# 1 GENERÁTORY NÁHODNÝCH ČÍSEL

## 1.1 ÚVOD

Bezpečnosť mnohých kryptografických systémov je závislá od generovania nepredikovateľných (náhodných alebo pseudonáhodných)[1] čísel alebo postupností čísel. Ako príklad môžeme uviesť jednorázovú šifru, tajný kľúč v algoritme DES, prvočísla $p$ a $q$ v algoritme RSA, digitálnych podpisoch a pod. Vo všetkých týchto prípadoch je potrebné generovať (binárne) postupnosti určitej dĺžky, ktoré sú dostatočne náhodné.

## 1.2 ZÁKLADNÉ PRINCÍPY GENEROVANIA NÁHODNÝCH ČÍSEL

V rámci cvičenia opíšeme základné metódy generovania náhodných postupností, ktoré sa využívajú v technickej praxi, pričom však nie všetky opísané metódy sú vhodné pre kryptografické aplikácie.

### 1.2.1 KONGRUENTNÉ GENERÁTORY

**Lineárny kongruentný generátor** produkuje pseudonáhodnú postupnosť čísel $x_1, x_2, x_3, \ldots$ s využitím lineárnej rekurentnej rovnice

$$x_n = (ax_{n-1} + b) \bmod m, \qquad n \geq 1 \tag{1.1}$$

pričom $a, b, m$ sú **parametre** charakterizujúce generátor a $x_0$ je (tajná) **počiatočná hodnota** (angl. **seed**).

Parameter $m$ je v prípade realizácie zvyčajne volený tak, aby operácia $\bmod m$ bola realizovaná automaticky bez nutnosti realizovať dodatočné matematické operácie. Typickým príkladom sú hodnoty $m = 2^{16}$ resp. $m = 2^{32}$, ktoré sú realizované automaticky pri výpočtoch s presnosťou 16 resp. 32 bitov. Parametre $m, a, b$ je možné zvoliť tak, aby perióda generovanej pseudonáhodnej postupnosti bola $m$. Vhodné hodnoty je často možné nájsť vo forme tabuliek [2].

[1] Náhodné čísla je možné generovať pomocou **špeciálnych technických** prostriedkov ako sú napr. čítače rádioaktívnych častíc, šumové generátory a pod. Tieto techniky sú stále používané a patria medzi **najbezpečnejšie generátory**. Typickým moderným riešením je integrácia šumového generátora priamo do najnovšej čipovej sady INTEL. Samozrejme takéto náhodné čísla **nie sú reprodukovateľné**. Postupnosti generované pomocou **deterministických metód** (posuvných registrov, procesorov, ...) **sú reprodukovateľné** (pokiaľ poznáme počiatočný stav systému, ktorý postupnosť generuje) a nazývajú sa **pseudonáhodné postupnosti**. Keďže tieto pseudonáhodné postupnosti sú generované číslicovými systémami, ktoré majú **konečný počet** stavov, **majú pseudonáhodné postupnosti vždy (aj keď zvyčajne veľmi veľkú) konečnú periódu**!

Typickým príkladom týchto generátorov sú generátory náhodných čísel napr. v knižniciach jazyka C

```
void srand( unsigned int seed );
int rand( void );
```

ktorých praktická implementácia môže vyzerať napr. takto (príklad publikovaný ANSI výborom):

```
unsigned long   next=1;

void srand( unsigned int seed ) {
        next = seed;
}

int rand( void ) {
        next = next*1103515245 + 12345;
        return (unsigned int) (next/65536) % 32768;
}
```

Tieto typy generátorov sú široko využívané v rámci rôznych simulácii a pravdepodobnostých algoritmov. Ich základnou výhodou je veľká rýchlosť, pretože vyžadujú malý počet operácií. Z pohľadu kryptografických algoritmov sú **nevhodné**, pretože z malej časti pseudonáhodnej postupnosti **je možne odvodiť** zvyšok postupnosti aj v prípade, že parametre $a, b, m$ a hodnota $x_0$ **nie sú známe**! Podobne je možné prelomiť aj zložitejšie kvadratické

