METODYKA TESTOWANIA BEZPIECZEŃSTWA GENERATORÓW LICZB PSEUDOLOSOWYCH W SYSTEMACH POMIAROWO-STERUJĄCYCH

Paweł Czernik
Instytut Lotnictwa

Streszczenie

W pracy dokonano oceny generatorów liczb losowych pod kątem ich dopasowania do potrzeb i specyfiki Rozproszonych, Bezprzewodowych Systemów Pomiarowo – Sterujących (RBSPS), coraz powszechniej wykorzystywanych w awionice m. in. na pokładach samolotów bezzałogowych. W pracy rozważane są zasadnicze kwestie dotyczące generatorów liczb losowych: badania „rzeczywistej losowości", „ataki" kryptoanalizy i sposoby zabezpieczenia się przed nimi. W ocenie jakości generatorów wzięto pod uwagę poziom rzeczywistego bezpieczeństwa kryptograficznego algorytmu, efektywność obliczeniową i możliwości implementacji układowej w szybkich i oszczędnych pod względem zapotrzebowania na moc systemach mikroprocesorowych.

1. WPROWADZENIE

Termin Rozproszone, Bezprzewodowe Systemy Pomiarowo - Sterujące (RBSPS) odnosi się do sieci tworzonych w trybie ad hoc przez nieduże, autonomiczne urządzenia nazywane węzłami pomiarowo-sterującymi, które oprócz funkcji komunikacyjnych realizują funkcje pomiarowe, przetwarzania, a często także funkcje wykonawcze na pokładach nowych typów samolotów bezzałogowych, czy śmigłowców. Komunikacja w systemie odbywa się coraz częściej za pomocą fal radiowych (lub innej bezprzewodowej techniki transmisyjnej). Węzły [1,2] poza elementami pomiarowymi, są wyposażone w niewielki mikrokontroler, pamięć, moduł radiowy oraz źródło zasilania. Ponieważ mogą one być zasilane bateryjnie [3], ich zasoby bywają znacznie ograniczone. Bezpieczeństwo przekazywania danych z urządzeń pomiarowych i sterowania, w współczesnej dobie światowego terroryzmu, w większym lub mniejszym stopniu jest istotne we wszystkich rodzajach sieci komunikacyjnych i pokładowych systemach awionicznych. Ale biorąc pod uwagę, jak dużym ograniczeniom podlegają systemy RBSPS staje się oczywiste, że zapewnienie go w tego typu sieciach stanowi szczególnie trudne wyzwanie.

Ograniczenia węzłów są konsekwencją wymagań co do ich rozmiarów, kosztów produkcji, lokalizacji, a przede wszystkim tego, aby stanowiły one autonomiczne terminale. Zasilanie bateryjne węzłów wiąże się z bardzo małymi zasobami energii. Od właściwego wykorzystania i gospodarowania energią zależy czas działania sieci, gdyż wymiana baterii często jest trudna lub w ogóle niemożliwa. Zarządzanie energią ma więc zasadnicze znaczenie. Z tego względu w większości rozwiązań rozproszone węzły charakteryzują się bardzo krótkim cyklem pracy.

Terminale bezprzewodowe większą część czasu pozostają w stanie uśpienia zużywając o kilka rzędów mniej energii niż w czasie aktywności, kiedy są wykonywane pomiary oraz transmisja.

W tego typu systemach występują również ograniczenia kluczowych parametrów węzłów jak: pamięć, moc obliczeniowa, przepływność łącza oraz zasięg przesyłu. Krótki czas życia i mała wartość pojedynczej informacji mogłyby sugerować, że w systemach RBSPS nie ma potrzeby stosowania silnych zabezpieczeń.

Jednak temat bezpieczeństwa w tych sieciach staje się coraz bardziej aktualny. Rosnąca liczba ataków szybko uświadomiła konieczność wprowadzania bardziej rygorystycznej polityki bezpieczeństwa. Do najbardziej spektakularnych można zaliczyć m. in. atak na system zabezpieczeń elektrowni Atomowej w Davise-Besse w USA w 2003 r, w tym samym roku został zaatakowany system sygnalizacji i sterowania kolei w USA, a rok później w Australii [5]. Zostało tez przeprowadzonych wiele mniejszych, skutecznych ataków. Jest więc oczywiste, że również sieci RBSPS w wielu przypadkach wymagają zachowania wysokiego poziomu bezpieczeństwa. Jednak ze względu na ich właściwości jest to szczególnie trudne. Atakujący może próbować zdobyć dostęp do informacji, do których nie jest uprawniony lub próbować zaburzyć, a nawet całkiem zablokować, komunikację w sieci.

Bezpieczeństwo dzisiejszych systemów kryptograficznych jest ściśle związane z generacją pewnych nieprzewidywalnych wartości. Przykładem tutaj może być generacja klucza do szyfru „z kluczem jednorazowym" (ang. one-time pad), klucza do algorytmu DES, AES lub jakiegokolwiek innego szyfru symetrycznego, liczb pierwszych do algorytmu RSA i innych losowych wartości używanych w szyfrach asymetrycznych oraz przy tworzeniu podpisów cyfrowych, czy uwierzytelnieniu typu challenge-response.

2. POJĘCIA PODSTAWOWE

Generator liczb losowych (ang. random number generator) [6] jest to program komputerowy lub układ elektroniczny generujący liczby losowe. Ze względu na sposób generowania liczb losowych można wyróżnić dwa rodzaje stosowanych generatorów: sprzętowe (TRNG) działające na zasadzie ciągłego pomiaru procesu stochastycznego i programowe (PRNG). Ich opis został zawarty w artykule [7].

3. OGÓLNOŚWIATOWE ZAINTERESOWANIE TEMATYKĄ GENERATORÓW LICZB

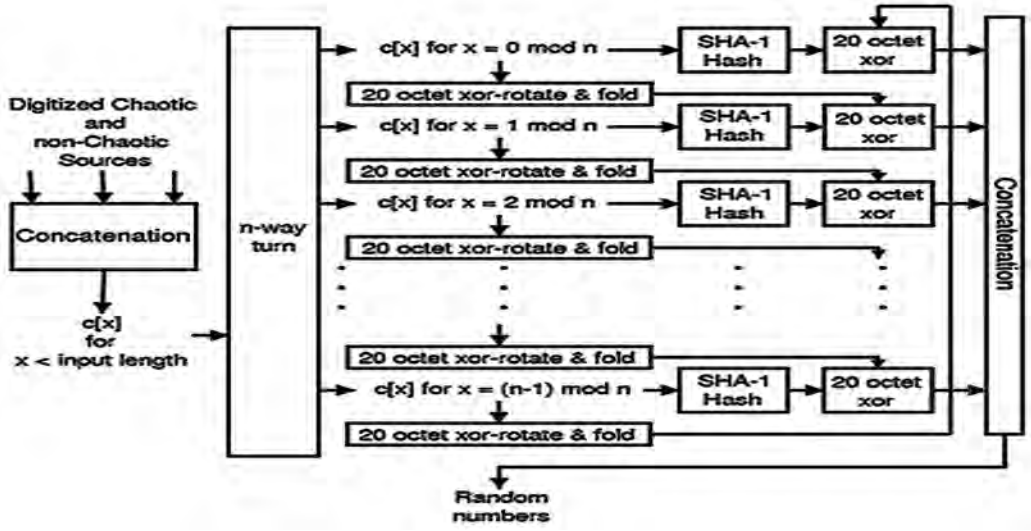
Projektowanie generatorów liczb losowych do celów kryptograficznych jest dzisiaj dynamicznie rozwijającą się dziedziną badań, nad którą pracują rzesze naukowców z całego świta, świadczą o tym coraz to wydajniejsze i bezpieczniejsze rozwiązania, za jakim stoją takie marki jak Intel, czy Microsoft.

