1. Marek LEŚNIEWICZ, 2. Piotr KOMOROWSKI

Wojskowy Instytut Łączności – Państwowy Instytut Badawczy
ORCID: 1. 0000-0001-5505-1268; 2. 0000-0002-0752-0055

doi:10.15199/48.2021.08.24

Metoda i układ powielania ciągów losowych

Streszczenie. *Przedmiotem pracy są metoda i układ powielania ciągów losowych. Powielanie polega na pseudolosowym próbkowaniu odpowiednio licznej próby ciągu prawdziwie losowego ze źródła w postaci generatora sprzętowego, a następnie na wymianie wykorzystanej próby na następną i skasowaniu próby zużytej. W wyniku powielania otrzymuje się ciągi wynikowe o zadowalających właściwościach probabilistycznych i parametrach statystycznych. Metoda będąca przedmiotem pracy pozwala na powielanie ciągów z generatorów sprzętowych do krotności sięgających 1000.*

Abstract*. The subject of the work is the method and system for duplicating random sequences. Reproduction consists in pseudo-random sampling of a sufficiently large sample of a truly random sequence from a source in the form of a hardware generator, and then on the exchange of the used sample for the next one and deletion of the used sample. As a result of duplication, result sequences with satisfactory probabilistic properties and statistical parameters are obtained. The method allows duplication of sequences from hardware generators up to 1000 times.*
*(**Method and system for duplicating random sequences.**)*

Słowa kluczowe: generacja ciągów (liczb) losowych
Keywords: random sequences (number) generation

Wstęp

Przedmiotem pracy są metoda i układ powielania ciągów losowych. Ciągi te mają liczne zastosowania w wielu dziedzinach nauki i techniki, ze wskazaniem na kryptografię, statystykę, obliczenia numeryczne, symulacje stochastyczne, cyfrowe przetwarzanie sygnałów, technikę algorytmów randomizowanych i wiele innych.

W zastosowaniach tych używa się obecnie wyłącznie ciągów prawdziwie losowych, pochodzących ze źródeł w postaci generatorów sprzętowych TRNG (ang. *True Random Number Generator*), których typowe rozwiązania umożliwiają wytwarzanie ciągów o przepływnościach rzędu 10 Mbit/s, tymczasem współczesne potrzeby sięgają nawet 1 Gbit/s na jedno urządzenie, czy aplikację.

Zwiększanie przepływności ciągów z takich generatorów jest możliwe, ale trudne technicznie i bardzo kosztowne.

Jeśli jednak wytwarzane przez nie ciągi charakteryzują się odpowiednimi, udowadnianymi właściwościami probabilistycznymi i dobrymi, potwierdzonymi pomiarami parametrami statystycznymi (niezależność i warunkowe prawdopodobieństwo występowania „0” i „1” równe $P(0|X_1, \ldots, X_N) = P(0) = 1/2$ i $P(1|X_1, \ldots, X_N) = P(1) = 1/2$), to takie ciągi można powielać. W wyniku powielania otrzymuje się ciągi o równie zadowalających właściwościach probabilistycznych i praktycznie identycznych parametrach statystycznych. Metoda będąca przedmiotem pracy pozwala na skuteczne powielanie ciągów z generatorów sprzętowych do krotności sięgających 1000. Metoda może mieć zastosowanie w dowolnych, wskazanych powyżej zastosowaniach, w których potrzebne są ciągi losowe o bardzo dużych przepływnościach, sięgających 1 Gbit/s.

Opis stanu techniki

Obecnie na rynku dostępnych jest wiele sprzętowych generatorów ciągów losowych [1], a ich przepływności wyjściowe zawierają się w przedziale od kilku kbit/s do kilku Gbit/s. Zdecydowana większość z nich jako źródło losowości wykorzystuje element szumiący, a pochodzący z niego sygnał szumowy jest następnie komparowany do postaci binarnej i przetwarzany w losowy ciąg binarny. Jednak taki źródłowy ciąg z natury rzeczy nigdy nie ma pożądanych właściwości i parametrów statystycznych i musi zostać poddawany operacji poprawiającej (ang. *post-processing*). Jako funkcję realizującą tę operację stosuje się zwykle kryptograficzną funkcję skrótu (ang. *hash*) wobec jednego ciągu, albo funkcję XOR (ang.*exclusiveOR*) wobec kilku ciągów. Obszerne omówienie powyższych zagadnień zawierają liczne publikacje [2], [3], [4], [5], [6], [7], [8].

