowanie generatora musi dawać podobne rezultaty niezalenie od początkowej wartości lub fizycznego zjawiska będącego źródłem „losowości”.

Dodatkowo generowane ciągi nie są całkiem losowe: jeśli ziarno jest reprezentowane przez k bitów informacji, to generator może wygenerować n-bitowy ciąg jedynie na 2^k sposobów spośród 2^n możliwych .

Generator Blum-Micali

Generator Blum-Micali [3] wykorzystuje trudność w obliczaniu logarytmu dyskretnego.

Na początek wybierane są dwie liczby pierwsze a i p oraz ziarno x_0 (ziarno).

Następnie obliczany jest kolejny wyraz x ze wzoru:

$$x_{i+1} = a^{x_i} \, (mod\ p), dla\ i = 1, 2, 3 \ldots$$

Pseudolosowy ciąg bitów obliczany jest ze wzoru:

$$k_i = 1, jeżeli\ x_i < (p-1)/2$$
$$k_i = 0, w\ przeciwnym\ wypadku$$

Generator RSA

Generator RSA [4] wykorzystuje trudność związaną z faktoryzacją liczb.

Na początek wybierane są dwie liczby pierwsze p i q $(N = p \cdot q)$ oraz liczba e względnie pierwszą z (p – 1)(q – 1).

Następnie wybierana jest losowe ziarno x_0 mniejsze od N, oraz obliczany jest kolejny wyraz:

$$x_{i+1} = {x_i}^e \, (mod\ N)$$

Generowanym bitem jest najmłodszy bit x_i.

Test statystyczny

Testy statystyczne pozwalają na oszacowanie prawdopodobieństwa spełnienia pewnej hipotezy statystycznej w populacji na podstawie próby losowej z tej populacji.

Najczęściej wykorzystywane parametry statystyczne w badaniach właściwości rozkładów liczb pseudolosowych w szczególności do oceny równomierności rozkładów są następujące:

Wartość średnia:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_n$$

Wariancja:

$$D^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Odchylenie standardowe:

$$D = \sqrt{D^2}$$

Odchylenie przeciętne:

$$d = \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}|$$

Gdzie: n – liczba pomiarów, x_i – wartość n-tego pomiaru.

Dla ciągu liczb o rozkładzie równomiernym powyższe wartości powinny być równe:

Wartość średnia:

$$\frac{n_{max} - n_{min}}{2}$$

Wariancja:

$$\frac{(n_{max} - n_{min})^2}{12}$$

Odchylenie standardowe:

$$\frac{n_{max} - n_{min}}{2\sqrt{3}}$$

Odchylenie przeciętne:

$$\frac{n_{max} - n_{min}}{4}$$

Gdzie: n_{max} – maksymalna możliwa wartość do wygenerowana, n_{min} – minimalna możliwa wartość do wygenerowania.

Test chi-kwadrat

Test χ2 jest jednym z tak zwanych testów zgodności. Testy zgodności sprawdzają, czy generator generuje liczby, które są niezależne i mają zadany rozkład.

Wzór na test zgodności chi-kwadrat ma postać:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{{\sigma_i}^2}$$

Gdzie: O_i – wartość mierzona, E_i – wartość teoretyczna (oczekiwana) wynikająca z hipotezy odpowiadająca wartości mierzonej, σ_i – odchylenie standardowe, n – liczba pomiarów.

Runs test

Test ten jest jednym z nieparametrycznych testów losowości próby. Stosujemy go m. in., gdy chcemy sprawdzić, czy wyniki eksperymentu spełniają postulat losowości próby.

Test najpierw przekształca ciąg liczb losowych całkowitych na liczby zmiennoprzecinkowe, zgodnie ze wzorem:

$$n_z = \frac{n}{n_{max} + 1}$$

Gdzie: n_z – liczna zmiennoprzecinkowa z przedziału [0, 1), n – wylosowana liczba, n_{max} – maksymalna możliwa do wylosowania liczba.