Przykładowo w artykule [8] firma Intel przedstawia możliwość szybkiej generacji liczb pseudolosowych z wykorzystaniem technologii Streaming SIMD Extensions 2 (SSE2). SSE2 to rozszerzony zestaw instrukcji SIMD (ang. Single Instruction, Multiple Data) wydanie drugie, dla systemów, w których przetwarzanych jest wiele strumieni danych w oparciu o pojedynczy strumień rozkazów (wektor). Technologia ta została wprowadzona w 2000r. na procesorach rodziny Pentium 4 oraz Athlon 64. Technologa ta podlega dynamicznemu rozwojowi. Popularne na dzień dzisiejszy procesory Intel Core 2 na rdzeniu Penryn (45nm), czy Intel Core i7 mają wbudowane SSE4 [9]. Dzięki technologii SSE2 Intel prezentuje możliwość 5.48-krotnie szybszej generacji liczb pseudolosowych w stosunku do popularnej funkcji rand() standardowej biblioteki języka C, omówionej w naszym artykule [7]. Dzięki wprowadzonym w SSE2 128-bitowym rejestrom XMM i działaniom wektorowym wyłania się możliwość generacji kluczy 32, 64, 96 lub 128 bitowych, jak również wydłużenie okresu generatora z 232 -1 do 234 -1.

Niestety tego typu generatora nie możemy zastosować w systemach kryptograficznych, ale może być on pomocny przy prostych badaniach procesów stochastycznych. Natomiast bardzo oczekiwane jest wydanie przez Intel Advanced Vector Extensions, planowane na rok 2010, które ma zapewnić sprzętowe wsparcie dla algorytmu AES, który będzie można wykorzystać jako kryptograficznie bezpieczny generator liczb pseudolosowych.

Kolejnym bardzo ciekawym pomysłem produkcji liczb pseudolosowych, świadczącym o szybkim rozwoju badań w tej dziedzinie, jest kryptograficznie bezpieczny programowy generator LavaRnd [10]. Jest to linuksowa aplikacja, która zbiera entropię pochodzącą z pomiaru szumu z urządzeń CCD (ang. charge-coupled device), najczęściej z niedrogich kamer internetowych, po czym dokonuje obróbki cyfrowej w oparciu o tzw. algorytm „Digital Blender", z czego jest tworzony jest wyjściowy strumień klucza.

Podstawą „Digital Blender" (rys. 1 – pochodzi z [10]) jest funkcja skrótu SHA-1, co sprawia że można ten generator uznać za złożony przykład algorytmu produkcji liczb pseudolosowych bazujących na kryptograficznej funkcji skrótu, omówionej w artykule [7].



Rys. 1. Algorytm LavaRnd Digital Blender

Generator ten ma niezwykle istotną właściwość jest kryptograficznie bezpieczny. Testy przeprowadzone przez „producenta" [10] w oparciu o pakiet NIST SP800-22 wykazały, że jest lepszy od generatora Blum-Blum-Shuba (przez wielu uważany za najlepszy obecnie generator).

4. LOSOWOŚĆ GENERATORÓW

Jak już wspomniano, na dzień dzisiejszy mamy do czynienia z szeroką gamą generatorów liczb losowych i pseudolosowych, ale niestety nie wiele z nich nadaje się do zastosowań w systemach kryptograficznych, a jeszcze mniej w kryptograficznie bezpiecznych sieciach RBSPS małej mocy. Więc w jaki sposób klasyfikować i „mierzyć" przydatności tych generatorów? Odpowiedź na to pytanie daje dość pasjonująca dziedzina matematyki, jaką jest probabilistyka. Umożliwia ona dogłębną analizę przydatności do określonych zastosowań rozmaitych typów obecnych, jak i nowo projektowanych generatorów liczb losowych i pseudolosowych. Tak więc spróbujmy dość pobieżne zrozumieć jej ideę, a więc zaczynamy!

Oceniając generatory zazwyczaj używamy pojęcia losowości, mniej lub bardziej intuicyjnie. Co jednak należy rozumieć przez losowość w odniesieniu do generatorów pseudolosowych? Wszystkie z przedstawionych w artykule [7] PRNG są po prostu pewnego rodzaju algorytmami deterministycznymi, tzn. znając parametry generatora i warunki początkowe jesteśmy w stanie obliczyć każdy wyraz produkowany przez taki generator. Ogólnie uważa się iż losowość generatora, czyli jego jakość, oceniamy po tym jak produkowana przez niego sekwencja jest różna od prawdziwie losowej sekwencji. Jak w ogóle zmierzyć jakość PRNG - losowość? Właśnie do tego używamy testów statystycznych. Nie udzielają one dokładnej, deterministycznej odpowiedzi, ale pozwalają z pewnym prawdopodobieństwem stwierdzić, czy dana sekwencja „wygląda" na losową. Pokazują, czy sekwencja wyprodukowana przez dany generator posiada pewne własności statystyczne, charakterystyczne dla sekwencji prawdziwie losowej lub czy nie występują w niej pewne defekty.

Należy podkreślić iż nie ma jednoznacznej metody pozwalającej określić, czy generator jest „dobry", czy „zły", raczej stosuje się założenie, że badany PRNG „wygląda na losowy" dopóki nie znajdziemy testu, którego produkowana sekwencja nie przejdzie, dopóty taką sekwencję uznajemy za losową, a generator za „dobry".

Pierwszym, historycznym podejściem do sformułowania metod oceny PRNG były postulaty Golomba, przytaczane tu za [11]:

- w cyklu sN (sekwencja okresowa) liczba jedynek różni się od liczby zer o co najwyżej jeden
- w cyklu sN przynajmniej połowa podsekwencji złożona z tych samych bitów ma długość 1, ¼ długość 2, 1/8 długość 3, itd. aż przekroczymy długość sekwencji; wśród tych podsekwecji powinno prawie tyle samo złożonych z zer i z jedynek

Funkcja autokorelacji przyjmuje tylko dwie wartości:

$$1 \le t \le N-1$$

$$N * C(t) = \sum_{i=0}^{N-1} \left(2_{S_i} - 1\right) * \left(2s_{i+t} - 1\right) = \text{N, jeśli t=0}$$

K, jeśli , gdzie *K* jest pewną liczbą całkowitą.

Sekwencję spełniającą postulaty Golomba nazywamy sekwencją pn (ang, pseudo-noise seąuence). Postulaty Gołomba są przykładem warunków jaki musi spełnić sekwencja, aby została uznana za losową. Te warunki spełniają sekwencje wyjściowe LFSR (omówione w [7]) o maksymalnej długości.

5. TESTY GENERATORÓW

Testy statystyczne można podzielić na: teoretyczne i empiryczne.

5.1. Testy teoretyczne

Przykładem testu teoretycznego jest test spektralny, badający strukturę kratową PRNG. Te testy są projektowane dla konkretnego typu generatora, w oparciu o number-theoretic methods. Na podstawie metody generowania sekwencji staramy się coś o niej powiedzieć.

Ponieważ jest to dosyć skomplikowane takich testów jest mało i znajdują zastosowanie dla niewielkiej grupy generatorów. Opis niektórych można znaleźć w [12].