O ile zastosowanie elementu szumiącego jako źródła losowości jest powszechnie uznane, o tyle operacja poprawiająca bywa przedmiotem kontrowersji i wiele ze stosowanych metod jest kwestionowanych, zwłaszcza w przypadku osiągania ekstremalnych przepływności. Zasadniczą wątpliwość budzi pytanie, w jakim stopniu losowość ciągu wyjściowego jest pochodną losowości źródła, a w jakim operacji poprawiającej. Można bowiem powiedzieć, że łatwo jest osiągnąć pożądane właściwości i parametry statystyczne ciągu dzięki odpowiedniemu użyciu np. funkcji skrótu, ale wtedy te właściwości i parametry są pochodną użytej funkcji, a nie źródła losowości i możemy tutaj mówić tylko o generatorze pseudolosowym.

W związku z powyższym optymalna metoda powielania musi opierać się za założeniu, że źródłowy ciąg jest w pełni losowy, tzn. pochodzący z generatora sprzętowego, a wynik powielenia próby ciągu opierać się na mechanizmie całkowicie niezależnym od zawartości tej próby, a jednocześnie być jej wierną kopią w sensie właściwości probabilistycznych i parametrów statystycznych.

Nie są znane publikacje, czy patenty, opisujące proponowaną metodę. Wynika to z dwóch powodów. Pierwszego – patenty dotyczące sprzętowej generacji ciągów losowych opisują zasadniczo klasyczny model generacji, opisany w pierwszym akapicie tego punktu, a liczba patentów tego typu nie jest duża i sięga 200 (hasła: *true random number generator*, *hardware random number generation*). Drugiego – metoda powielania wymaga użycia zaawansowanych metod matematycznych i nowoczesnych technologii elektronicznych.

Pokrewne związki z przedmiotem ma technika obliczeń stochastycznych (ang. *stochastic processing*, *stochastic computing*). Ma ona relatywnie ubogą literaturę – najpełniejszy przegląd w artykule [9] – ponieważ sama technika nie znalazła zastosowania w komercyjnej technice komputerowej. Znane są natomiast nieliczne prace dotyczące symulacji procesów losowych, w których zastosowano tę technikę, implementując ją głównie w szybkich układach programowalnych [10], [11].

Znane są również nieliczne patenty, np. US 5412587 (A) (*Pseudorandom Stochastic Data Processing*), czy US US6745219 (B1) *Arithmetic Unit Using Stochastic Data Processing*), opisujące realizacje podstawowych operacji logicznych na ciągach pseudolosowych. Ponadto znanych jest wiele patentów z dziedziny cyfrowego przetwarzania

obrazów oraz symulacji sieci neuronowych, w których zastosowano technikę obliczeń stochastycznych. Nie mają one jednak związku z przedmiotem niniejszej pracy. Inną pokrewną dziedziną jest modelowanie matematyczne zwane metodą *Monte Carlo*, polegające na losowym próbkowaniu realizacji złożonego procesu lub jego symulacji. Pod tym względem metoda ta przypomina przedmiot pracy, ale jej intencje są inne – aby jak najmniejszą liczebnością próby dokładnie określić właściwości i parametry badanego procesu. Podobne intencje ma metoda *bootstrap*, stosowana w statystyce do modelowania nieznanych rozkładów prawdopodobieństw, a polegająca na wielokrotnym losowaniu próbek ze zwracaniem. Intencje przedstawionych metod są więc dalekie od przedmiotu pracy, ale wszystkie potwierdzają, że losowe próbkowanie może wiernie odtwarzać właściwości probabilistyczne i parametry statystyczne dowolnych procesów, w tym procesów losowych.

Opis metody

Źródłem pomysłu na metodę jest spostrzeżenie, że o ile przepływności wyjściowe sprzętowych generatorów ciągów losowych są ograniczone, o tyle generowane przez nie ciągi mają właściwości i parametry bliskie doskonałości, wyrażające się m.in. niezależnością i warunkowym prawdopodobieństwem występowania „0” i „1” równym $P(0|X_1, \ldots, X_N) = P(0) = 1/2$ i $P(1|X_1, \ldots, X_N) = P(1) = 1/2$). Skoro tak, to odpowiednio wybierając z próby ciągu prawdziwie losowego kolejne, niezależne od siebie elementy, można kreować nowy ciąg losowy, mający równie zadowalające właściwości probabilistyczne i praktycznie identyczne parametry statystyczne.