Następnie analizowany jest otrzymany ciąg, przechodząc od pierwszej do ostatniej liczby, licząc przy tym gdzie następowały przejścia z wartości mniejszej na większą (run up) i odwrotnie (run down). Przejścia są zliczane dla odpowiedniej sekwencji liczb. Idealny przypadek zakłada równą ilość przejść w górę jak i w dół. [5]

Obliczanie liczby pi metodą MC

Test Pi (Test Π), jest prostym sposobem na sprawdzenie czy liczby losowane przez generator są jednorodnie rozłożone dla danego zakresu. Im równomierniej rozłożone są losowane punkty, tym z większą dokładnością wyznaczmy liczbę pi.

Wyznaczamy wewnątrz kwadratu bardzo dużo losowych punktów, następnie zliczamy te, które wpadają do wnętrza koła wpisanego w kwadrat (promień koła – r, bok kwadratu – 2r). Stosunek liczby punktów zawierających się w kole do wszystkich wylosowanych punktów będzie dążył w nieskończoności do stosunku pola koła do pola kwadratu:

$$\pi = 4\frac{P_{koła}}{P_{kwadrat}} \rightarrow \pi = 4\frac{n_{koło}}{n_{kwadrat}}$$

Gdzie: $n_{koło}$ – liczba punktów w kole, $n_{kwadrat}$ – liczba wszystkich punktów.

Zastosowania praktyczne implementowanego zagadnienia

Generatory liczb pseudolosowych mają swoje zastosowanie w wielu dziedzinach. Jedną z najważniejszych z nich jest z pewnością kryptografia. Wiele algorytmów, protokołów opiera się na liczbach wygenerowanych losowo.

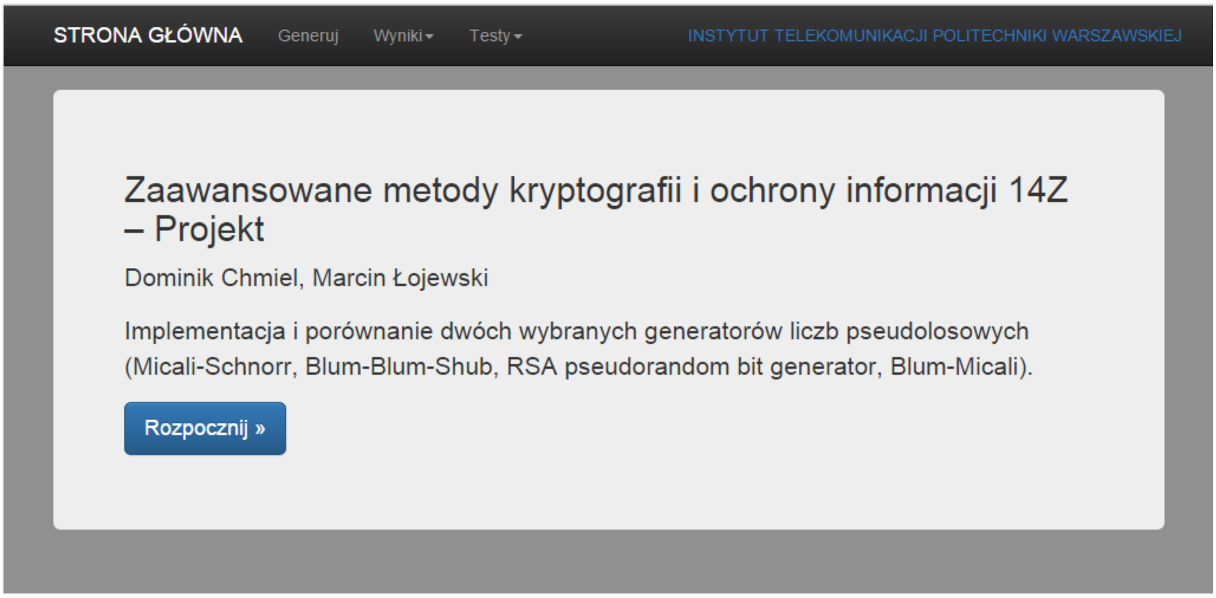
Zaimplementowanych przez nas generator może posłużyć przede wszystkim do celów edukacyjnych.

