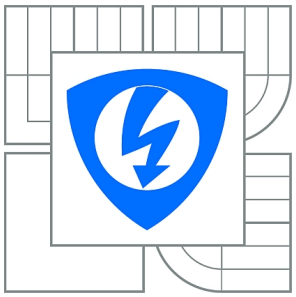
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

GENERÁTOR NÁHODNÝCH ČÍSEL

RANDOM NUMBER GENERATOR

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

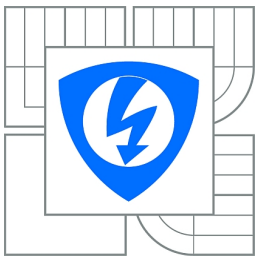
AUTOR PRÁCE — Bc. PETR ZOUHAR

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ SOBOTKA

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Petr Zouhar	***ID:***	78323
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Generátor náhodných čísel

POKYNY PRO VYPRACOVÁNÍ:

Zabývejte se problémem generování náhodných čísel a analyzujte existující generátory náhodných čísel. Zhodnoťte způsoby generování náhodných čísel na základě různých fyzikálních veličin. Pokuste se navrhnout vlastní generátor náhodných čísel. Proveďte analýzu vygenerovaných čísel a zjistěte, zda se jedná skutečně o náhodná čísla.
Poté aplikujte výsledek svého generátoru náhodných čísel na některý z kryptografických algoritmů. Zhodnoťte úroveň šifrování za použití vámi vytvořeného řetězce náhodných znaků.

DOPORUČENÁ LITERATURA:

[1] HAAHR, Mads, Introduction to Randomness and Random Numbers, 2009, dostupne online: http://random.org/randomness/.
[2] SCHNEIER, Bruce. Applied cryptography. 2nd edition. [s.l.] : John Wiley & Sons, 1996. 784 s. ISBN 0-471-11709-9 .

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Jiří Sobotka

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce se věnuje problematice náhodných čísel, jejich generování a použití v kryptografii. Úvod je zaměřen na rozlišení generátorů na náhodné a pseudonáhodné. Dále je uvedeno často používané dělení na softwarové a hardwarové. Jsou zde zmíněny výhody a nevýhody jednotlivých typů a oblast jejich použití. Posléze jsou popsány příklady generátorů náhodných i pseudonáhodných čísel, zejména pak těch hardwarových založených na fyzikálních veličinách jako je rozpad radioaktivního materiálu či využití atmosférického šumu. Následující část je věnována návrhu vlastního generátoru náhodných čísel a popisu jeho funkčnosti. V druhé polovině práce se pak věnujeme oblasti kryptografie. Seznámíme se se základními typy kryptografických systémů, tedy symetrickými a asymetrickými kryptosystémy. Představíme si typické zástupce jednotlivých skupin a jejich vlastnosti. V závěru práce se opět vrátíme k našemu generátoru náhodných čísel a provedeme testy k ověření náhodnosti vygenerovaných čísel a získaných kryptogramů.

KLÍČOVÁ SLOVA

Náhodné číslo, generátor náhodných čísel, generátor pseudonáhodných čísel, kryptografie, baterie testů.

ABSTRACT

The thesis deals with issues of random numbers, their generating and use in cryptography. Introduction of work is aimed to resolution of random number generators and pseudo–random number generators. There is also included often used dividing generators on software and hardware. We mention advantages and disadvantages of each type and area of their use. Then we describe examples of random and pseudorandom numbers, mainly hardware based on physical phenomenon such as the decay of radioactive material or use atmospheric noise. The following part is devoted to suggestion own random number generator and a description of its functionality. In the second half of the work we devote to the field of cryptography. We know basic types of cryptographic systems, namely symmetric and asymmetric cryptosystems. We introduce a typical representant the various type and their properties. At the end of the work we again return to our random number generator and verify the randomness generated numbers and obtained cryptograms.

KEYWORDS

Random number, true random number generator, pseudo–random number generator, cryptography, battery of tests.

ZOUHAR, Petr. *Generátor náhodných čísel*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2010. 46 s. Vedoucí práce byl Ing. Jiří Sobotka.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Generátor náhodných čísel“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Jiřímu Sobotkovi za cenné rady a pomoc při psaní práce.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Diplomová práce je zaměřena na problematiku náhodných čísel a jejich generování. V úvodu se seznámíme s rozdělením generátorů náhodných čísel. Rozlišujeme dvě základní skupiny, a to generátory náhodných a pseudonáhodných čísel. Po úvodní části, věnované základnímu popisu obou typů generátorů, se hlouběji zaměříme na generátory náhodných čísel. Ty bývají hardwarové, založené na využití fyzikálních veličin.

Součástí práce je nejen popis metod a postupů používaných u generátorů náhodných a pseudonáhodných čísel, ale též návrh vlastního programu, který bude sloužit ke generování náhodných čísel. Hodnoty získané námi navrženým generátorem následně podrobíme testům.

Diplomová práce navazuje na semestrální projekt [21].

1 NÁHODNÁ A PSEUDONÁHODNÁ ČÍSLA

Při zmínce „generování náhodných čísel“ si asi lidé většinou představí nejjednodušší představitele ze svého života. Těmi je házení kostkou a losování Sportky. Tedy dva případy, které představují generátory náhodných čísel, jejichž výsledek nemůžeme ovlivnit.