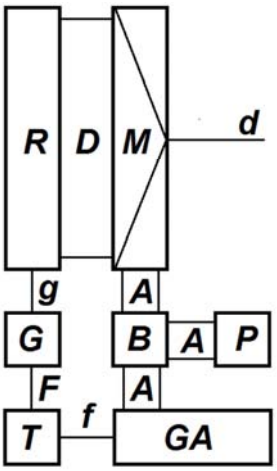
Istotą realizacji metody jest zatem powielanie ciągów losowych, polegające na pseudolosowym próbkowaniu odpowiednio licznej próby ciągu prawdziwie losowego z generatora sprzętowego, a następnie na wymianie wykorzystanej próby na następną i skasowaniu próby zużytej. Ponieważ mówimy o powielaniu, to z natury rzeczy liczebność ciągów powielonych będzie krotnością liczebności ciągów próbkowanych, a więc przepływność ciągu powielonego będzie krotnością przepływności ciągu próbkowanego.

Wynika stąd, że optymalną metodą powielania jest próbkowanie zmieniającej się próby ciągu prawdziwie losowego. Oczywiście w każdym przypadku metoda, czyli pseudolosowy mechanizm próbkowania nie może mieć żadnego związku z zawartością ciągu próbkowanego, co spełni założenie, że losowość ciągu powielonego będzie pochodną losowości ciągu z generatora sprzętowego, a nie metody próbkowania.

Metoda i układ opierają się na następujących założeniach, zilustrowanych na schemacie blokowym przedstawionym na Rysunku 1.

Układ składa się z

1. sprzętowego generatora ciągów prawdziwie losowych ***G*** o przepływności wyjściowej BR_G,
2. *n*-bitowego rejestru przesuwnego ***R*** z wejściem szeregowym,
3. multipleksera ***M*** z *n*-bitowym, równoległym wejściem danych, $m = [\lg_2 n]$-bitowym, równoległym wejściem adresowym ***A*** i 1-bitowym, szeregowym wyjściem danych ***d***,
4. generatora adresów ***GA*** o przepływności wyjściowej BR_P w postaci np. liniowego rejestru *L*-tego stopnia ze sprzężeniem zwrotnym LFSR *(Linear Feedback Shift Register*) opisanego wielomianem pierwotnym,
5. bufora adresów ***B*** i pamięci ostatniego adresu ***P***,
6. generatora taktu zegarowego ***T*** dla generatora adresów ***GA*** o częstotliwości *f*, równej liczbowo przepływności wyjściowej BR_P i taktu dla generatora ciągu losowego ***G*** o częstotliwości *F*, równej liczbowo przepływności wyjściowej BR_G.



Rys. 1. Schemat blokowy układu w którym zastosowano metodę

Połączenia w układzie są następujące:

1. wyjście sprzętowego generatora ciągów prawdziwie losowych ***G*** połączone jest z szeregowym wejściem rejestru przesuwnego ***R***, gdzie nadaje ciąg ***g***,
2. wszystkie *n* wyjść rejestru przesuwnego ***R*** połączonych jest równolegle ze wszystkimi *n* wejściami danych multipleksera ***M*** (wektor danych ***D***), a kolejność tych połączeń może być dowolna,
3. *m* wybranych wyjść z generatora adresów ***GA*** podanych jest poprzez bufor adresów ***B*** na *m* wejść adresowych multipleksera ***M*** (wektor adresów ***A***),
4. pamięć ostatniego adresu ***P*** połączona jest z buforem adresów ***B***,
5. wyjście generatora ***T*** z taktem *f* połączone jest z wejściem zegarowym generatora adresów ***GA***, a z taktem *F* z wejściem zegarowym generatora ciągu losowego ***G***,
6. aby zapewnić systematyczne próbkowanie, przepływność wyjściowa BR_P generatora adresów ***GA*** powinna być 2^k razy większa od przepływności wyjściowej BR_G generatora ciągów losowych ***G***, co pozwala opisać współczynnik powielania jako $K = BR_P / BR_B = f / F = 2^k$, gdzie *k* – liczba naturalna.