Instrukcja użytkowania programu

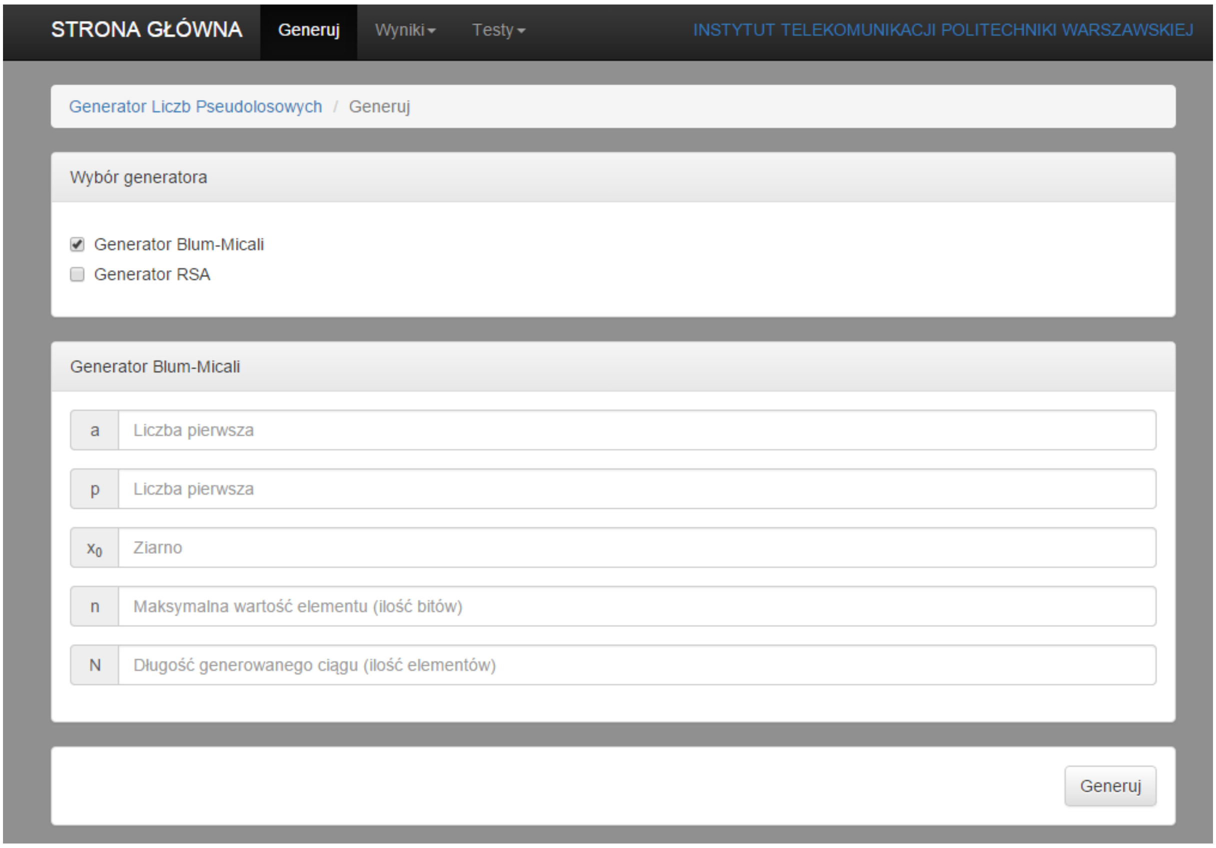
Środowisko uruchomieniowe: CentOS 7 + Apache + PHP.

Aby uruchomić program należy doinstalować w systemie dodatkowe biblioteki: **PHP GMP**, **PHP BC Math** oraz wszystkie zależności, których wymaga framework **Laravel**. Dodatkowo folder **app/storage** powinien mieć prawo zapisu dla użytkownika, który uruchamia skrypty.