$$x_n = \left(ax_{n-1}^2 + bx_{n-1} + c\right) \bmod m \tag{1.2}$$

a kubické kongruentné generátory

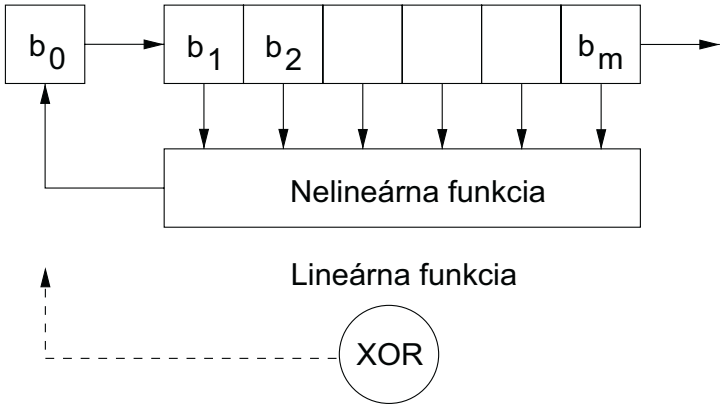
$$x_n = \left(ax_{n-1}^3 + bx_{n-1}^2 + cx_{n-1} + d\right) \bmod m \tag{1.3}$$

### 1.2.2 GENERÁTORY S VYUŽITÍM POSUVNÝCH REGISTROV SO SPÄTNOU VÄZBOU

Generátory na báze posuvných registrov so spätnou väzbou sú v kryptografii populárne predovšetkým vzhľadom na jednoduchú technickú realizáciu a teda možnosť generovať pseudonáhodné postupnosti s veľkou výstupnou rýchlosťou. Tieto generátory využívajú dve základné časti:

- Posuvný register,
- Spätoväzobnú funkciu.

ktorých vzájomné prepojenie je znázornené na obrázku 1 (tento spôsob zapojenia sa tiež nazýva **Fibonacciho zapojenie**).




*Obr.1 Fibonacciho forma realizácie pseudonáhodného generátora*

Najlepšie je prepracovaná[2] teória posuvných registrov **s lineárnou spätnou väzbou** (LSFR – **L**inear **F**eedback **S**hift **R**egister), kde je spätoväzobná funkcia tvorená **XOR operáciou** medzi vybranými bitmi posuvného registra.

LSFR s $m$- bitmi môže byť v jednom z $2^m - 1$ rôznych nenulových stavov[3]. Teoreticky tak $m$-bitový LSFR môže generovať pseudonáhodnú binárnu postupnosť s periódou maximálne $2^m - 1$. Ak má generovaná postupnosť periódu $2^m - 1$, nazýva sa $m$ **-sekvencia** a príslušný LSFR – **LSFR s maximálnou periódou**.

Nutnou a postačujúcou podmienkou aby určitý LSFR bol LSFR s maximálnou periódou je splnenie podmienky aby polynóm vytvorený z váh[4] vstupujúcich do operácie XOR a hodnoty 1 bol tzv. **primitívny polynóm [2]**. Primitívne polynómy je možné nájsť napr. v tabuľkách [2] prípadne vypočítať pomocou programu MATLAB príkazom (je súčasťou komunikačného toolboxu):

```
gfprimfd
```

Napr. aplikovaním príkazu gfprimfd(4,'all') (pre $m = 4$) dostaneme dva polynómy:

1 1 0 0 1 – t.j. polynóm $f_1(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 = 1 + x + x^4$

1 0 0 1 1 – t.j. polynóm $f_2(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 = 1 + x^3 + x^4$

pričom hodnota $b_i = 1$ ($b_i = 0$) znamená, že existuje (neexistuje) prepojenie medzi príslušným registrom a blokom spätnej väzby.

**Príklad**

*Nakreslite štruktúru LSFR ktorý zodpovedá polynómom $f_1(x)$ a $f_2(x)$ a overte, že príslušné LSFR generujú m-sekvencie (t.j. postupnosti s periódou $2^4 - 1 = 15$.*

Počet rôznych (cyklicky odlišných) $m$-sekvencií, ktoré môžu byť generované pomocou LSFR s $m$-bitovým posuvným registrom je [3]

[2] Vzhľadom na **linearitu** sú z kryptografického hľadiska tieto generátory **nevhodné**. Princíp LSFR sa však často využíva v zložitejších **nelineárnych zapojeniach**. Typickým príkladom je prúdová šifra A5 použitá v systéme GSM.

[3] Nulový stav v LSFR bude generovať konštantnú nulovú postupnosť.