Dla przypadku generatorów LCG (omówionych w [7]), test spektralny bada korelację generowanych liczb losowych. Dokładnie testuje rozmieszczenie zachodzących na siebie, kolejnych s - tek: $x_n = (x_n, x_{n+1}, ..., x_{n+s-1}), n \ge 0$.

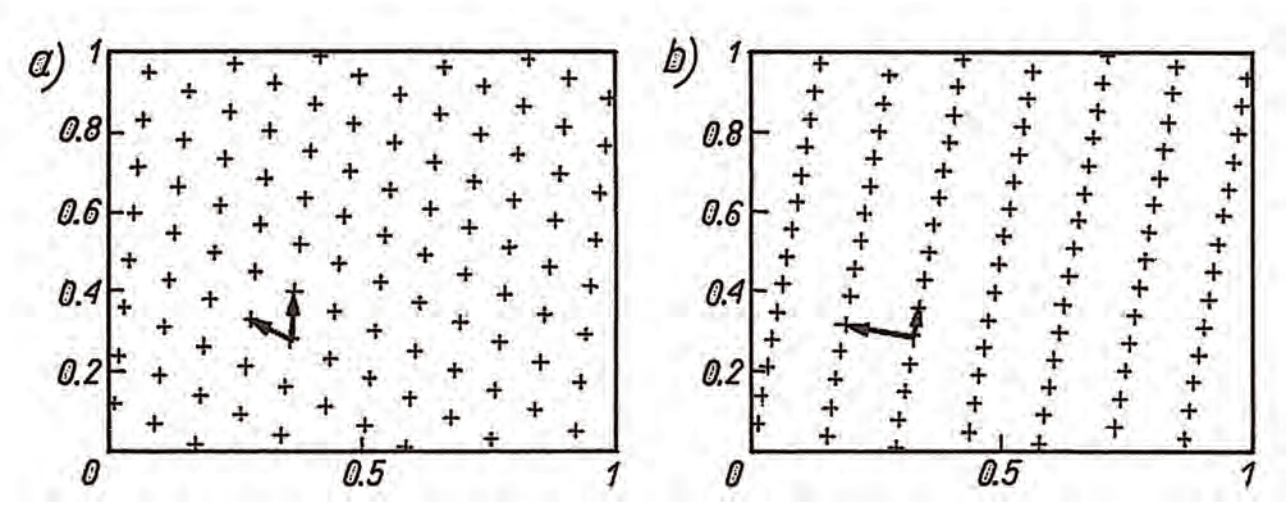
Takie s - tki umieszczane są następnie w s-wymiarowej przestrzeni, gdzie badana jest losowość ich rozkładu. Całość możemy znormalizować do hiperkostki [0, 1)s. Analizując rozłożenie punktów, test bada zależności między elementarni generowanej sekwencji w wielu wymiarach.

W [12] możemy znaleźć dokładne omówienie testu oraz algorytm obliczania wartości γs, nazwaną przez Knutha „s-wymiarową precyzją" (ang. accurancy). Autor podaje również warunki jakie powinna spełniać otrzymana wartość, aby można było uznać, że generator przeszedł test w danym wymiarze. Generalnie γs powinna być niezależna od s (ilości wymiarów), ewentualnie może się zmniejszać. W [12] podano, że jeśli generator przechodzi test dla zakresu: 2 < s < 6, to możemy uznać go za dobry. Inne metody to obliczanie współczynnika Beyer'a (ang. Beyer quotient), czy mierzenie odległości między hiperpłaszczyznami, gdzie się przyjmuje, że test spektralny ma swoją naturalną interpretację geometryczną.

Dla dwóch i trzech wymiarów możemy bez problemu zilustrować rozmieszczenie s-tek danego generatora i „gołym okiem" stwierdzić, że punkty układają się w proste (2D) i płaszczyzny. Jak widać punkty na poniższych rysunkach (zaczerpniętych z [13]) tworzą wyraźną i regularną strukturę, zwaną strukturą kratową - lattice structure. Jest to ich unikalna cecha i bardzo poważny defekt, zwłaszcza z punktu widzenia kryptografii.

Daje to możliwość „przewidywania" kolejnych elementów, na podstawie znajomości poprzednich, bez znajomości parametrów generatora.

W [14] można znaleźć listę generatorów o dość dobrej strukturze kratowej. Nadal jednak jest ona regularna.



Rys. 2. Dwuwymiarowa struktura kratowa (Xi/m, Xi+1/m), i = 0,1,2,... generatora multiplikatywnego dla: a) m = 101, a = 12; b) m = 101, a = 7 (rysunek zaczerpnięty z [13])

5.2. Testy empiryczne

Testy empiryczne są przeprowadzane dla sekwencji wyprodukowanej przez badany generator. Dla danej sekwencji bitów lub liczb obliczana jest konkretna statystyka, którą zazwyczaj porównujemy z konkretnym rozkładem. Testowanie PRNG to nic innego jak testowanie odpowiednich hipotez statystycznych o wyprodukowanej sekwencji (przez dany PRNG). Celem każdego testu statystycznego jest sprawdzenie, czy badana sekwencja wykazuje cechy prawdziwej sekwencji losowej lub sprawdzenie, czy nie ma pewnych defektów statystycznych. Stwierdzenie, że dana sekwencja jest losowa, jest badaną hipotezą oznaczaną Hq (ang. null hypothesis). Test statystyczny, z definicji to funkcja, rzeczywista T operująca na skończonej liczbie wartości wyjściowych z generatora, która mówi jak duże jest prawdopodobieństwo, że założona hipoteza jest prawdziwa. Aby przeprowadzić określony test trzeba znać dystrybuantę F zmiennej losowej T (lub chociaż jej dobre przybliżenie) dla H0.

Znając tę dystrybuantę (lub rozkład zmiennej losowej) możemy zinterpretować otrzymaną wartość (wynik testu). Najważniejszym pojęciem przy testowaniu hipotez statystycznych to poziom istotności, oznaczany α. Poziom istotności α hipotezy statystycznej H0 to prawdopodobieństwo odrzucenia hipotezy H0, jeśli jest prawdziwa. Inaczej: jeśli dla a = 0.05 dany ciąg nie spełnia testu, to istnieje 5% prawdopodobieństwo, że ten ciąg jest jednak losowy. W praktyce używa się poziomów istotności z przedziału (0.1,0.001). Najczęściej spotykana wartość to 0.01. Przy testowaniu hipotez rozróżniamy dwa rodzaje błędów: I rodzaju - uznajemy hipotezę H0 za fałszywą, kiedy ona jest prawdziwa, II rodzaju - uznajemy hipotezę H0 za prawdziwą, w przypadku kiedy jest ona fałszywa.

W naszym przypadku błąd I rodzaju popełniamy, gdy na podstawie wyniku testu odrzucamy ciąg, który jest losowy, błąd II rodzaju, gdy dopuszczamy jako ciąg losowy sekwencję, która ma pewne defekty statystyczne. Oba rodzaje błędów nie są związane ze sobą. Prawdopodobieństwo błędu II rodzaju oznaczamy β, zaś I rodzaju to α (poziom istotności). Błąd I rodzaju uważany jest za ważniejszy i dlatego zazwyczaj tylko podanie jego prawdopodobieństwa (α) jest wymagane przy testach. Często α nazywane jest wielkością testu (ang. size), zaś β mocą testu (ang. power). Zazwyczaj wymaga się, aby dla danego testu β było pomijalnie małe.