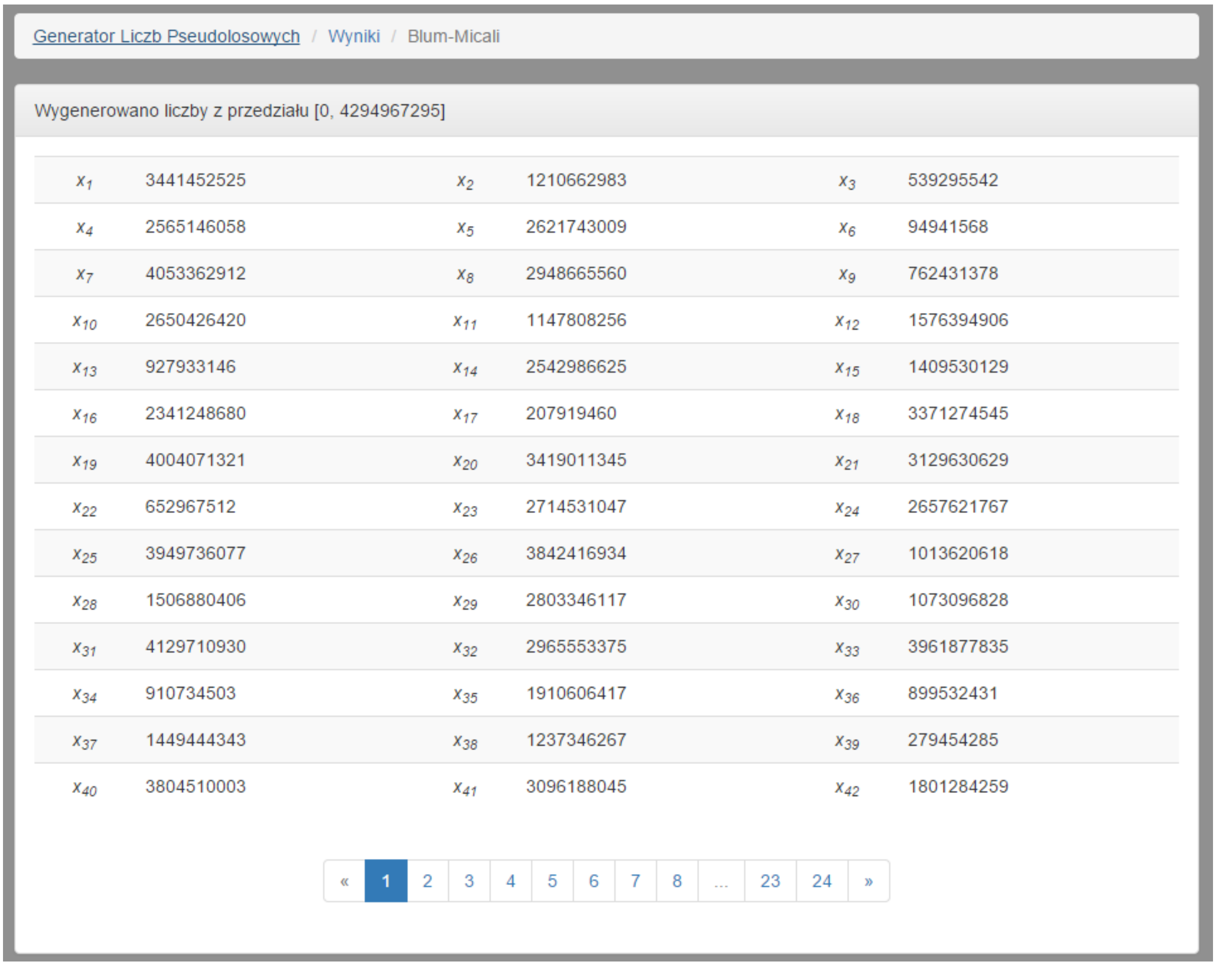
Aplikacja dostępna jest pod adresem *http://vps126353.ovh.net/*.



Do nawigowania aplikacją służy górne menu aplikacji na którym widnieją trzy zakładki: Generuj, Wyniki, Testy. Ta pierwsza dostępna jest zawsze, natomiast dwie pozostałe widoczne są dopiero po wygenerowaniu wyników.