[4] Tieto váhy majú hodnotu 0 ak medzi operáciou XOR a príslušným bitom posuvného registra nie je prepojenie resp. hodnotu 1 ak toto prepojenie existuje. Príslušné polynómy sú tak polynómami nad Galoisovym telesom GF(2).

$$N(m) = \frac{2^m - 1}{m} \prod_{i=1}^{I} \frac{p_i - 1}{p_i} \tag{1.4}$$

pričom $I$ prvočísiel $p_i$, $i = 1, 2, \ldots, I$, je možné určiť z kanonického rozkladu čísla

$$2^m - 1 = \prod_{i=1}^{I} p_i^{e_i} \tag{1.5}$$

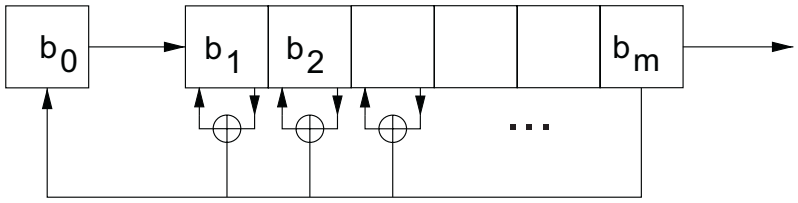
Programová realizácia LSFR je relatívne neefektívna. Nasledujúci programový kód v jazyku C realizuje LSFR s primitívnym polynómom

$$f = x^{32} + x^7 + x^5 + x^3 + x^2 + x + 1 \tag{1.6}$$

a vychádza z Fibonacciho formy LSFR

```
static   unsigned long   ShiftRegister = 1;        /* lubovolna hodnota okrem 0 */

int LSFR( void ) {
        ShiftRegister = (((( ShiftRegister >> 31)
                        ^ (ShiftRegister >> 6)
                        ^ (ShiftRegister >> 4)
                        ^ (ShiftRegister >> 2)
                        ^ (ShiftRegister >> 1)
                        ^ ShiftRegister ))
                        & 0x00000001)
                        << 31)
                        | (ShiftRegister >> 1);
        return   ShiftRegister & 0x00000001;
}
```

Z pohľadu programovej realizácie je vhodnejšia tzv. **Galoisova forma** LSFR, ktorá je znázornená na nasledujúcom obrázku




*Obr.2 Galoisova forma realizácie pseudonáhodného generátora*

Programová realizácia v jazyku C vychádzajúca z Galoisovej formy vyzerá takto

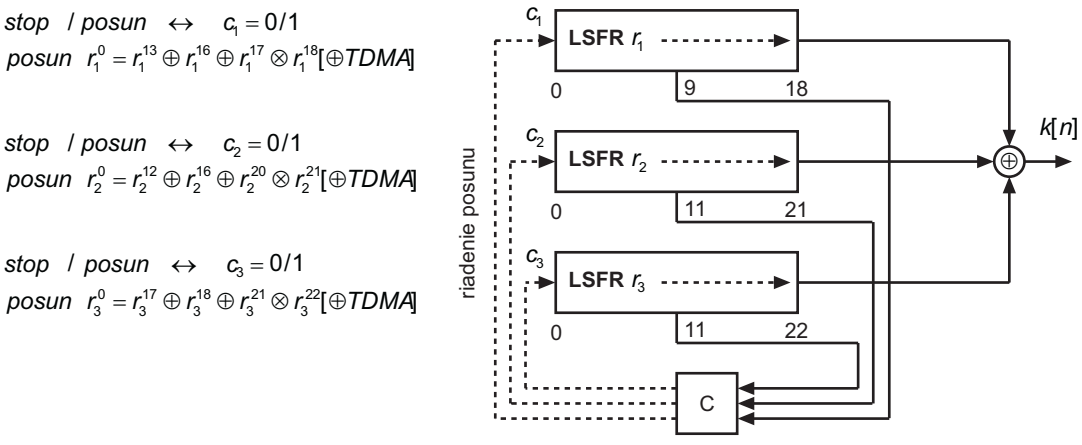
```
#define  MASK            0x80000057;
static   unsigned long   ShiftRegister = 1;    /* lubovolna hodnota okrem 0 */

int modified_LSFR( void ) {
        if ( ShiftRegister & 0x00000001 ) {
                ShiftRegister = ( ShiftRegister ^ MASK >> 1 ) | 0x80000000;
                return 1;
        }
        else {
                ShiftRegister >>=1;
                Return 0;
        }
}
```

a je podstatne efektívnejšia ako predchádzajúca implementácia.