S náhodnými čísly, potažmo jejich generováním, se dnes setkávají denně snad všichni z nás, přestože si to často ani neuvědomujeme. Jedná se zejména o oblast kryptografie, tedy o potřebu zabezpečení elektronických dat, jako je internetové bankovnictví, elektronická pošta a další místa související s přenosem či archivací citlivých informací. Přítomnost generátorů náhodných čísel vyžadují mnohé programy, především potom software určený na simulace nějakého děje (například výpadek datové sítě v náhodném okamžiku, generování náhodného rušení) a modelační programy. To klade jisté požadavky i na programovací jazyky, po kterých je vyžadováno, aby generování náhodných čísel bylo možno implementovat do zdrojových kódů při tvorbě programů.

1.1 Typy generátorů náhodných čísel

Hovoříme-li o náhodných číslech, respektive jejich generátorech, je třeba vždy rozlišovat, zda se jedná opravdu o náhodná čísla, nebo máme na mysli generátory čísel pseudonáhodných. V angličtině se pro odlišení používají zkratky PRNG (Pseudo-Random Number Generator) a TRNG (True Random Number Generator) [5]. Zde vzniká drobný problém, jak tuto zkratku překládat do češtiny. Lze použít několik tvarů: „Generátory pravdivých náhodných čísel“, „Generátory opravdu náhodných čísel“ či „Pravé generátory náhodných čísel“. Nejjednodušší by asi bylo vynechat při překladu slovo „true“ a používat pouze „Generátory náhodných čísel.“ Nastal by ovšem problém s rozlišením, kdy v případě použití posledního tvaru máme na mysli TRNG a kdy generátory obecně. Proto budeme TRNG vždy specifikovat.

Základním rozlišovacím znakem mezi těmito dvěma typy je jejich realizace. V praxi se můžeme setkat s generátory hardwarovými a softwarovými, přičemž ty softwarové bývají obvykle označovány jako generátory pseudonáhodných čísel, neboť „náhoda“ je u nich vypočtena určitým algebraickým vzorcem, popřípadě získána z předem připravených tabulek, a po čase dochází k opakování výstupních čísel. Posloupnost generovaných čísel se tedy pouze jeví jako náhodná, a z tohoto důvodu bývá označována jako pseudonáhodná.[1]

[1] Výstižné definování pseudonáhodných čísel je uvedeno v [4]: „*Generování náhodných čísel tedy spočívá v paradoxu, že náhoda může být vypočtena!*“

S generátory opravdu náhodných čísel (TRNG) se setkáváme v hardwarové podobě. Typickým příkladem ryze mechanického zařízení je losovací zařízení Sportky a ruleta. Nesmíme si však představovat, že se vždy jedná o nějaký mechanický přístroj. Obvykle je to kombinace hardwarové části, která zpracovává nějakou fyzikální veličinu, a softwarové části, která následně na základě získaných hodnot provádí vlastní generování.

Do tohoto rozdělení bychom mohli zařadit i generátory, které jsou sice softwarové bez využití fyzikálních veličin, ale lze je označit za generátory pravých náhodných čísel. Jedná se například o získání čísla na základě nepředvídatelného chování obsluhy PC (např. pohyb myši po podložce, stisknutí kláves).[2] Příklad programu, který generuje čísla na základě pohybu myši, je uveden v kapitole věnující se generátorům pravých náhodných čísel.

1.2 Výhody a nevýhody jednotlivých typů

Výhodou pseudonáhodných generátorů je jejich jednoduchost a rychlost (jsou nám schopny vygenerovat velké množství čísel v krátkém čase). Jedná se obvykle, jak bylo zmíněno výše, o určitou rovnici, což ovšem skrývá potíže v opakování čísel. Proto je třeba, aby opakování nastalo po co nejdelší době (nejlépe samozřejmě nikdy). Vlastností pseudonáhodných generátorů, kterou nemůžeme stoprocentně označit za výhodu ani nevýhodu, je opakovatelnost jejich výsledků. Známe-li počáteční inicializaci (označovanou jako seed-viz kap. 2), lze jejím zadáním získat totožnou výstupní posloupnost čísel. To představuje nevýhodu pro procesy jako je generování kryptografických klíčů (v podstatě to vylučuje možnost použití pseudonáhodných generátorů v této oblasti). Naopak v případě simulací takový deterministický systém oceníme, neboť lze danou simulaci opakovat se stejnými výsledky. Výhodou také je, že vystačíme s počítačem a oproti TRGN nepotřebujeme další zařízení zpracovávající měřenou veličinu.

Předností generátorů pravých náhodných čísel je nemožnost předpovídat generovanou posloupnost. Tyto generátory jsou, narozdíl od pseudonáhodných, neperiodické. Jejich konstrukce je však náročnější o hardwarovou část a s tím souvisí nutnost dalšího zpracování získaných hodnot. To do celého procesu generování vnáší zpoždění, a proto nevýhodou hardwarových zařízení je, že jsou pomalé.

Pro shrnutí vlastností můžeme převzít tabulku od Madse Haahra [5] (tab. 1.1).

[2] Snad jediným znakem, podle kterého bychom je mohli zařadit k hardwarovým, je použití hardwaru počítače (myš, klávesnice).

Tab. 1.1: Porovnání vlastností generátorů náhodných čísel (převzato z [5]).

Vlastnost	**PRNG**	**TRNG**
Výkonnost	vysoká	nízká
Determinističnost	deterministický	nedeterministický
Periodicita	periodický	neperiodický

1.3 Náhodná čísla v praxi

S náhodnými čísli se setkáváme, jak jsme se již několikrát zmiňovali, v mnoha oblastech. Pro shrnutí můžeme opět využít tabulku uvedenou ve zdroji [5]. Z této tabulky (1.2) je patrné, že záleží-li nám na zajištění náhodnosti, častěji využijeme generátory pravých náhodných čísel. Tyto generátory se využívají zejména v oblasti hazardních her a loterie, protože u výherních automatů a různých losovacích zařízení je velmi důležitá nepředvídatelnost.