W celu wygenerowania ciągu liczb pseudolosowych (Zakładka Generuj) musimy wybrać odpowiedni generator, a następnie podać odpowiednie dla niego dane wejściowe, które przed wysłaniem żądania na serwer są walidowane zgodnie z ich opisem. Wszystkie istniejące już wyniki dla danego generatora zostaną nadpisane.



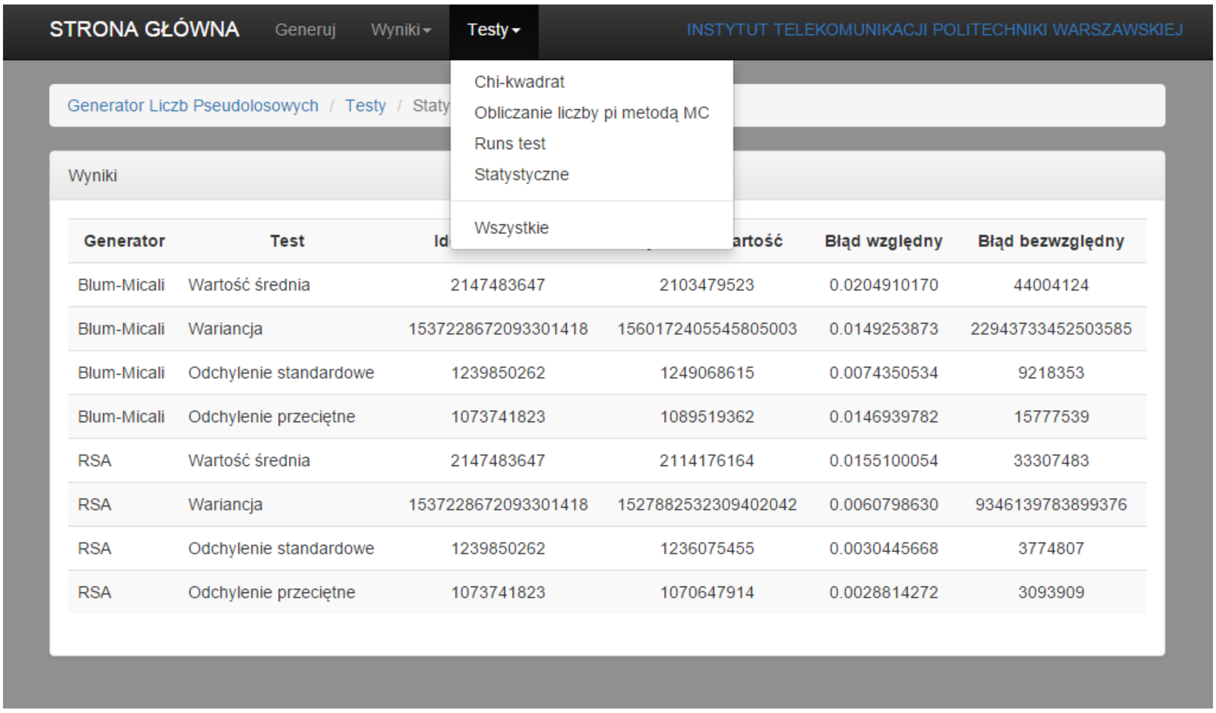
Generator Liczb Pseudolosowych / Wyniki / Blum-Micali

Wygenerowano liczby z przedziału [0, 4294967295]

x_1	3441452525	x_2	1210662983	x_3	539295542
x_4	2565146058	x_5	2621743009	x_6	94941568
x_7	4053362912	x_8	2948665560	x_9	762431378
x_{10}	2650426420	x_{11}	1147808256	x_{12}	1576394906
x_{13}	927933146	x_{14}	2542986625	x_{15}	1409530129
x_{16}	2341248680	x_{17}	207919460	x_{18}	3371274545
x_{19}	4004071321	x_{20}	3419011345	x_{21}	3129630629
x_{22}	652967512	x_{23}	2714531047	x_{24}	2657621767
x_{25}	3949736077	x_{26}	3842416934	x_{27}	1013620618
x_{28}	1506880406	x_{29}	2803346117	x_{30}	1073096828
x_{31}	4129710930	x_{32}	2965553375	x_{33}	3961877835
x_{34}	910734503	x_{35}	1910606417	x_{36}	899532431
x_{37}	1449444343	x_{38}	1237346267	x_{39}	279454285
x_{40}	3804510003	x_{41}	3096188045	x_{42}	1801284259