### 1.2.2.1 PRÍKLAD VYUŽITIA LSFR V SYSTÉME GSM

K ochrane rádiovej komunikácie v systéme GSM sa používajú kryptografické techniky, ktorých cieľom je sťaženie odpočúvania cudzích hovorov. Samotné šifrovanie je realizované len medzi mobilnou a bázovou stanicou. Ďalšia komunikácia medzi bázovou stanicou a zvyškom GSM siete je nešifrovaná. Samotný šifrovací algoritmus A5 (resp. jeho varianty A5/1 a A5/2) je stále oficiálne utajovaný, na základe dohody o mlčanlivosti je poskytovaný výrobcom čipov a mobilných telefónov. Nepodpísanie (spôsobené administratívnou chybou) tejto dohody medzi Britskou telefónnou spoločnosťou a Dr. Simonom Stepherdom z Bradfordskej univerzity spôsobilo únik algoritmu na verejnosť. Aj keď články Dr. Stepherda o možných metódach kryptoanalýzy algoritmu A5 boli tajnou službou vyhlásené za neverejné, bolo už len otázkou času, kedy podobné metódy opísali iní.

Šifra A5 je typ prúdovej šifry, pri ktorej sa pôvodná binárna postupnosť transformuje pomocou XOR operácie s pseudonáhodnou binárnou postupnosťou $k[n]$ generovanou algoritmom A5, ktorého štruktúra je znázornená na obrázku 3.




*Obr.3 Generovanie pseudonáhodnej postupnosti v algoritme A5*

Štruktúra je zložená z *troch posuvných registrov* $r_1$ (19-bitový), $r_2$ (22-bitový) a $r_3$ (23-bitový) s lineárnou spätnou väzbou, pričom riadenie posunu jednotlivých posuvných registrov je realizované pomocou **nelineárnej trojvstupovej funkcie C**, ktorá má tri vstupy a tri výstupy. Výstup funkcie *C* určuje, či sa konkrétny register posunie ($c_i = 1$, „go“), alebo zostane stáť ($c_i = 0$, „stop“). Úlohou nelineárnej funkcie *C*, ktorej

pravdivostná tabuľka pre systém GSM je v tabuľke 1, **je zvýšenie krytografickej bezpečnosti algoritmu**. Z tabuľky je zrejmé, že v každom takte je vždy najviac jeden posuvný register neposunutý.

Počiatočné naplnenie je realizované 64-bitovou hodnotou (ktorá sa počas nadviazaného spojenia nemení) získanou zo SIM karty mobilného telefónu algoritmami A3 a A8 počas autentifikačnej fázy a 22-bitovej identifikačnej hodnoty TDMA rámca, ktorá sa mení pre každý paket.

*Tab.1 Pravdivostná tabuľka funkcie C*

| Vstupy | | | Výstupy | | |
|---|---|---|---|---|---|
| $r_1$ bit 10 | $r_2$ bit 12 | $r_3$ bit 12 | $c_1$ | $c_2$ | $c_3$ |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

## *1.3 KRYPTOGRAFICKY BEZPEČNÉ PSEUDONÁHODNÉ GENERÁTORY*

Tieto generátory sú **podstatne zložitejšie** (a teda aj podstatne pomalšie a cenovo náročnejšie) ako predchádzajúce lineárne generátory. Ich základnou **výhodou** je **nemožnosť odvodiť** predchádzajúce alebo nasledujúce bity pseudonáhodnej postupnosti resp. parametre generátora **na základe znalosti určitého úseku** pseudonáhodnej postupnosti. Tieto algoritmy sú založené na zložitosti niektorých matematických problémov v modulárnej aritmetike (podobne ako napr. algoritmus RSA, ...). V ďalšej časti budú uvedené dva typické príklady pseudonáhodných binárnych generátorov z tejto oblasti.