Z punktu widzenia kryptografii wydaje się, że błąd II rodzaju jest równie ważny jak błąd pierwszego rodzaju. Dopuszczenie nielosowej sekwencji, z jakimiś defektami, jako np. klucza szyfrującego wydaje się być „gorsze" niż odrzucenie dobrego ciągu. Dlatego ważną cechą testu jest, aby jego wielkość przekraczała moc. Test o takich własościach nazywa się unbiassed. Zauważmy, że im mniejszy poziom istotności, tym bardziej prawdopodobne, że zaakceptujemy sekwencję nielosową jako dobrą.
Kolejną bardzo istotną kwestią jest prawidłowa interpretacja wartość statystyki testowej, jeśli zmienna losowa ma np. rozkład χ2 - test jednostronny lub normalny - test dwustronny.

5.3. Test jednostronny

Test jednostronny jest używamy, gdy zmienna losowa powinna mieć rozkład x^2 z ν stopniami swobody [15]. Rozkład x^2 stosuje się do porównywania dopasowania zaobserwowanych częstotliwości występowania zdarzeń do ich oczekiwanych częstotliwości zgodnych z danym, hipotetycznym rozkładem. Zmienna losową o rozkładzie x^2 z ν stopniami swobody otrzymujemy po zsumowaniu kwadratów v niezależnych zmiennych losowych o rozkładzie standardowym. Na potrzeby tego testu zakłada się, iż w takim wypadku zmienna będzie przyjmować większe wartości dla sekwencji nielosowych. Aby wiec osiągnąć poziom istotności α, znajdujemy w stablicowanych wartościach rozkładu x^2 odpowiadającą mu wartość $x\alpha$, taką że $P(X > x\alpha) = P(X < -x\alpha) = \alpha/2$. Jeśli wartość statystyki testowej XS spełnia nierówność $XS > x\alpha$ wtedy sekwencja nie spełnia testu. W przeciwnym wypadku ciąg spełnia test.

5.4. Test dwustronny

Testu dwustronnego używa się, gdy zmienna losowa powinna mieć rozkład normalny N(0,l) [15]. Rozkład normalny otrzymuje się, poprzez zsumowanie dużej liczby niezależnych zmiennych losowych o takiej samej wartości oczekiwanej i wariancji. Na potrzeby tego testu zakłada się, że dla ciągów nielosowych zmienna może przyjmować zarówno większe, jak i dużo mniejsze wartości. Aby osiągnąć poziom istotności α, znajdujemy w stablicowanym rozkładzie standardowym, taką wartość $x\alpha$, że $P(X > x\alpha) = P(X < -x\alpha) = \alpha/2$. Jeśli wartość statystyki testowej XS badanej sekwencji spełnia jedną z nierówności: $X > x\alpha$ lub $X < -x\alpha$, to ciąg nie przechodzi testu. W przeciwnym wypadku ciąg przechodzi test.

Na dzień dzisiejszy większość gotowych pakietów testów nie zmusza użytkownika do samodzielnego znajdowania odpowiednich wartości odpowiednich rozkładów. Występuje tam za to pojęcie p — wartości, czyli wyniku testu, będącego liczbą z przedziału (0,1). P - wartość, dla danej wartości statystyki testowej i testu T, wyraża się wzorem: $p = P[T > t|H0] = 1 – F(t)$, gdzie H0 oznacza hipotezę podstawową. Jeśli dystrybuanta zmiennej losowej F jest ciągła, to p ma rozkład równomierny w przedziale (0,1). Jak więc w takich warunkach zinterpretować wynik testu postaci p - wartości? W tym przypadku uznaje się, że jeśli uzyskana p - wartość jest zbyt bliska 0 lub 1, to sekwencja nie przeszła testu. Oczywiście pojęcie „zbyt bliska" jest umowne i zależy od uznania użytkownika, który może zdecydować się na mniej lub bardziej ostre kryterium oceny testu. Można zastosować dokładniejszą metodę oceny wyników takiego testu - analogiczną do opisanego poniżej testu dwustopniowego (który jest bardziej ogólny). Generujemy pewną ilość sekwencji i odpowiadających, im p - wartości, a następnie stosujemy np. test Kolomogorowa - Smirnova, wskazujący, czy dana sekwencja ma rozkład równomierny.

W książce [13] została zaproponowana nieco inna metoda testowania, mająca zastosowanie głównie do badania generatorów liczb losowych, nie pojedynczych ciągów bitów. Należy wygenerować n ciągów losowych (różnych), dla każdego obliczamy statystykę testową T i jej dystrybuantę F(T) w badanym hipotetycznym rozkładzie. Jeśli hipoteza jest prawdziwa to ciąg dystrybuant powinien być ciągiem niezależnych zmiennych losowych o rozkładzie standardowym N(0,1). Wynik testowania generatora zależy od prawdziwości powyższej hipotezy.

6. NAJCZĘŚCIEJ WYKORZYSTYWANE ZESTAWY STATYSTYCZNYCH DLA PRNG

Poniższe przykłady testów zostały zaczerpnięte z publikacji NIST: FIPS 140 - 2 [16], z pakietu DieHARD [17] i SP800-22 [18]. Są to standardowe testy badające losowość ciągu bitów, zalecane w przypadku „modułów" kryptograficznych (do badania generatorów losowych i szyfrów).

•**Test Kołmogorowa-Smirnowa** - test zgodności Kołmogorowa służy do weryfikacji hipotezy, że dana zmienna losowa ma rozkład o dystrybuancie *F*. Statystyka testu jest oparta na różnicy między hipotetyczną dystrybuantą *F* a dystrybuantą empiryczną F_n. Statystyka testowa wygląda następująco:

$$D_n = \sup_{-\infty<x<\infty} \left|F_n(x) - F(x)\right|,$$

gdzie dystrybuantę empiryczną definiujemy wzorem:

$$F_n(x) = n^{-1}\sum_{j=1}^{n} 1_{(-\infty,x)}\left(X_j\right)$$

Jeżeli próba użyta do obliczenia dystrybuanty empirycznej pochodzi z rozkładu o dystrybuancie *F*, to $D_n \to 0$ z prawdopodobieństwem 1.

•**The monobit test** - w ciągu zliczamy wszystkie wystąpienia bitów o wartości 1 (ozn. to jako *X*). Gdy: 9654 < *X* < 10346 to badany ciąg przeszedł test.

•**The poker test** - zliczamy ilość wystąpień wszystkich rodzajów 4 - bitowych słów (0000, 0001, ..., 1111). Umieszczamy je w tablicy f[i], i = 0, 1, ..., 15. Następnie obliczamy statystykę:

$$X = \tfrac{16}{5000}\sum_{i=0}^{15} \left[f(i)\right]^2 - 5000$$

Jeśli *X*: 1.03 < *X* < 57.4 to badany ciąg spełnia test.

•**The runs tes**t - w tym teście zlicza się ilość ciągów samych zer lub samych jedynek. Ciągi o długości ≥6 są zliczane wspólnie. Jeśli zliczone ilości (oddzielnie dla zer i dla jedynek) spełniają poniższe nierówności:

Tablica 1. Dopuszczalne ilości sekwencji o różnych długościach

Dł. ciągu	większe niż	mniejsze niż
1	2267	2733
2	1079	1421
3	502	748
4	223	402
5	90	223
6	90	223

to badany ciąg przeszedł test.

•**The long run test** - ten test sprawdza, czy w badanym ciągu występuje sekwencja zer lub jedynek, dłuższa niż 34 bity. Jeżeli nie ma takiej sekwencji, to ciąg przeszedł test.