« 1 2 3 4 5 6 7 8 ... 23 24 »

Zakładka z wynikami prezentuje ciąg wygenerowanych liczb, podzielonych na strony (jedna strona zawiera maksymalnie 42 wyniki).



Generator	Test	Id	artość	Błąd względny	Błąd bezwzględny
Blum-Micali	Wartość średnia	2147483647	2103479523	0.0204910170	44004124
Blum-Micali	Wariancja	1537228672093301418	1560172405545805003	0.0149253873	22943733452503585
Blum-Micali	Odchylenie standardowe	1239850262	1249068615	0.0074350534	9218353
Blum-Micali	Odchylenie przeciętne	1073741823	1089519362	0.0146939782	15777539
RSA	Wartość średnia	2147483647	2114176164	0.0155100054	33307483
RSA	Wariancja	1537228672093301418	1527882532309402042	0.0060798630	9346139783899376
RSA	Odchylenie standardowe	1239850262	1236075455	0.0030445668	3774807
RSA	Odchylenie przeciętne	1073741823	1070647914	0.0028814272	3093909

Zakładka z testami zawiera wyniki testów, które wyliczone zostały na podstawie wygenerowanych wcześniej wyników generatorów pseudolosowych.

Raport z testów aplikacji

Do testowania zaimplementowanych funkcjonalności aplikacji użyty został framework PHPUnit.

Wszystkie testy dostępne są w folderze *App/tests* w dostarczonym kodzie źródłowym aplikacji.

Testy generatorów polegają na ustawieniu wcześniej dobranych ich parametrów, a następnie sprawdzane są generowane przez nie wyniki, które porównywane są z wyliczonymi teoretycznie wartościami dla wcześniej ustalonych parametrów.

Poniżej przedstawiony jest wynik testów:

```
PHPUnit 4.1.6 by Sebastian Bergmann.

Configuration read from /var/www/mkoi/phpunit.xml

........

Time: 822 ms, Memory: 13.75Mb

OK (8 tests, 62 assertions)
```

Jak widać testy zostały pomyślnie zakończone.

Bibliografia

[1] Zbigniew Kotulski, *Generatory liczb losowych: algorytmy, testowanie, zastosowania* [online], Warszawa, Polska Akademia Nauk, 2001 [dostęp: 12.05.2014], Dostępny w Internecie: http://bluebox.ippt.pan.pl/~zkotulsk/MS.pdf.

[2] Paweł Czernik, *Kryptograficzne generatory liczb losowych w rozproszonych systemach pomiarowo-sterujących małej mocy* [online], Warszawa, Instytut Lotnictwa, [dostęp: 12.05.2014], Dostępny w Internecie: http://ilot.edu.pl/prace_ilot/public/PDF/spis_zeszytow/201_2009/01.%20Czernik%20P..pdf.

[3] Manuel Blum i Silvio Micali, *How to Generate Cryptographically Strong Sequences of Pseudorandom Bits*, SIAM Journal on Computing 13, no. 4 (1984): 850-864.

[4] Ryszard Tanaś, *Kryptografia z elementami kryptografii kwantowej* [online], Poznań, Uniwersytet im. Adama Mickiewicza [dostęp: 15.12.2014], Dostępny w Internecie: http://zon8.physd.amu.edu.pl/~tanas/krypt08.pdf

[5] Mendenhall, Scheaffer, i Wackerly (1986), *Mathematical Statistics with Applications, 3rd Ed.*, Duxbury Press, CA.