### 1.3.1 PSEUDONÁHODNÝ BITOVÝ RSA GENERÁTOR

**Cieľ:**
Generovanie pseudonáhodnej binárnej postupnosti $b_1, b_2, \ldots, b_k$

**Algoritmus:**

1.) Vygenerovanie dvoch veľkých (tajných) prvočísiel $p$ a $q$, určenie hodnôt $n$ a $\phi$

$$n = pq \tag{1.7}$$

$$\phi = (p-1)(q-1) \tag{1.8}$$

a výber náhodného čísla $e$, $1 < e < \phi$, pre ktoré platí $GCD(e,\phi) = 1$.

2.) Výber náhodného celého čísla $x_0$ (seed) z intervalu $[1, n-1]$.

3.) Pre $i = 1, 2, \ldots k$ určenie[5]

$$x_i = x_{i-1}^e \bmod n \tag{1.9}$$

$$b_i \leftarrow LSB(x_i) \tag{1.10}$$

4.) Výstupná postupnosť je $b_1, b_2, \ldots b_k$.

### 1.3.2 BLUM-BLUM-SHUB PSEUDONÁHODNÝ BITOVÝ GENERÁTOR

Tento generátor (tiež známy ako $x^2 \bmod n$ alebo ako **BBS generátor**) je možné opísať takto:

**Cieľ:**
Generovanie pseudonáhodnej binárnej postupnosti $b_1, b_2, \ldots, b_k$

**Algoritmus:**

1.) Vygenerovanie dvoch náhodných veľkých (tajných) prvočísel $p$ a $q$ pre ktoré platí[6] $p \equiv 3 \bmod 4$, $q \equiv 3 \bmod 4$ a určenie čísla

$$n = pq \tag{1.11}$$

**2.)** Výber náhodného čísla $s$ (seed) z intervalu $[1, n-1]$, pre ktoré platí $GCD(s,n) = 1$ a výpočet $x_0 = s^2 \bmod n$.

3.) Pre $i = 1, 2, \ldots k$ určenie

$$x_i = x_{i-1}^2 \bmod n \tag{1.12}$$

$$b_i \leftarrow LSB(x_i) \tag{1.13}$$

4.) Výstupná postupnosť je $b_1, b_2, \ldots b_k$.

[5] LSB je najmenej významový bit.
[6] Sú to tzv. Blumove (celé) čísla.

## *1.4 TESTOVANIE NÁHODNOSTI*

Existuje niekoľko štatistických testov, ktoré je možné využiť na testovanie kvality (náhodnosti) generovaných (binárnych) postupností (**Frequency test** (monobit test), **Serial test** (two-bit test), **Poker test**, **Runs test**, **Autocorrelation test**) [1]. Niektoré z nich sú dokonca štandardizované. Typickým príkladom je americká norma **FIPS PUB 140-2** s názvom „**Security Requirements for Cryptographic Modules**", ktorá definuje nasledujúce testy (uvedené v originálnej podobe)[7].

*Statistical random number generator tests*. Cryptographic modules that implement a random or pseudorandom number generator shall have the capability to perform statistical tests for randomness. A single bit stream of 20,000 consecutive bits of output from each generator shall be subjected to the all of the following four tests: monobit test, poker test, runs test, and long runs test. For Levels 1 and 2, the tests are not required. For Security Level 3, the tests shall be callable upon demand. For Security Level 4, the tests shall be performed at power-up and shall also be callable upon demand.

*The monobit test*

1. Count the number of ones in the 20,000 bit stream. Denote this quantity by X.
2. The test is passed if 9,725 < X < 10,275 (Type I Error of .0001).

*The poker test*

1. Divide the 20,000 bit stream into 5,000 contiguous 4 bit segments. Count and store the number of occurrences of the 16 possible 4 bit values. Denote *f(i)* as the number of each 4 bit value *i* where $0 \leq i \leq 15$.
2. Evaluate the following:

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

3. The test is passed if 2.16 < X < 46.17 (Type I Error of .0001).

*The runs test*

1. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros that is part of the 20,000 bit sample stream. The incidences of runs (for both consecutive zeros and consecutive ones) of all lengths (≥ 1) in the sample stream should be counted and stored.
2. The test is passed if the runs that occur (of lengths 1 through 6) are each within the corresponding interval specified in the table below. This must hold for both the zeros and ones (i.e., all 12 counts must lie in the specified interval). For the purposes of this test, runs of greater than 6 are considered to be of length 6.

| Length of Run | Required Interval (Type I Error of .0001) |
|---|---|
| 1 | 2,343 □ 2,657 |
| 2 | 1,135 □ 1,365 |
| 3 | 542 - 708 |
| 4 | 251 - 373 |
| 5 | 111 - 201 |
| 6+ | 111 - 201 |

Table 3. *Required intervals for length of runs test*

*The long runs test*

1. A long run is defined to be a run of length 26 or more (of either zeros or ones) for a Type 1 Error of .0001.
2. On the sample of 20,000 bits, the test is passed if there are no long runs.