•**Test spektralny Feldmana** - sprawdza, czy widmo badanego ciągu odpowiada widmu ciągu losowego. Został opisany w artykule „A New Speciral Test for Nonrandomness and the DES" [19]. Obliczana statystyka testowa zależy od dwóch parametrów: badanego ciągu wejściowego i parametru zwanego momentem, oznaczanego jako $r : r \geq 2, r \in Z$. Długość badanego ciągu musi być potęgą dwójki. Na początku bity badanego ciągu przedstawiamy jako ciąg 1 i -1 i obliczamy współczynniki FFT (szybkiej transformaty Fouriera). Te współczynniki będziemy oznaczać odpowiednio am i bm, gdzie $1 \leq m \leq (n/2) - 1$.

Następnie obliczamy estymatę I_m:

$I_m = \frac{1}{n}\left[a_m^2 + b_m^2\right], 1 \le m \le (n/2)-1$ i jej r-tą potęgę (dla każdego m): $I_m^r = \left[I_m\right]^2$ gdzie *r* oznacza r-ty testowy moment.

Następnie obliczamy właściwą statystykę Θ:

$$\Theta r = \frac{1}{v_r}\sum_{k=1}^{n/2-1}\left[I_k^r - m_r\right], \text{gdzie } m_r = r\left[1 - \frac{r(r-1)}{2n}\right],\ v_r = \left((n/2)-1\right)\left[m_{2r} - \left(1+r^2\right)m_r^2\right],$$

m_r i v_r są odpowiednio wartością oczekiwaną i wariancją pewnej statystyki Wr, opisanej w artykule Feldmana [19]. Obliczona statystyka Θr ma rozkład standardowy N(0,1). Decyzja o wyniku testu zostaje podjęta na podstawie testu jednostronnego, dla zadanego poziomu istotności. Poza opisem nowego testu spektralnego artykuł Feldmana zawiera wyniki przeprowadzonych testów dla szyfru DES. Otrzymane wyniki pozwalają stwierdzić, iż DES już po dwóch rundach może pełnić rolę dobrego generatora liczb losowych.

•**Birthday spacings test (BS)-** Przez m oznaczmy liczbę urodzin w n-dniowym roku, a j liczbę wartości urodzin występującą więcej niż raz.

Taka zmienna losowa powinna mieć rozkład Poissona z $\frac{m^3}{4n}$ - sprawdzamy testem x^2.

•**Overlapping 5-permutation test (O5P)**- w sekwencji sprawdza się ciąg miliona 32-bitowych liczb: każda piątka liczb oznacza jeden stan (może być ich 5! = 120). Zliczone wystąpienia każdego stanu porównywane są testem x^2 z rozkładem normalnym.

•***Binary rank test (BR1, BR2 i BR3)*** - formujemy macierz 6 x 8 / 31 x 31 / 32 x 32 bitów z pierwszych 6 / 31 / 32 liczb. Zliczamy rzędy uzyskanych macierzy i przeprowadźmy test zgodności z odpowiednim rozkładem.

•**Testy „słów":**

-bitstream test (BSM) - bity 0 i 1 traktujemy jako dwie litery alfabetu. W ciągu o długości 221 bitów zliczamy te 20 - literowe, zachodzące na siebie słowa, które nie wystąpiły w ciągu. Zmienna losowa powinna mieć rozkład normalny z odpowiednimi parametrami.

-OPSO: overlapping - pairs - sparse – test (OPSO)- jak powyżej, z tym że słowa są dwuliterowe, a alfabet ma 1024 litery.

-OQSO: overlapping - ąuadruples - sparse – test (OQSO)- jak wcześniej, tylko słowa są 4 - literowe, a alfabet ma 32 litery.

- DNA **(DNA)** - analogicznie do powyższych: A, T, G, C - dwubitowe słowa (wybrane z sekwencji), słowa mają 10 liter.

•**Count-the-1's test (CT1 – dla ciągów bajtów, CT2 – dla wybranych bajtów)**– ilość jedynek w bajcie definiuje literę: 0-2: A, 3: B, 4: C, 5: D, 6 - 8: E (wystąpienie każdej ma inne prawdopodobieństwo). Zliczamy wystąpienia wszystkich 5-literowych, zachodzących na siebie słów i porównujemy z odpowiednim rozkładem.

•**This is a parking lot test (TPL)**- w kwadracie o boku 100 losowo parkujemy samochód (kółko o promieniu 1). Jeśli okaże się, że na miejscu, w którym chcemy zaparkować (losowym), to następuje kolizja i próbujemy od nowa, jeśli nie to sukces. Wykres prób parkowania (zakończonych sukcesem lub kolizją) do liczby zaparkowanych aut powinien wyglądać jak dla idealnego generatora losowego.

•**Testy najmniejszej odległości:**

- *The minimum distance test* **(TMD)** - losujemy 1000 punktów w kwadracie o boku 10000 i obliczamy najmniejszą odległość między poszczególnymi punktami. Zmienna losowa powinna mieć rozkład wykładniczy

- *The 3Dspheres test* **(3D)** - analogicznie do powyższego: losujemy 4000 pkt w kostce o krawędzi 1000 pkt. W każdym punkcie umieszczamy kulę o promieniu umożliwiającym

„dotknięcie" najbliższego pkt. Zmienna losowa - objętość najmniejszych kul, powinna mieć rozkład wykładniczy.

•**Testy „podłogujące" liczby losowe na przedział [0,1):**

- *The sqeeze test* **(SQ)** - zliczamy liczbę iteracji potrzebną do zredukowania liczby k = 231 do 1 poprzez k=ceiling(k*U), gdzie U jest liczbą losową, z przedziału [0,1). Porównujemy wynik z odpowiednim rozkładem.

- *The overlapping sums test* **(TOS)**- liczby losowe przekształcamy na ciąg liczb U(l), U(2),..., następnie generujemy sumy: S(1) = U(1) + U(2) + .,., 5(2) = U(2) + U(3) + ..., itd. Pewna liniowa transformacja S powinna przekształcić je do niezależnych zmiennych losowych o rozkładzie standardowym.

- *The runs test* **(RT)**- zliczamy malejące i rosnące, niezachodzące na siebie, sekwencje w ciągu, różnych długości. Porównujemy ze znanym rozkładem.

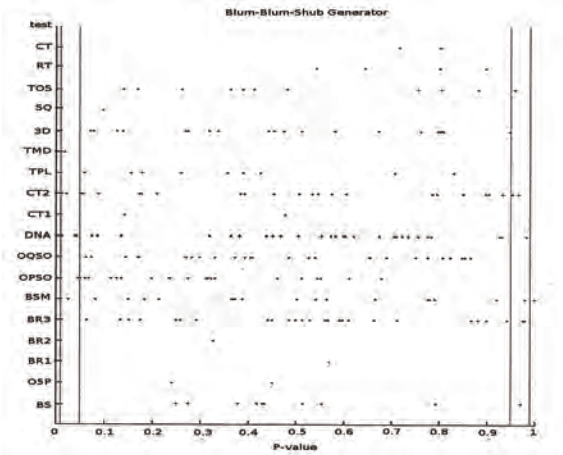
•**The craps test (CT)** - wykonujemy 200000 partii gry craps: każda liczba losowa z ciągu jest traktowana jako wynik rzutu kostką. Zliczamy liczbę zwycięstw i ruchów potrzebnych do zakończenia gry - powinny być zgodne z rozkładem normalnym o pewnych parametrach