Zasada działania układu jest następująca:

1. ciąg ***g*** ze sprzętowego generatora ciągów losowych ***G*** o przepływności wyjściowej BR_G jest ładowany z taktem o częstotliwości *F* do rejestru przesuwnego ***R***,
2. po załadowaniu rejestru ***R*** rozpoczyna się pseudolosowe wybieranie bitów z jego wyjść przez multiplekser ***M*** z częstotliwością *f*, a wybrana wartość jest podawana na jego wyjście *d*, jako element powielonego ciągu ***d***,
3. po wykonaniu *K* próbkowań z wyjścia generatora ***G*** podawany jest następny bit, który wchodzi na pierwszą pozycję w rejestrze ***R***, wszystkie pozostałe bity są przesuwane o jedną pozycję w przód, a bit znajdujący się na ostatniej pozycji jest nieodwracalnie kasowany,
4. po wykonaniu kolejnych *K* próbkowań operacja wprowadzania nowego bitu i przesuwania pozostałych odbywa się tak samo,
5. ponieważ statystyka wektorów ***A*** z generatora adresów musi być równomierna, to prawdopodobieństwo powtórzenia się kolejnego adresu o długości *m* wynosi $P = 2^{-m} = 1/n$, a więc jest równe odwrotności długości rejestru ***R***; w takim przypadku zostanie wybrane

powtórnie to samo wejście i ta sama wartość; ponieważ dana wartość nie jest niezależna od samej siebie, tylko taka sama, to wprowadza korelację, w wyniku której P(0,0) = P(1,1) > P(0,1) = P(1,0), dokładnie P(0,0) = P(1,1) = 1/4 + 1/n i P(0,1) = P(1,0) = 1/4 – 1/n; aby temu zapobiec, bieżący adres jest zapamiętywany w pamięci ostatniego adresu ***P***, a każdy kolejny adres jest porównywany z poprzednikiem; w przypadku, kiedy zdarzy się zgodność, wybrana wartość jest odrzucana.

Do prawidłowego działania układu konieczny jest właściwy wybór parametrów n i m, z czego wynika sprzętowa konfiguracja układu. Z rachunku prawdopodobieństwa wiadomo, że każda próba ciągu losowego o liczebności n, nawet ciągu opisanego prawdopodobieństwami P(0) = P(1) = 1/2, charakteryzuje się chwilową nierównowagą „0" lub „1", daną zależnością $b = (2\pi\, n)^{-1}$, a więc tym mniejszą, im próba jest liczniejsza. Wynika stąd, że dla zapewnienia minimalnej nierównowagi w ciągu powielonym, należy wybrać jak najdłuższy rejestr ***R***. W przypadku implementacji na układach dyskretnych długość ta nie może być zbyt duża, ale w układach programowalnych można implementować rejestry o długościach rzędu kilku kilobitów. Można również wykorzystać do tego celu nie klasyczne rejestry, budowane na przerzutnikach, a specjalizowane bloki pamięci z wejściami szeregowymi i wyjściami równoległymi. Nie zmienia to oczywiście istoty rzeczy, przedstawionej na Rysunku 1. W praktyce wskazane jest, by rejestr ***R*** miał długość co najmniej n = 8192 bitów. Pozwala to nie tylko uzyskiwać dobre statystyki ciągów powielanych, ale również znacząco zmniejszyć prawdopodobieństwo korelacji, zwłaszcza przy powielaniu o dużych krotnościach K. Dla n = 8192 słowo adresowe multipleksera ***M*** będzie równe m = 13. W przypadku, kiedy jako generator adresów ***GA*** zostanie użyty liniowy rejestr L-tego stopnia ze sprzężeniem zwrotnym LFSR, to jego długość L musi być odpowiednio duża z dwóch względów. Pierwszego – musi być znacząco większa od długości m słowa adresowego multipleksera ***M***, aby można było wybrać odczepy odpowiednio odległe od siebie i uniemożliwiające wprowadzanie korelacji wynikających z potencjalnego, wielokrotnego użycia fragmentów tych samych sekwencji w kolejnych słowach adresowych. Drugiego – musi być na tyle duża, żeby okres sekwencji generowanej przez rejestr był znacząco dłuższy od najdłuższej liczebności ciągów planowanych do wygenerowania. Zakładając przepływność BR_P = 1 Gbit/s i minimalny, 10-letni czas generacji, równy $3{,}16 \cdot 10^8$ sekund, mamy liczebność $N = 3{,}16 \cdot 10^{17} \approx 2^{58}$. W zastosowaniach kryptograficznych przyjmuje się profilaktyczne podwajanie parametrów uznanych za bezpieczne, stąd można przyjąć długość L = 127, co daje czas generacji, równy $1{,}7 \cdot 10^{29}$ sekund, czyli $5{,}39 \cdot 10^{21}$ lat. Wartość ta jest optymalna również z tego powodu, że proste zapętlenie XOR odczepów 126 i 127 w rejestrze LFSR stanowi implementację wielomianu pierwotnego $1 + x^{126} + x^{127}$.