Pseudonáhodné generátory využijeme především v simulacích a modelování, neboť zde potřebujeme získat co nejrychleji velké množství náhodných čísel. Dalším místem, kde se uplatňují pseudonáhodná čísla, jsou mp3 přehrávače. Detailnější informace se nám sice nepodařilo najít, nicméně lze předpokládat, že využívají určitý algoritmus, který určí posun v seznamu skladeb například na základě hodin či délce poslední přehrávané skladby.

Tab. 1.2: Aplikace využívající náhodná čísla a obvykle využívaný typ generátoru (převzato z [5]).

Oblast využití náhodných čísel	**Použitý generátor**
Loterie a losovací zařízení	TRNG
Bezpečnostní klíče	TRNG
Simulace a modelování	PRNG
Hry a hazardní hry	TRNG
Umění	TRNG/PRNG
Náhodný výběr skladeb v různých přehrávačích	PRNG

2 PSEUDONÁHODNÁ ČÍSLA A JEJICH GENERÁTORY

Nejznámějším generátorem náhodných čísel je lineární kongruentní generátor, který zároveň patří k nejstarším [4]. Pseudonáhodným generátorům se pro úvodní inicializaci zadávají vstupní data, která se doplní do příslušného algoritmu, a poté je provedeno samotné generování.[3] Tyto vstupní parametry se označují jako `seed` (v překladu semínko). Jako inicializační hodnota se často používá systémový čas, což si ukážeme níže. [12, 13]

Ne všechny pseudonáhodné generátory jsou kvalitně navrženy. Jako příklad udává Mads Haahr [5] porovnání generování náhodných čísel v programovacím jazyku PHP (funkce rand()), které provedl Bo Allen. PHP určené pro GNU/Linux dosahuje značně lepších výsledků než verze pro Microsoft Windows.

Generátory pseudonáhodných čísel mají svého typického představitele v podobě funkcí ke generování náhodných čísel v programovacích jazycích.

2.1 Náhodná čísla v C/C++

V jazyce C/C++ je standardně definována funkce `rand()`, která vrací pseudonáhodné číslo mezi nulou a `RAND_MAX` [12]. Aby nebylo generováno stále stejné číslo, je třeba provést nastavení pomocí `srand(unsigned seed)`, které slouží k inicializaci generátoru. V případě, že bychom zadali konstantní číslo, opět bychom při každém spuštění generátoru dostávali stejné „náhodné“ číslo. Je tedy třeba, aby byla vkládaná hodnota proměnná. Pro správnou funkci příkazu `rand()` proto v referenci [12] nacházíme následné nastavení srand: `srand(time(NULL))`. Metoda `time()` nám vrací počet vteřin uplynulých od 1.ledna 1970, a je tedy zajištěna jedinečnost při každém spuštění generátoru. Hodnota vteřin představuje číslo typu integer a proto v případě zadání stejné hodnoty inicializace generátoru získáváme shodné vygenerované číslo. Další možností v Borland C++Builder je funkce `randomize()`, při jejímž zavolání se provede nastavení inicializace na systémový čas (`srand((unsigned) time(NULL))`).

Pro názornou ukázku jsme zvolili následující kód, pomocí kterého jsme získali 5 náhodných čísel, a provedli porovnání výsledků dvou vývojových nástrojů (Borland C++Builder 6 a Microsoft Visual Studio 2008).

[3] Lze si to představit jako běžnou rovnici o x neznámých, kterou lze po jejich dosazení spočítat a stanovit tak výsledné číslo.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
void main(void)
{
    int i;
    //srand(13); // příklad inicializace konstantní hodnotou
    srand(time(NULL)); // inicializace aktuální hodnotou času
    printf("Pet nahodnych cisel od 0 do 9\n");
    for(i=0; i<5; i++)
       printf("%d\n", rand()%10);//výpočet modulo 10 z vygenerovaného čísla
}
```

Při nastavení `seed` na konstantní hodnotu získáváme při každém spuštění v C++Builderu řetězec $0; 2; 1; 4; 4$ a ve Visual Studiu hodnoty $1; 6; 6; 8; 2$. V obou případech tedy generátory pracují stejně, ovšem s odlišnou rovnicí, do níž zadáváme stejnou inicializaci.

2.2 Generování náhodných čísel v jazyce Java

V jazyce Java je situace podobná. Zde se jedná o metodu `Random()`. Popis opět provedeme na části zdrojového kódu.