7. WYNIKI WYBRANYCH ZESTAWÓW TESTÓW

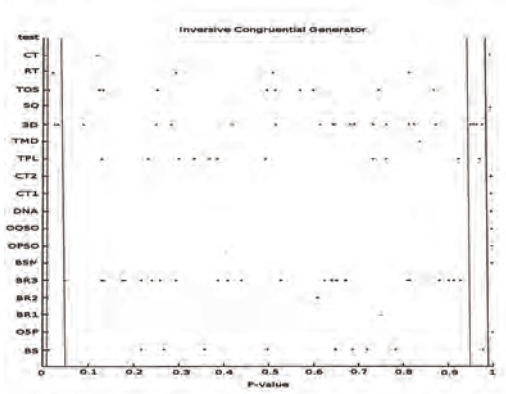
Na dzień dzisiejszy dostępne są dwa główne pakiety testów empirycznych, tj. DIEHARD i SP800-22. Na potrzeby tego artykułu został wybrany pakiet DIEHARD, dostępny pod adresem: http://stat.fsu.edu/pub/diehard/ ze względu na większą ilość oferowanych testów. Jako wynik wykonanych badań środowisko DIEHARD zwraca nam plik z wynikami wybranych testów.

Wynikiem testu jest zazwyczaj p-wartość lub szereg takich p-wartości. Dla niektórych testów otrzymuje się cząstkowe p-wartości np. jeśli test był wykonywany na paru sekwencjach i dla każdej z nich wyliczono pewną p-wartość, to końcową wartość dla testu otrzymano stosując test Kołomogarowa-Smirnova. Dlatego też, na poniższych wykresach dla niektórych testów umieszczony został szereg niebieskich punktów, czyli wyniki testów cząstkowych i punktów czerwonych, oznaczający ogólny wynik testu. Autor pakietu zaleca aby otrzymane p-wartości porównywać z zadanym poziomem istotności tzn.: jeśli otrzymana p-wartość jest mniejsza od α lub większa niż 1 - α, to należy uznać, że ciąg nie przeszedł testu. Do przeprowadzenia wszystkich testów potrzeba ciągu ok. 11000000 bitów.

Na wykresach zostały zamieszczone na osi pionowej konkretne zrealizowane testy (oznaczona skrótami ich nazwy, zgodnie informacjami zawartymi w nawiasach przy omówionych powyżej typach testów), a na poziomej odpowiadające im p-wartości. Pionowe czerwone linie oznaczają granice poziomów istotności α równe 0.01 i 1 - α równe 0.99, a zielone odpowiednio równe 0.05 i 0.95.Jak widać na ilustracji poniżej (Rys. 3), generator BBS przechodzi pomyślnie wszystkie testy.

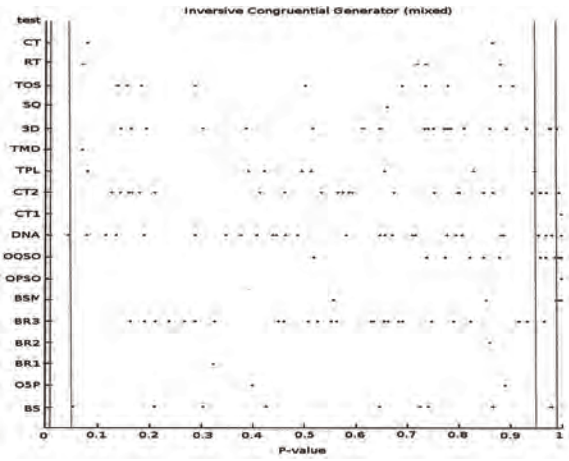


Rys. 3. Wyniki testów dla generatora BBS



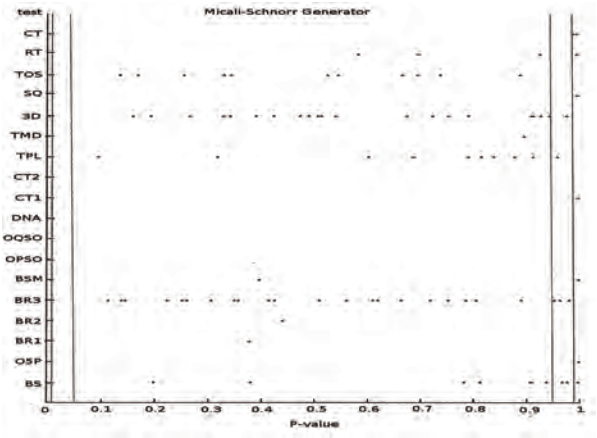
Rys. 4. Wyniki testów dla generatora inwersyjnego

Jak widzimy, wyniki testów dla generatora inwersyjnego (Rys. 4) nie są dobre. Należy w tym miejscu zauważyć, że badany generator ma dosyć mały okres: rzędu 220 . Taki generator był jeszcze w miarę efektywny obliczeniowo - ciąg bitów rzędu 10000000 generuje się ok. 8 minut na komputerze z procesorem ok. 1,5 GHz. Aby wygenerować sekwencję 11MB potrzeba dużo silniejszej maszyny (np. wieloprocesorowej) lub specjalizowanego układu, który będzie wykonywał obliczenia. Przeprowadzone testy pokazały, że potrzeba ok. kilkunastu godzin na wygenerowanie kilkunastu tysięcy bitów dla generatora o okresie rzędu 231. Jednak taki generator prawdopodobnie wypadłby dużo lepiej w późniejszych testach. Tutaj, z powodu zbyt krótkiego okresu generatora, do badania zamiast ciągu bitów wygenerowano ciąg bajtów. Pośrednio na korzyść tego generatora może świadczyć kolejny wykres, przedstawiający wynik kolejnych testów.



Rys. 5. Wyniki testów dla generatora inwersyjnego mieszanego

Powyższy wykres (Rys. 5) został wykonany dla generatora mieszanego tzn. w tym wypadku badany ciąg tworzyły ciągi z paru różnych generatorów inwersyjnych min. z badanego poprzednio. Wykorzystane ciągi się „przeplatały": bajt pierwszego, bajt drugiego, itd. Jak widać taki ciąg przechodzi prawie wszystkie testy. Można z tego wnioskować, iż generator inwersyjny o odpowiednio długim okresie przeszedłby wszystkie testy pakietu.



Rys. 6. Wyniki testów dla generatora Micali-Schnorra

Jak widać generator ten (Rys. 6) nie przechodzi niektórych testów pakietu, są to te testy dla których p-wartości nie są zaznaczone na wykresie: test OPSO, OQSO, DNA, CT2. Te testy to tzw. „małpie testy", które operują na bardzo dużej sekwencji wejściowej i zliczają ilość wystąpień różnych „słów" ciągów bitów. Fakt, iż generator nie przeszedł tych testów wskazuje, iż pewne długie ciągi lub podciągi bitów powtarzają się, czyli ma pewne defekty statystyczne, niewykrywalne przy krótkich sekwencjach.

W przeprowadzonych testach najlepiej wypadł generator Blum-Blum-Shuba - nie tylko przechodzi wszystkie testy, ale także jest w szybki. W miarę dobrze wypadł także generator inwersyjny, jednak jest on zdecydowanie wolniejszy od pozostałych. Nie zaprezentowano tutaj badań innych generatorów takich jak LCG, MWC, czy Fibonacciego - nie przeszły one większości testów.