Układ badawczy

Badaniom podano ciągi z układu przedstawionego na Rysunku 1, zrealizowanego na strukturze FPGA z interfejsem Ethernet 1000Base-TX, o następujących właściwościach i parametrach:

a) źródłem ciągów prawdziwie losowych był sprzętowy generator ***G*** o przepływności wyjściowej BR_G = 16 Mbit/s,
b) rejestr przesuwny ***R*** miał długość n = 8192 bitów, zatem w każdej próbie powinny pojawić się błędy korelacji o wartości co najmniej $K = 1/n = 1{,}22 \cdot 10^{-4}$,
c) multiplekser ***M*** miał n = 8192 bitowe wejście danych i $m = [\lg_2 n\,]$-bitowe sterujące, tzn. m = 13,
d) jako generatora adresów ***GA*** o przepływności wyjściowej BR_P użyto liniowego rejestru L = 127 stopnia ze sprzężeniem zwrotnym LFSR *(Linear Feedback Shift Register*) opisanego wielomianem pierwotnym $1 + x^{126} + x^{127}$,
e) sprawdzono różne sposoby wykorzystania bitów sterujących z generatora adresów ***GA***, ale nie stwierdzono żadnych zależności w sensie pozycji lepszych, czy gorszych,
f) użyto trzech buforów adresów ***B*** i pamięci ostatniego adresu ***P*** – bez pamięci, z pamięcią ostatniego adresu i z pamięcią 8 kolejnych adresów,
g) generator taktu zegarowego ***T*** dla generatora adresów ***GA*** miał częstotliwość f dobieraną w zależności od krotności powielania $E = BR_P / BR_B = f / F = 2^k$, gdzie k = 3 dla powielania E = 8, k = 7 dla powielania E = 128, k = 10 dla powielania E = 1024.

Wyniki statystycznych badań ciągów losowych wytworzonych metodą powielania

Najprostszą, choć dającą tylko wskaźnikowe wyniki, jest metoda badań statystycznych. Pakiety testów do takich badań zawierają zwykle dziesiątki różnych testów, od najprostszych (statystyki nierównowagi „0" i „1", par elementów „00", „01", „10" i „11" oraz dłuższych sekwencji), do zaawansowanych, konstruowanych z intencją wykrywania określonych nielosowości, najczęściej korelacyjnych. W naszych badaniach posłużyliśmy się zestawem testów opracowanych w WIŁ-PIB i służących wykrywaniu charakterystycznych nielosowości spotykanych w sekwencjach, stanowiących wyniki szyfrowania lub skracania. W testach tych zalecane jest wielokrotne, co najmniej 3 krotne badanie prób o dużych liczebnościach ze wskazaniem na co najmniej 100 MB. Okazuje się, że dopiero takie liczebności prób pozwalają ustabilizować statystyki i przy analizie móc oceniać tylko probabilistyczne charakterystyki ciągu, nieobciążone rozproszeniami wyników charakteryzujących próby o niewielkich liczebnościach.

Wykonano ponad 1 000 doświadczeń, polegających na wygenerowaniu prób ciągów losowych o liczebności 100 MB i zbadaniu ich właściwości i parametrów statystycznych. Przytoczenie nawet cząstkowych wyników tych badań jest praktycznie niemożliwe, jako że raport z badania każdej próby liczy kilkanaście stron tekstu – poza zbiorczymi, syntetycznymi ocenami każdy raport zawiera ponadto obszerne analizy ergodyczne, polegające na długich zestawieniach statystyk próby podzielonej na mniejsze odcinki i porównywaniu ich ze sobą.