```
Random rand = new  Random();// inicializace systémovým časem
//rand.setSeed(13); // nastavení inicializační hodnoty
for(int i=0; i<5; i++) System.out.println(rand.nextInt()%10);
```

Inicializaci seed lze zadávat přímo do konstruktoru `Random(long seed)`, popřípadě pomocí `setSeed(long seed)`, jak je vidět v uvedeném kódu. V případě, že konstruktoru nezadáme žádnou vstupní hodnotu, je automaticky využit systémový čas, což odpovídá nastavení inicializační hodnoty na `System.currentTimeMillis()`.

3 GENERÁTORY PRAVÝCH NÁHODNÝCH ČÍSEL

Tato kapitola představuje stěžejní část práce a je věnovaná TRNG. V kapitole 1.1 jsme se již zmiňovali, že se obvykle jedná o generátory závislé na fyzikálních veličinách, přičemž naměřené hodnoty je třeba přenést do počítače a převést je na číslo z požadovaného rozsahu. Může se jednat o rozpad radioaktivních částic, atmosférický šum či využití fotoelektrického jevu. [5]

Generátor opravdu náhodných čísel může být také založen na využití pohybu myši či stisku kláves. Příklad takového generátoru je uveden v následující podkapitole, poté se zaměříme na generátory zpracovávající fyzikální veličiny.

3.1 Získání čísla na základě nepředvídatelného chování obsluhy PC

Jak bylo zmíněno výše, lze mezi generátory pravých náhodných čísel zařadit generátory založené na pohybu myší. Při pohybu se vyhodnocují souřadnice pohybu, z nichž je poté určitým algoritmem vygenerováno číslo ze zadaného rozsahu. Příkladem je program Randomgen [1], který pro výpočet kromě pohybu myši využívá systémové informace, jejichž hodnota se často mění. Konkrétně se jedná o velikost volné paměti a swapovací oblasti.

Při využití klávesnice lze jako vstupní hodnoty brát frekvenci stisku kláves, mezery mezi jednotlivými stlačeními, popřípadě posloupnost stisknutých kláves. K tomuto způsobu generování čísel zmiňuje Mads Haahr [5] nevýhodu, která spočívá v tom, že stisky kláves bývají operačním systémem ukládány do bufferu. Následkem tohoto postupu dochází k jejich nahromadění a až poté k odeslání do programu, který provádí vyhodnocení. Taková stlačení kláves program vyhodnotí, jako kdyby byly provedeny s minimálním časovým odstupem (ne-li v jeden okamžik), a dochází tak ke zkreslení výsledků.

V praxi se proto využívá spíše metoda založená na pohybu myši ve vyznačeném poli na monitoru, a to například v bankách při generování bezpečnostních klíčů.

3.2 Generátory náhodných čísel založené na fyzikálních jevech

Jedná se o generátory, které zpracovávají určité fyzikální veličiny a na základě změřených hodnot provádí generování. Důležitý je u nich převod informací do počítače pro další zpracování.

3.2.1 Generování náhodných čísel pomocí atmosférického šumu

Využití atmosférického šumu ke generování náhodných čísel se věnuje irský univerzitní profesor Mads Haahr z Trinity College Dublin [5]. Ten s kolegy pracoval na návrhu výherních automatů a zabýval se problematikou náhodných čísel. S generováním náhodných čísel začali v roce 1997, kdy jako zdroj vybrali radiový příjem, protože byl levný a poměrně jednoduchý oproti využívání radioaktivního zdroje. Generování bylo založeno na příjmu radiových vln pomocí radiopřijímače, který neměl hlukový filtr[4]. Získaný zvuk (či spíše hluk) byl pomocí zvukové karty převeden do počítače, kde byl programem vytvořeným v C/C++ využit ke generování náhodných čísel. Poté byl vývoj ukončen a Mads Haahr se dále náhodným číslům věnoval na Trinity College v Dublinu. Zde se již zaměřil na využití atmosférického šumu a vytvořil citované stránky [5]. V současnosti je celý systém tvořen více servery umístěnými v různých lokalitách.