Powyższe badania potwierdzają kryptograficzną bezpieczność generatora Blum-Blum-Shuba, który został przedstawiony w artykule [7]. Generator ten jest dobrym narzędziem produkcji liczb pseudolosowych w bardziej zaawansowanych systemach RBSPS, gdzie bezpieczeństwo gra dominującą rolę.

8. ANALIZA BEZPIECZEŃSTWA KRYPTOGRAFICZNEGO WYBRANYCH GENERATORÓW LICZB PSEUDOLOSOWYCH

Generatory używane w kryptografii poza tym, że powinny dobre w sensie statystycznym, muszą być także nieprzewidywalne, tak aby przeciwnik nawet znając poprzednie bity produkowane przez generator nie był w stanie przewidzieć kolejnych. Ta ważna cecha jest nazywana bezpieczeństwem kryptograficznym generatora (cryptographicaly secure). Generatory bezpieczne krytograficznie bezpieczne w literaturze określane są skrótem CSPRNG [7].

Generator Blum-Blum-Shub jest jednym z CSPRBG, można go zdefiniować następująco [7]:

$$x_{i+1} = x_i^2 \bmod N$$

gdzie N jest iloczynem dwóch liczb pierwszych

Generator ten ma następujące własności:

- Znajomość N i $x0$ pozwala na wygenerowanie sekwencji $x0$, $x1$, ... w przód, dla dowolnej wartości początkowej $x0$.
- Aby efektywnie wygenerować „wstecz" sekwencję niezbędna jest znajomość czynników N.
- Co więcej, czynniki N są niezbędne aby „zgadywać" z większym prawdopodobieństwem parzystość x-1 (elementów poprzedzających), mając N i losowe $x0$.

W pracy [20] można znaleźć dowody podanych wyżej własności. Ich podstawą są założenia o problemie logarytmu dyskretnego i reszty kwadratowej, które mówią, że te dwa problemy są nierozwiązywalne w czasie wielomianowym (przy doborze odpowiednio dużych liczb).

Te założenia i wynikające z nich własności generatora BBS pozwoliły na udowodnienie, że żaden algorytm w czasie wielomianowym nie może przewidzieć kolejnego bitu sekwencji na podstawie poprzednich.

-Generatory inwersyjne – nad bezpieczeństwem tego typu generatorów nie prowadzono dotychczas żadnych prac. Są one głównie badane pod kątem przydatności w zastosowaniu w symulacjach. W kryptografii może pełnić funkcją nieliniowej funkcji filtrującej np. w NFSR. Samodzielnie raczej nie powinien być stosowany, gdyż jego bezpieczeństwo jest niezbadane.

-Generatory oparte na LFSR - są równie podatne na różnego typu ataki jak generatory typu LCG. Dalej zostanie omówione bezpieczeństwo dwóch typów generatorów: shrinking generator i altrenating step generator:

•*shrinking generator* - ten generator został opisany artykule [7]. Załóżmy, że LFSR składowe generatora mają długość L_1 i L_2. Jeśli wielomiany obu LFSR-ów są znane, zaś warunki początkowe nie, to najlepszy atak prowadzący do odkrycia warunków początkowych ma $0\left(2^{L_1} * L_2^3\right)$ kroków. Jeśli jednak wielomiany charakterystyczne obu rejestrów są tajne i uzależnione od zmiennych najlepszy atak potrzebuje $0\left(2^{2L_1} * L_1 * L_2\right)$ kroków.

Istnieje również atak wykorzystujący złożoność liniową, który wymaga $0\left(2^{L_1} * L_2^2\right)$ kroków. Nie jest wtedy istotne czy wielomian charakterystyczny jest jawny, czy tajny,
ale ten atak wymaga $2^{L_1} * L_2$ kolejnych bitów sekwencji wyjściowej, co oznaczę, że jest dosyć trudny do przeprowadzenia dla dużych LFSR-ów. Aby shrinking generator był bezpieczny należy tak dobrać LFSR-y aby, ich długości były względnie pierwsze, a wielomiany charakterystyczne tajne. Jeśli założymy, że $L_1 \approx l$ i $L_2 \approx l$, to poziom bezpieczeństwa takiego generatora wynosi około 22*l*. Dobór odpowiednio dużych rejestrów gwarantuje odpowiednie bezpieczeństwo.

•*alternating step generator* - aby zapewnić maksymalne bezpieczeństwo tego generatora składowe rejestry powinny być maksymalnej długości, zaś ich długości być parami względnie pierwsze. Dodatkowo długości tych generatorów powinny być zbliżone do siebie, gdyż najlepszy znany atak to atak typu „dziel i rządź" wymaga około 2*l* kroków, gdzie *l* to długość rejestru. Odpowiednio duża długość wszystkich rejestrów zapewnia odporność na ten atak.

9. ATAKI NA PRNG

Ataki na generatory pseudolosowe nie są szeroko rozpowszechnione w literaturze. Na początku analizy tego problemu należy zwrócić szczególną uwagę na to, aby przeciwnik nie był w stanie przewidzieć sekwencji generatora w żadną stronę: ani do przodu ani wstecz. Wynika z tego, że nie tylko generator dla przeciwnika musi być czarną skrzynką: jego parametry nie mogą być znane, ale przede wszystkim jego wartość początkowa musi być tajna. Ataki na generatory pseudolosowe możemy podzielić na parę grup (poniższa klasyfikacja pochodzi z [21]).:

- *bezpośredni atak kryptoanalityczny* - kiedy atakujący jest w stanie bezpośrednio odróżnić sekwencje produkowaną przez generator od prawdziwie losowej.
- *atak na podstawie znajomości wartości początkowych (ziarna)* - atakujący może przewidzieć i manipulować danymi wejściowymi generatora, wykorzystując tą wiedzę do kryptoanalizy generatora, tego typu ataki możemy jeszcze podzielić na ataki ze znanymi danymi wejściowymi (ang. known input attack), wybranymi danymi wejściowymi (ang. chosen input attack), powtórzonymi danymi wejściowymi (ang. replayed input attack).
- *state compromise extension attkcs* - tego typu ataki wykorzystują uprzednią znajomość stanu generatora do przewidywania sekwencji wyjściowej generatora, zarówno wprzód jak i wstecz, taki atak ma największe szansę powodzenia, gdy generator „startuje" z łatwego do zgadnięcia dla przeciwnika stanu. Możemy wyróżnić pewne podklasy tego ataku:
- *backtracking attack*-atakujący używa zdobytych wiadomości o stanie generatora, aby odnaleźć poprzednie sekwencje wyjściowe generatora

- permament compromise attacks-jeśli atakujący odkrył stan generatora w chwili *t* to wszystkie przeszłe i przyszłe stany generatora są podatne na atak
- *atak typu iteracyjne zgadywanie*-atakujący używa wiedzy o stanie generatora w chwili *t* i sekwencji wyjściowych do przewidzenia stanu w chwili *t* + *e*, zakładamy, że dane wejściowe nie są znane atakującemu, ale mogą zostać zgadnięte
- atak typu spotkanie w środku-atakujący używa wiedzy o stanie generatora w chwilach *t* i *t* + 2*e* do przewidzenia stanu generatora w chwili *t* + *e*.