[7] Tzv. **Security levels** definujú úroveň ochrany kryptografických modulov, vyššia úroveň znamená vyššiu úroveň zabezpečenia.

Je zaujímavé porovnať hodnoty uvádzané v tejto norme s hodnotami v predchádzajúcom vydaní tejto normy [5],[7].

Napr. pri prvom teste boli hranice určené vzťahom

$$9654 < X < 10346 \tag{1.14}$$

pri druhom

$$1,03 < X < 57,4 \tag{1.15}$$

pri treťom

| Dĺžka reťazca | Interval |
|---|---|
| 1 | 2267-2733 |
| 2 | 1079-1421 |
| 3 | 502-748 |
| 4 | 223-402 |
| 5 | 90-223 |
| 6+ | 90-223 |

a pri štvrtom bol reťazec definovaný ako postupnosť **34 alebo viac** rovnakých znakov (jednotiek alebo núl). Na základe uvedených údajov je možné konštatovať, že **požiadavky na vlastnosti generátorov sa sprísnili**. Naviac v októbri 2001 boli opäť modifikované (chybné) hranice tretieho testu a v súčasnosti platia nasledujúce hodnoty [6]:

| Dĺžka reťazca | Interval |
|---|---|
| 1 | 2315-2685 |
| 2 | 1114-1386 |
| 3 | 527-723 |
| 4 | 240-384 |
| 5 | 103-209 |
| 6+ | 103-209 |

V prípade potreby realizovať náročnejšie testy náhodných resp. pseoudonáhodných generátorov je možné využiť špecializované štatistické balíky. V súčasnosti medzi najkvalitnejšie a najlepšie dokumentované verejne dostupné patrí súbor štatistických testov NIST [8]. Tento softvérový balík je poskytovaný v zdrojových kódoch (pre platformu UNIX) a je tak široko modifikovateľný aj koncovým užívateľom. Tento balík okrem kvalitného opisu testovacích metód poskytuje aj základnú metodiku na testovanie náhodných a pseoudonáhodných generátorov.

# LITERATÚRA

[1] Menezes, J.A. – Oorschot, P.C. – Vanstone, S.A.: Applied Cryptography. CRC Press, New York 1997, ISBN 0-8493-8523-7.

[2] Schneier, B.: Applied Cryptography: Protocols, Algorithms and Source Code in C. John Wiley & Sons, Inc., New York 1996, ISBN 0-471-12845-7.

[3] Glisic, S. – Vucetic, B.: Spread Spectrum CDMA Systems for Wireless Communications. Artech House, Inc., Boston 1997, ISBN 0-89006-858-5.

[4] Security Requirements for Cryptographic Modules - FIBS PUB 140-2, Federal Information Processing Standard Publications, 1999. U.S. Department of Commerce/National Institute of Standards and Technology. Dostupné v elektronickej forme **– fips140-2.pdf**.

[5] Security Requirements for Cryptographic Modules - FIBS PUB 140-1, Federal Information Processing Standard Publications, 1994, January 11. U.S. Department of Commerce/National Institute of Standards and Technology. Dostupné v elektronickej forme **– fips1401.pdf**.

[6] Security Requirements for Cryptographic Modules - FIBS PUB 140-2, Federal Information Processing Standard Publications, updated on December 3, 2002. U.S. Department of Commerce/National Institute of Standards and Technology. Dostupné v elektronickej forme **– fips140-2_upd.pdf**.

[7] Klíma, V.: Generátory náhodných čísel: Život je jen náhoda... CHIP 3/1998, s.62-63.

[8] Rukhin, A., Soto, J., Nechvastal , J., Smid, M., Barker, E., Leight, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: 'A statistical test suite for random and pseudorandom number generators for cryptographic applications', NIST Special Publication 800-22, May 15, 2001, pp.1-153, www.nist.gov