Zdrojem atmosférického šumu jsou například bouřky. Atmosférický šum je poměrně snadné získat z běžných radiových vln. Jak udává autor, lze použít šum na pozadí v kanceláři či laboratoři, ale je třeba dávat pozor na zdroje tohoto hluku. K šumu v kanceláři může přispívat například ventilátor počítače. Jedná se ovšem o otáčivé zařízení, a výsledný hluk tak nemusí být čistě náhodný jako v případě atmosférického šumu. Je tedy nutné vyvarovat se zdrojům, které by do měření vnášely prvky periodičnosti.

Ke generování čísel využívá Mads Haahr malé změny v amplitudě atmosférického šumu.

3.2.2 Generování náhodných čísel na základě rozpadu radioaktivního materiálu

Využití rozpadu radioaktivního zdroje je dobrý způsob, jak získat náhodná čísla. Čas rozkladu je zcela nepředvídatelný a tyto okamžiky lze poměrně snadno detekovat a

[4] Autor popisuje problémy se sehnáním takového rádia.

převést do počítače. [5]

Příkladem generátoru náhodných čísel založeného na radioaktivním rozkladu je HotBits. Jedná se o generátor vytvořený Johnem Walkerem v rámci švýcarských stránek Fourmilab [16]. Jako zdroj záření se pro službu HotBits využívá cesium. Přesněji jde o radioaktivní izotop cesia 137Cs, který má poločas rozpadu 30, 17 let. Při beta rozpadu se tento izotop mění na metastabilní jádro barya (137Ba), jenž má poločas rozpadu 156 sekund. Metastabilní baryum-137 je pak zdrojem gama záření.

Hardwarová realizace

K zaznamenávání rozpadu je použit radiační detektor od firmy Aware Electronics, model RM-80, který je k počítači připojen přes sériový port. Data jsou získávána na základě rozdílu mezi dvěma změnami v radioaktivním rozpadu, které se zjišťují Geiger-Müllerovou trubicí[5].

Jedná se již o třetí generaci HotBits generátorů. První dvě generace (z roku 1996 a 1998) byly postaveny na systému Windows. Současná verze, spuštěná v září 2006, využívá Linux[6].

Získávání a zpracování dat

K rozpadu dochází v náhodných okamžicích a tato nepředpovídatelnost se využívá ke generování čísel. Měří se doba mezi dvěma impulzy (T_1) a následně doba mezi dalším párem impulzů (T_2). Jsou-li tyto hodnoty stejné, provede se nové měření. V opačném případě se provede porovnání těchto hodnot. Jestliže je $T_1 > T_2$, výstupem je binární jednička. Pokud je $T_1 < T_2$, je výstupní bit nula.

Abychom se vyvarovali zkreslení výsledků z důvodu nenáhodných systematických chyb měřícího zařízení nebo způsobu měření, jsou při srovnávání následující dvojice hodnot T_1 a T_2 invertovány výsledné bity porovnávání.

Na stránkách si lze nechat vygenerovat náhodná čísla, kde si můžeme vybrat, kolik bytů a v jakém formátu chceme. Pro zajištění bezpečnosti při jejich přenosu je využit protokol https. Po odeslání čísel žadateli jsou příslušné hodnoty vymazány, aby stejná data nebyla poskytnuta dalšímu uživateli. Také je možné si zde stáhnout použitý zdrojový kód a balíček `randomX` určený pro Javu. Použitím tohoto balíčku lze využívat různé pseudonáhodné generátory, nebo využít generátor pravých náhodných čísel HotBits.

[5] Označován také jako Geiger-Müllerův počítač. Jedná se o detektor ionizujícího záření, který je tvořen baňkou s inertním plynem (nejčastěji neon, argon). V této buňce je umístěna soustava elektrod. Na výstupních svorkách se částice nebo kvantum záření projeví v podobě impulzů. Dochází tedy k převodu počtu těchto částic nebo kvant záření na příslušný počet impulzů. [6, 7]

[6] Konkrétně se jedná o distribuci Fedora 5

Práce s radioaktivním materiálem představuje velké nebezpečí v případě nehody a vyžaduje vysokou opatrnost. Obrovskou výhodou je ale nemožnost předpovědět, kdy dojde k rozpadu, a takový generátor lze tedy označit za generátor pravých náhodných čísel.

3.2.3 Využití lávových lamp a služba LavaRnd

V souvislosti s touto službou hovoří Mads Haahr [5] o nejlepším vizuálním přístupu. Jedná se o výtvor Silicon Graphics, který ke generování náhodných čísel využívá snímky lávové lampy. Lavarand, jak byl tento hardwarový generátor náhodných čísel označován, však již neexistuje. Jeden z jejich autorů založil stránky LavaRnd [10], na kterých se dále zabývá problematikou náhodných čísel. Je zde zmíněna i původní verze generátoru Lavarand.