Poprzestańmy zatem na syntetycznej konkluzji, że zgodnie z oczekiwaniami, im powielenie było częstsze, tym wyniki były gorsze, ale mechanizm zmniejszający prawdopodobieństwo powtórzenia się kolejnego adresu skutecznie je poprawiał.

Wnioski ze statystycznych badań ciągów losowych wytworzonych metodą powielania

Na podstawie przeprowadzonych badań można powiedzieć, że

a) zasadniczy wynik naukowo-techniczny został osiągnięty – teza, że źródłowy, prawdziwie i doskonale losowy ciąg binarny może zostać skutecznie powielony, a funkcja powielająca nie „popsuje" znacząco jego właściwości i parametrów statystycznych, została doświadczalnie dowiedziona,
b) uzyskiwane ciągi powielone, nawet w układach z mechanizmem zmniejszającym prawdopodobieństwo powtórzenia się dwóch takich samych w ciągu kolejnych ośmiu adresów – w żadnym przypadku nie

są jeszcze ciągami o doskonałych właściwościach i parametrach statystycznych, wymagają zatem jakiejś operacji poprawiającej,

c) wydaje się, że taką operacją może być tylko funkcja XOR wobec kilku powielonych ciągów – założenie to wynika przede wszystkim stąd, że funkcja XOR posiada matematycznie udowodnioną właściwość minimalizacji obu rodzajów błędów losowości – nierównowagi i korelacji międzysymbolowych.

Uwzględniając powyższe wnioski dokonano badań wygenerowanych ciągów, dokonując wcześniej operacji XOR na dwóch i trzech próbach niezależnych od siebie składowych ciągów powielonych, tzn.

$$\{ \text{ciąg X} \} = \{ \text{ciąg A} \} \oplus \{ \text{ciąg B} \}$$

i $$\{ \text{ciąg Y} \} = \{ \text{ciąg A} \} \oplus \{ \text{ciąg B} \} \oplus \{ \text{ciąg C} \} .$$

Jako ciągów A, B i C użyto najlepszych ciągów, tzn. uzyskanych z wykorzystaniem mechanizmu zmniejszającego prawdopodobieństwo powtórzenia się dwóch takich samych w ciągu kolejnych ośmiu adresów.

Okazało się, że niezależnie od stopnia powielania E = 8, 128 i 1024

a) dla ciągu X każdy test daje wynik formalnie pozytywny, ale z zauważalnym, choć minimalnym poziomem nielosowości,

b) dla ciągu Y każdy test daje wynik formalnie pozytywny, jednak zupełnie nieodróżnialny od statystyk ciągu doskonale losowego.

Wniosek może być tylko jeden – metoda powielania może być uznana za skuteczną pod warunkiem zastosowania operacji poprawiającej w postaci operacji XOR na minimum trzech niezależnych od siebie, powielonych ciągach składowych.

Wiąże się to oczywiście z kosztami – musimy użyć aż trzech generatorów ciągów prawdziwie losowych, a każdy z nich zawiera 8 generatorów składowych, co wymaga użycia aż 24 generatorów składowych. Zauważmy jednak, że dla osiągnięcia wynikowej przepływności choćby 1 Gbit/s metodą sumowania, tzn. składania niezależnych ciągów musielibyśmy użyć aż 512 generatorów o źródłowej przepływności 16 Mbit/s, co jest zupełnie nierealne technicznie i ekonomicznie.

Wracając do założeń naszej metody można jeszcze zadać pytanie, czy jako źródła w postaci sprzętowego generatora ciągów losowych ***G*** z Rysunku 1 nie można by użyć słabszego generatora ciągów niedoskonale losowych? Należałoby wtedy założyć, że generowane przez niego ciągi o niedoskonałych statystykach zostaną dodatkowo „popsute" w procesie powielania, ale później poprawione operacją XOR kilku, nawet więcej niż trzech ciągów, co w każdym przypadku byłoby opłacalne, choćby ekonomicznie. Niestety, tak powielone ciągi wykazują znacząco zwiększony i nieidentyfikowany poziom nielosowości względem ciągów uzyskiwanych w już opisany sposób, zatem odrzuciliśmy ten pomysł – niosący nieidentyfikowane zagrożenia bezpieczeństwa ciągów wynikowych.