10. ZABEZPIECZENIA PRZED ATAKAMI

Można zalecić parę sposobów zabezpieczania PRNG przed opisanymi powyżej atakami:

- Jeśli sekwencja wyjściowa generatora jest podatna na ataki, należy używać funkcji skrótu lub innego silnego losowego przekształcenia - to nie do końca gwarantuje bezpieczeństwo PRNG, ale może je poprawić.
- Warunki początkowe generatora powinny być uzyskiwane z użyciem funkcji skrótu i znacznika czasowego lub licznika,
- od czasu do czasu generator powinien być ponownie inicjalizowany.
- Warunki początkowe i ziarna powinny być trzymane w tajemnicy i szczególnie chronione oraz wybierane ze szczególną uwagę, gdyż stanowią one podstawę bezpieczeństwa generatora.
- Także przy projektowaniu nowych generatorów należy sprawdzić ich odporność na wymienione ataki, zwłaszcza te bazujące na wybranych danych początkowych.

11. GENERATORY LICZB W PRAKTYCE

W tym artykule przedstawione zostały sposoby badania rzeczywistej „losowości" generatorów liczb i na tej podstawie dokonano weryfikacji przydatności generatorów do zastosowań w kryptografii, jak również różne sposoby ataków na nie i możliwości zabezpieczenia się przed nimi.

Pewnie wielu szanownych Czytelników zastanawia się - „po co to wszystko", gdzie to znajduje jakiegokolwiek zastosowanie, czy to może jest po prostu czysta, zawiła matematycznie teoria niczemu nie służąca? Kto by miał ochotę, czy potrzebę dokonywać jakichś „ataków" na generatory liczb losowych w systemach RBSPS?

Otóż okazuje się, że często powodzenie udanego „ataku" kryptoanalizy na generator liczb pseudolosowych może stanowić najprostszą drogę do przejęcia kontroli w strategicznym systemie RBSPS, działającym na pokładzie samolotu bezzałogowego, a w konsekwencji przejąć całkowicie kontrolę nad nim. Powodzenie takiego ataku może doprowadzić w najczarniejszym scenariuszu do śmierci nawet tysięcy ludzi.

12. PODSUMOWANIE

Bezpieczeństwo przekazywania danych z urządzeń pomiarowych i sterowania, jest niezwykle istotne we wszystkich rodzajach sieci komunikacyjnych, a zwłaszcza w pokładowych systemach awionicznych. Zapewnienie kryptograficznego bezpieczeństwa komunikacji, wymusza zaprojektowanie dostatecznie bezpiecznego prymitywu, jakim są generatory liczb losowych.

W literaturze spotykamy się z różnymi możliwościami praktycznej realizacji takich systemów „produkcji" liczb wyglądających na losowe, ale czy takie one w rzeczywistości są? To właśnie należy sprawdzić przy pomocy pakietów testów „losowości", jakie zostały zaimplementowane np. w pakiecie DIEHARD czy SP800-22.

Gdy zakładamy, iż dany generator ma pracować w systemach RBSPS, należy się jeszcze zastanowić czy spełnia specyfikę tych systemów, tzn. niską moc obliczeniową, małą zajętość pamięci operacyjnej i mały pobór mocy zasilania. Przy tak ostro sprecyzowanych wymaganiach najlepszym algorytmem produkcji liczb pseudolosowych wydaje się być generator oparty na szyfrach strumieniowych, bazujący na algorytmie SNOW 2.0 czy alternating step.

Dla przypadku postawienia na pierwszym miejscu potrzeby bezwzględnego bezpieczeństwa kryptograficznego, należałoby zastosować generator Blum-a, Blum-a i Shub-a (BBS).

W niektórych zastosowaniach systemów RBSPS, węzły mogą być wyposażone w urządzenia typu CCD (ang. charge-coupled device), co umożliwia skorzystanie z generatora LavaRND (bezpieczniejszego kryptograficznie od BBS-a), który należy zaimplementować zgodnie z specyfiką określonego węzła.

Jak zostało wspomniane w tym artykule, bardzo ważną kwestią jest pełna losowość zarodka generatora. W warunkach RBSPS dla przypadku najprostszego rozwiązania można skorzystać z aktualnej wartości czasu systemowego węzła zaszyfrowanej za pomocą funkcji skrótu np. SHA-1. W systemach bardziej rozbudowanych, gdzie do bezpieczeństwa przywiązujemy wysoką wagę, zaleca się wykorzystanie entropii pochodzącej z niestałości częstotliwości własnej oscylatora węzła RBSPS.

LITERATURA

[1] **W. Nawrocki:** Rozproszone systemy pomiarowe, WKŁ, Warszawa 2006

[2] **I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci:** A Survey on Sensor Networks, IEEE Communication Magazine, 2002

[3] **B. Holt, L. Doherty, E. Brewer:** Flexible Power Scheduling for Sensors Networks, Information Processing in Sensor Networks (IPSN), Berkeley, CA, 2004

[4] **I. Dunajewski, Z. Kotulski:** Optimal Wireless Sensors Location For Widespread Structures Monitoring. 36th Solid Mechanics Conferen-ce, Gdańsk 2008

[5] **E. Micha:** Tendencje rozwojowe w obszarze systemów pomiarowo-sterujących, Systemy Pomiarowe w Badaniach Naukowych i w Przemyśle, Łagowo, 2006

[6] **R. Zieliński:** Generatory liczb losowych, WNT, Warszawa 1972

[7] **P. Czernik, J. Olszyna:** Cryptographic random number generators for low-power distributed measurement system, Proc. SPIE 2009;

[8] **K. Owens, R. Parikh:** Fast Random Number Generator on the Intel Pentium 4 Processor, 9.09.2008

[9] **Original 45nm Intel Core Microarchitecture:**, http://www.intel.com/technology/itj/2008/v12i3/3-paper/6-examples.html

[10] **Projekt LavaRnd, http:**//www.lavarnd.org/

[11] **A. Menezes, P. van Oorschot, S. Yanstone:** Handbook oj Applied Cryptography, WNP, 2005

[12] **D. Knuth:** Sztuka programowania, WNT, 2002

[13] **R. Wieczorkowski, R. Zieliński:** Komputerowe generatory liczb losowych, WNT 1997

[14] **P. L'Ecuyer:** Tables of Linear Congruential Generators od Different Sizes and Good Lattice Structure, Mathematics of Computation 68, 1999

[15] **M. Sobczyk:** Statystyka, WNP, 2008

[16] **FIPS 140 - 2 ,** http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[17] **DieHARD**, http://www.stat.fsu.edu/pub/diehard/

[18] **SP800-22**, http://www.nist.gov/

[19] **F. A. Feldman:** A New Spectral Test for Nonrandomness and the DES, IEEE Transactions on Software Engineeńng, vol. 16, 1990

[20] **L. Blum, M. Blum, M. Shub:** Comparison of two pseudorandom generators, Crypto'82. Advances in Cryptology 1981 - 1997, Sprmger, 1997

[21] **J. Kelsey, B. Schneier, D. Wagner, C. Hali:** Cryptoanalytic Attacks on Pseudorandom Number Generators, http://www.schneier.com/paper-prngs.html

Pawel Czernik

TESTING METHODOLOGY OF SECURE RANDOM NUMBER GENERATORS IN THE MEASUREMENT AND CONTROL SYSTEMS

Summary

This paper focuses on the study reviewed evaluation and random number generators to determine whether they fit demands and characteristics Low Power Distributed Measurement and Control Systems (LPDMCS), becoming more widely used in avionics, among others on boards Unmanned Aerial Vehicles (UAV). In this work are considered basic categories of random number generators: study the "real randomness", cryptanalysis attacks and ways to protect against them. The evaluation quality of generators took into account the level of security algorithms, computational efficiency and the possibility of systemic implementation of rapid and efficient in terms of demand on the power of the latest microprocessor systems.