Lavarand

Metoda byla vytvořena v roce 1996. Náhodná čísla generovaná pomocí Lavarand sloužila jako inicializace (seed) pro generátor pseudonáhodných čísel. Přesto se tato metoda řadila mezi generátory pravých náhodných čísel. Jednalo se sice o pseudonáhodný generátor, avšak inicializován byl pomocí skutečně náhodných a nepredikovatelných čísel. Lavarand generoval seed o délce 140 bytů.

Celý systém vyžadoval digitální kameru a 6 lávových lamp. To z toho důvodu, že se mohly spálit žárovky a lampy potřebovaly určitý čas k zahřátí.

LavaRnd

Landon Curt Noll jako jediný z původních autorů pokračoval v práci s náhodnými čísly a vytvořil stránky LavaRnd [10]. K tomu ho zřejmě vedla nemožnost pokračovat v původním projektu, protože na Lavarand má ochranou známku SGI a také na použitý algoritmus má tato společnost patent.

Proto v roce 2000 vzniká LavaRnd využívající ke generování náhodných čísel webovou kameru. Tento generátor dokáže za sekundu vyprodukovat 77 477 až 206 443 náhodných bitů. Jedná se o software s otevřeným zdrojovým kódem. Sami autoři LavaRnd označují jako kryptograficky spolehlivý generátor náhodných čísel. Na první pohled zaujmou podrobně vypracované stránky [10], z nichž jsme čerpali informace.

Proces generování je tvořen třemi částmi:

1. **Digitalizace chaotického zdroje**

   Při tomto kroku se využívá chaotický zdroj označovaný jako LavaCan. To je zařízení používající webovou kameru (autoři se často zmiňují o CCD snímači), která je umístěna v krytu bránícímu přístupu světla. Kamera tak snímá „tmu“, přesněji okolní šum. Pracuje se s modelem YUV, přičemž se využívá jasová složka Y, neboť zbylé dvě složky se z důvodu nepřítomnosti barev příliš neprojeví. Výsledky potom mohou být zobrazeny jako šedotónový obraz (vhodný pro obecné vzory), nebo barevný obraz, ve kterém pixely s vyšším jasem jsou světlejší. Snímek obsahuje 19 200 pixelů a je tvořen jak náhodnými chaotickými daty, tak i nechaotickými. Proto přichází na řadu druhý krok celého procesu.

2. **Digitální směšovač**

   Směšovač přeskupuje data a vytváří rovnoměrně rozložená náhodná čísla. Pro odstranění nechaotických dat se využije algoritmus LavaRnd digitálního směšovače. Nejprve se provádí přeskládání pixelů 17 různými způsoby. Poté následuje 17 různých hashovacích operací pomocí hashovací funkce SHA-1 a 17 operací XOR. Ve výsledku je vytvořeno 340 oktetů náhodných čísel.

3. **Prezentace výsledků**

   Uložená, rovnoměrně rozložená náhodná čísla jsou použita jen jednou k vytvoření náhodné hodnoty na požádání. Poté jsou odstraněna, podobně jako u služby HotBits.

Výhodou této metody je jednoduchost zařízení, nízké pořizovací náklady a také prostorová nenáročnost. Na stránkách tohoto projektu [10] lze najít také spoustu obrázků a podrobných popisů.

4 NÁVRH PROGRAMU PRO GENEROVÁNÍ NÁHODNÝCH ČÍSEL

Součástí této práce je také návrh vlastního programu. Při návrhu jsme vyšli z metody pohybu myši ve vymezeném poli zmiňované v kapitole 3.1, i když z programového hlediska nám z tohoto typu generátoru zůstala jen ona ohraničená oblast. Program je, stejně jako popisovaná metoda, založen na využití náhodné činnosti obsluhy počítače. Nikoliv však na pohybu myši, ale na klicích ve vymezené oblasti.

Důležitým prvkem je neviditelný (nezobrazený) obrázek, který je umístěn na ploše.[7] Prostor pro kliky (označený nápisem „Klikací plocha“) má rozměr 800 x 600 pixelů, protože tyto rozměry má použitý obrázek. Po kliknutí na vyhrazenou oblast se zjistí souřadnice kliku v ose x a y, a poté se načte pixel obrázku na této pozici.[8] Z něj se získají hodnoty barevných složek (R, G, B) a vygenerované číslo se stanoví podle vztahu

$$V = DM + (R + G + B) \bmod (HM - DM + 1), \tag{4.1}$$