Analiza formalna

Na zakończenie odnieśmy się do współczesnych analiz bezpieczeństwa ciągów pseudolosowych [8]. Mechanizm generacji tych ciągów jest oczywiście deterministyczny i teoretycznie zawsze odwracalny, w związku z tym nie wszystkie jego pojęcia i scenariusze dają się przełożyć na opis naszej metody i generatora. Niemniej, warto jest odnieść się do takiego podejścia, choćby po to, żeby sprawdzić, czy nasza metoda już w swoich założeniach nie zawiera jakiś słabości, które choćby formalnie mogły posłużyć do wskazania elementarnych błędów, czy zagrożeń bezpieczeństwa generowanych ciągów. W takich analizach rozważa się następujące zagadnienia.

A. Nierozróżnialność statystyczna

Określa się tzw. rozróżniacz (ang. *distinguisher*) i wyrocznię (ang. *oracle*), które mają sens procedur testujących i decyzyjnych. W bardziej zaawansowanych analizach używa się ich w schematach gier. W naszym przypadku użyjemy ich do rozstrzygania prostej alternatywny.

W przypadku dwóch ciągów powinny one odpowiedzieć na pytanie o nierozróżnialność statystyczną ciągów N-wymiarowych zmiennych losowych X_N i Y_N, tzn. niemożności identyfikacji różnych rozkładów prawdopodobieństwa w sensie ich nieodróżnialności obliczeniowej

$$P(X_N) - P(Y_N) \le \varepsilon_P$$

lub jednowartościowym sensie niemożności identyfikacji różnych entropii

$$H(X_N) - P(Y_N) \le \varepsilon_E$$

gdzie ε_P i ε_E są kryteriami identyfikacji. Oczywiście wartości ε_P i ε_E powinny odpowiadać jakimś realnym kryteriom – w naszej analizie użyliśmy pojęcia niezbędnego, minimalnego czasu generacji ciągu T_{MIN}, który stanowi intuicyjnie oczywiste kryterium bezpieczeństwa [2].

B. Szacowanie entropii

Cały dowód bezpieczeństwa naszej metody generacji sprowadza się nie tylko do szacowania, ale jednocześnie do pomiarowego potwierdzania entropii oczekiwanej [1] każdej próby ciągu. Pomiary te są w pełni i każdym przypadku zgodne z szacowaniami z wysoką dokładnością.

C. Krzepkość (ang. *robustness*)

Można by ją też nazwać siłą mechanizmów. Zawiera ona analizę

a) odporności (ang. *resilience*) – pytania o to, czy ciąg „wygląda jak losowy" (ang. *looks like random*) dla obserwatora, który nie zna aktualnego stanu wewnętrznego generatora, ale wie wszystko o źródle entropii, oczywiście w sensie właściwości, ale nie jego aktualnego stanu? Ponieważ nasze źródło entropii w postaci sprzętowego generatora ciągów losowych ***G*** jest prawdziwie i doskonale losowe, to nasz generator jest odporny. Zakłada się też, że obserwator może wpłynąć na źródło entropii lub wewnętrzny stan generatora. W przypadku sprzętowego generatora ciągów losowych ***G*** założenie to jest bezzasadne – nie ma takiej możliwości technicznej (fizyczne zabezpieczenia mechaniczne i elektromagnetyczne), a gdyby nawet, to wewnętrzne funkcje i mechanizmy kontrolno-pomiarowe zidentyfikują taki fakt i wyłączą generator z eksploatacji.

b) zabezpieczenia przed czytaniem wstecz (*forward security*) – pytania o to, czy znana teraźniejszość, czy poznawana przyszłość nie pozwalają na odtworzenie przeszłości? Uniemożliwia to źródło entropii w postaci sprzętowego generatora ciągów losowych ***G***.

c) zabezpieczenia przed czytaniem w przód (*backward security*) – pytania o to, czy znana przeszłość nie pozwala na przewidywanie przyszłości. Uniemożliwia to źródło entropii w postaci sprzętowego generatora ciągów losowych ***G***.

Można jeszcze raz przypomnieć, że stan wewnętrzny sprzętowego generatora ciągów losowych ***G*** i generatora w sensie układu powielającego – zasadniczo rejestru przesuwnego ***R*** – jest stanem chwilowym, bezpamięciowym i niemonitorowanym.

D. Stabilność bezpieczeństwa

Pod tym pojęciem rozumie się odtwarzanie bezpieczeństwa (ang. *recovering security*) i utrzymywanie bezpieczeństwa (ang. *preserving security*). Ponownie można je scedować na źródło entropii w postaci sprzętowego generatora ciągów losowych ***G***. Odtwarzanie bezpieczeństwa można interpretować jako doskonałą losowość ciągów z takiego źródła przy każdej inicjacji (włączeniu), a utrzymywanie jako kontrolę w trakcie eksploatacji. W naszym generatorze gwarantują to wbudowane funkcje i mechanizmy autotestowania losowości produkowanych ciągów, wszczynające alarmy w przypadku ciągu niespełniającego wymagań i wyłączające generator z eksploatacji. Zakładamy domyślnie, że sama metoda powielania ciągów jest algorytmiczna i poprawnie zaimplementowana sprzętowo, a więc ma bezbłędny i stabilny charakter.

Podsumowanie

Przedstawiona metoda i układ pozwalają na generowanie ciągów losowych o dużych przepływnościach, wykorzystując sprzętowy generator ciąg losowych o małej przepływności. Metoda została sprawdzona, zaimplementowana i zastosowana w prototypowych urządzeniach kryptograficznej ochrony informacji, okazując swoją użyteczność oraz pełną zgodność oczekiwanych właściwości i obliczonych parametrów.

Opisana metoda i układ do jej realizacji są wynikiem pracy statutowej nr 815 zrealizowanej w Wojskowym Instytucie Łączności (Generacja i powielanie ciągów losowych metodami sprzętowymi – cz.1: analiza możliwości matematyczno-technicznych, cz. 2: implementacje techniczne i zastosowania praktyczne) oraz są przedmiotem zgłoszenia patentowego w UP RP, nr P.432641, pt. „Sposób powielania ciągów losowych i układ do stosowania tego sposobu”.

***Autorzy**: dr hab. inż. Marek Leśniewicz, profesor WIŁ-PIB, mgr inż. Piotr Komorowski, Zakład Kryptologii WIŁ-PIB, 05-130 Zegrze, ul. Warszawska 22A, E-mail: m.lesniewicz@wil.waw.pl*

LITERATURA

[1] https://en.everybodywiki.com/Comparison_of_hardware_random_number_generators

[2] Leśniewicz M., Sprzętowa generacja losowych ciągów binarnych, *Wydawnictwo Wojskowej Akademii Technicznej*, 2009 r., ISBN 978-83-61486-31-2.

[3] Leśniewicz M., Sprzętowa generacja ciągów losowych z przepływności 100 Mbit/s, *Przegląd Telekomunikacyjny - Wiadomości Telekomunikacyjne*, 2011, nr 11/2011, s. 1608-1613.

[4] Borowski, M., Leśniewicz M., Wicik R., Grzonkowski M., Generation of random keys for cryptographic systems, *Annales UMCS Informatica*, AI XII, Number 3 (2012), pp. 75-87.

[5] Borowski, M., Leśniewicz M., Modern usage of “old” one-time pad, *Communications and Information Systems Conference (MCC)*, 2012.

[6] Leśniewicz M., Expected Entropy as a Measure and Criterion of Randomness of Binary Sequences, *Przegląd Elektrotechniczny*, 2014, nr 1, s. 42-46.

[7] Leśniewicz M., Analyses and Measurements of Hardware Generated Random Binary Sequences Modeled as Markov Chains, *Przegląd Elektrotechniczny*, 2016, nr 11, s. 268-274.

[8] Güneysu T., True random number generation in block memories of reconfigurable devices, *International Conference on Field-Programmable Technology (FPT)*, 2010.

[9] Alaghi A., Hayes J.P.: Survey of Stochastic Computing, ACM Transactions on Embedded Computing Systems, Vol. 12, No. 2s, Article 92, May 2013.

[10] Kawalec P., Układy arytmetyki stochastycznej i ich implementacja w strukturach FPGA, *Pomiary-Automatyka-Kontrola*, vol.55, 8/2000.

[11] Kawalec P., Analiza metod generowania zmiennych wejściowych dla probabilistycznych modeli procesów transportowych, Zeszyty Naukowe Politechniki Śląskiej, Nr 1895, 2013.