kde V je vygenerované číslo, DM a HM představují dolní a horní hranici rozsahu generovaných čísel.

Podobu hlavní části programu lze vidět na obrázku 4.1.

4.1 Menu programu

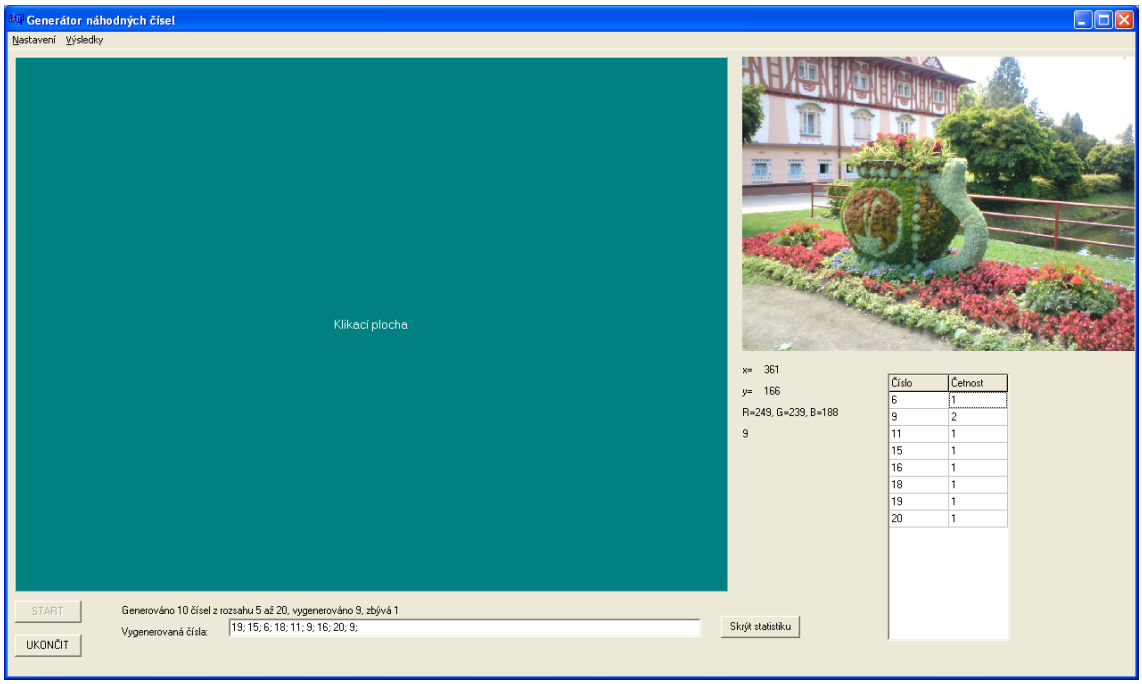
Hlavní menu programu je tvořeno dvěma položkami, které si přiblížíme v následujících podkapitolách.

4.1.1 Nastavení

Možnost hlavního menu Nastavení (viz obr. 4.2) slouží k volbě parametrů generování čísel. První položkou je počet generovaných čísel a volba, zda se mohou čísla opakovat (to pro případ, že by byl program využit například ke generování čísel nějaké loterie, kde se každé číslo smí objevit pouze jedenkrát).

Dále lze nastavit minimální požadovaný odstup mezi dvěma čísly (jedná o vymezení jakéhosi zakázaného pásma). Je to minimální vzdálenost mezi dvěma zpracovávanými kliknutími, která musí být dodržena, aby byl tento druhý klik vyhodnocen. V případě, že tomu tak není, je příslušné kliknutí ignorováno a výpočet se neprovádí. Hodnota této položky musí být z rozsahu hodnot 10–50.

[7] Tento obrázek je možné vidět v případě, že v nastavení vybereme režim prezentace programu.
[8] Počátek číslování (0,0) leží v levém horním rohu plochy.



Obr. 4.1: Okno námi navrženého generátoru.

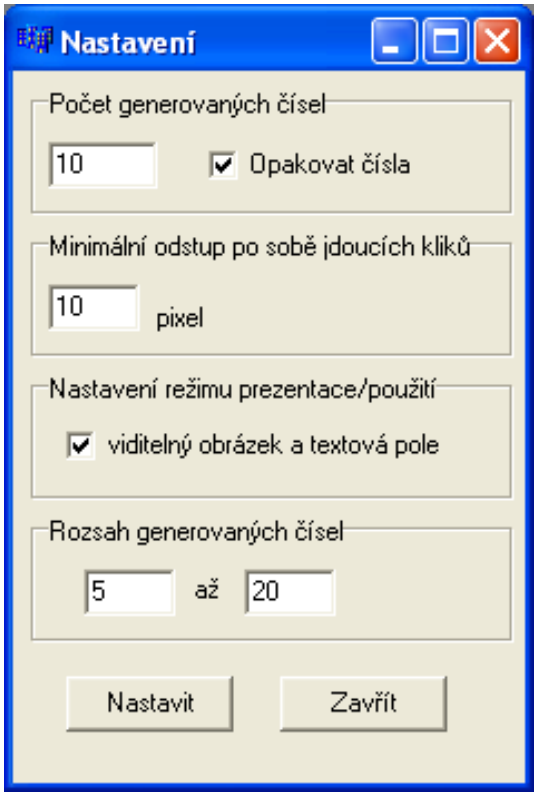
Můžeme si také vybrat, v jakém režimu chceme provádět generování čísel. Při režimu prezentace je zobrazen obrázek i informace o souřadnicích a získaných hodnotách.[9]

Posledním prvkem nastavení je volba rozsahu, který je omezen na 100 čísel. Tato hodnota byla zvolena z důvodu omezeného rozsahu čísel, které součtem hodnot R, G, B můžeme získat (0 až 765).

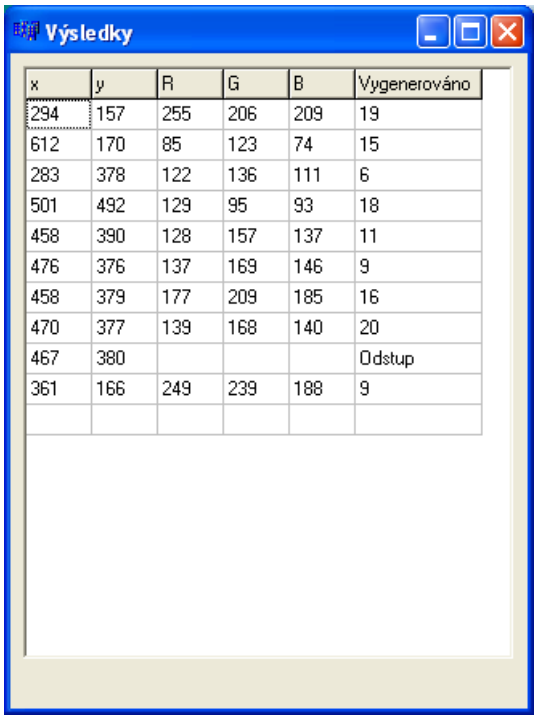
4.1.2 Výsledky

Položka hlavního menu Výsledky slouží k zobrazení tabulky s hodnotami. Jsou zde uvedeny souřadnice kliku, jednotlivé složky R, G, B pixelu odpovídajícího daným souřadnicím a vypočtená hodnota. Nebyl-li dodržen požadovaný odstup, jsou zde zaznamenány pouze souřadnice daného kliku a výpočet se neprovádí. Namísto číselné hodnoty je vložena textová informace „Odstup“. Pokud by nebyla v nastavení zvolena možnost opakování čísel, byl by v případě shody vložen text „Opakuje se“. Příklad tabulky je uveden na obrázku 4.3. Z něj je patrné, že po osmém úspěšném vygenerování proběhl jeden neplatný klik, neboť x-ová i y-ová souřadnice se změnily pouze o 3 pixely, a bod tak nepřekročil nastavenou hranici (v tomto případě 10 pixelů).

[9] Režim prezentace byl vytvořen pro snazší prezentaci programu při obhajobě práce.



Obr. 4.2: Možnosti nastavení programu.



x	y	R	G	B	Vygenerováno
294	157	255	206	209	19
612	170	85	123	74	15
283	378	122	136	111	6
501	492	129	95	93	18
458	390	128	157	137	11
476	376	137	169	146	9
458	379	177	209	185	16
470	377	139	168	140	20
467	380				Odstup
361	166	249	239	188	9

Obr. 4.3: Ukázka tabulky výsledků.

4.2 Použití programu

Program využívá 12 obrázků, které byly upraveny z fotek na rozměr 800 x 600 pixelů a převedeny do bitmapové podoby. Při spuštění generování tlačítkem **START** se pseudonáhodně vybere jeden z těchto obrázků. Použita je funkce `rand()` a dělení modulo 12, čímž získáme jednu z dvanácti hodnot.

Po startu se barva plochy pro kliknutí změní z šedé na modrozelenou a objeví se textová informace, která nám říká, kolik čísel generujeme, kolik jich již bylo vygenerováno a počet zbývajících čísel. Do komponenty Edit umístěné pod tímto textem jsou vypisována jednotlivá čísla oddělená středníkem (viz obr. 4.1). Po ukončení generování panel pro kliky opět zešedne. Tlačítkem **UKONČIT** můžeme přerušit proces generování. Tlačítko **Zobrazit statistiku** slouží k zobrazení tabulky, ve které jsou v prvním sloupci uvedena vygenerovaná čísla, ve druhém pak jejich četnost.

V možnosti Výsledky v hlavním menu si lze prohlédnout jednotlivé kliky, a to i v průběhu generování (viz obr. 4.3).

Příklad výpočtu náhodného čísla pro hodnoty z obr. 4.1, dosazené do vzorce 4.1:

$$V = 5 + (249 + 239 + 188) \bmod (20 - 5 + 1) = 5 + 676 \bmod 16 = 9. \qquad (4.2)$$

K dolní mezi 5 je přičten výsledek dělení modulo 16. Výstupem generátoru jsou tedy čísla 5 až 20.

4.3 Rozložení jednotlivých hodnot

V této závěrečné části se podíváme na volbu obrázků a získané hodnoty. Obrázky byly vybrány náhodně z fotek z dovolených a z domova, o to více nás překvapily výsledky zkoumání pravděpodobnosti vygenerování jednotlivých číslic při zvoleném rozsahu 0–9. Fotky jsme otestovali pixel po pixelu a zaznamenávali počet výskytu jednotlivých hodnot. Výsledky jsou uvedeny v tabulce 4.1.

Při již zmiňovaném rozlišení 800 x 600 pixelů to znamená, že obrázek je tvořen 480 000 body. Při rovnoměrném rozložení a stejné pravděpodobnosti (10 %) bychom tedy měli mít možnost každé číslo získat v 48 000 bodech. Obrázky jsme dle výsledků rozdělili do čtyř skupin.

První skupinu tvoří ty, které se svými hodnotami vešli do tolerance ±1000 (tedy počet jednotlivých číslic se pohybuje v rozmezí 47 000–49 000). Jsou to obrázky 2, 6, 7, 9 a 10. Druhou skupinu tvoří snímky 1 a 5, které mají hodnoty v rozmezí ±2000 vzhledem k číslu 48 000. Předposlední skupinu představují obrázky 4 a 11, jejichž hodnoty se pohybují v rozsahu 45 500–50 500). Zbylé tři fotografie pak tvoří čtvrtou skupinu, která má vždy u některé hodnoty poměrně extrémní četnost. Obrázky 3 a 12 mají výrazně vyšší výskyt čísla 5, zbylé hodnoty jsou opět rozloženy

v úzkém rozsahu. Specifický je pak také obrázek 8, který má nejméně hodnot uprostřed rozsahu, a je tedy opakem k předchozí dvojici fotek.

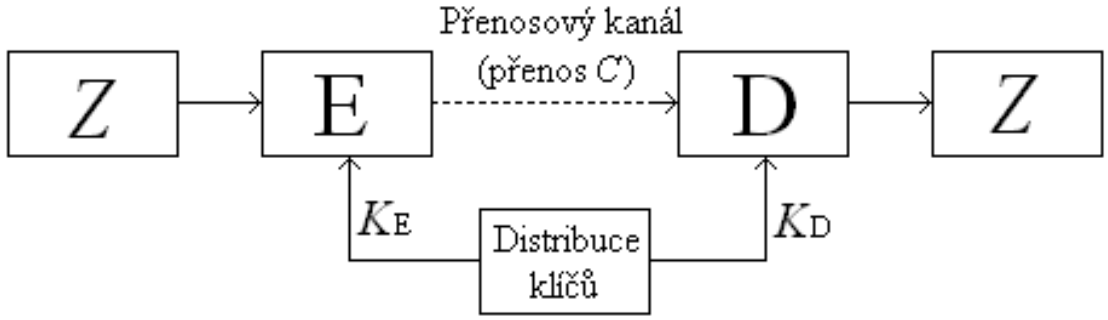
Tab. 4.1: Počet číslic 0–9, které lze vygenerovat v obrázcích.

	0	1	2	3	4	5	6	7	8	9
1	48860	48442	46569	48296	49384	49700	46817	47549	48059	46324
2	47049	48291	48839	48463	47744	48709	47963	47147	48381	47414
3	46805	46282	46223	46509	48160	61293	46130	46165	46510	45923
4	46862	49694	47758	47456	48099	48946	46693	46967	50165	47360
5	47039	47181	49367	48068	48077	49275	48229	47654	47218	47892
6	48167	47937	48062	47893	48524	47466	47763	47824	48247	48117
7	48115	48054	48491	48288	47543	48350	47930	47493	48144	47592
8	55872	54419	56443	42836	36737	40035	48594	40420	48734	55910
9	48950	48111	47410	48118	48250	47342	47777	48313	47830	47899
10	48607	47316	48037	48632	47795	47985	48578	47730	47136	48184
11	45582	47512	47813	47028	48687	48369	50281	47427	48544	48757
12	44521	45015	45578	45130	46245	74945	44648	44229	45130	44559

5 NÁHODNÁ ČÍSLA V KRYPTOGRAFII

V úvodu práce jsme se zmínili, že významnou oblast využití náhodných čísel představuje kryptografie. Nyní se tedy blíže seznámíme se základním dělením kryptografických systémů a s jejich přednostmi či nevýhodami. Hlavním podkladem pro tuto část je [11] a skripta k předmětům Bezpečnost informačních systémů [2] a přednášky předmětu Kryptografie v informatice [18, 20].

Rozlišujeme systémy symetrické a asymetrické, které se vzájemně odlišují způsobem použití klíče (klíčů) k šifrování a dešifrování. Obecné schéma těchto procesů je uvedeno na obrázku 5.1.



Obr. 5.1: Obecné schéma kryptografického systému.

Z ... zpráva v nešifrované podobě (prostý text)
C ... kryptogram (zašifrovaná zpráva)
E ... šifrování (encryption)
D ... dešifrování (decryption)
K_{E} ... šifrovací klíč
K_{D} ... dešifrovací klíč

Šifrování zprávy pomocí klíče K_{E} provádíme podle vztahu

$$C = \mathrm{E}(Z, K_{\mathrm{E}}). \tag{5.1}$$

Příjemce poté původní zprávu získá dešifrováním kryptogramu za pomoci dešifrovacího klíče K_{D}

$$Z = \mathrm{D}(C, K_{\mathrm{D}}). \tag{5.2}$$

Vztah mezi klíči K_{E} a K_{D} lze vyjádřit pomocí funkce

$$K_{\mathrm{D}} = f(K_{\mathrm{E}}). \tag{5.3}$$

V případě symetrických kryptosystémů se jedná o tajné klíče, protože dešifrovací klíč K_{D} lze v reálném čase odvodit ze znalosti klíče šifrovacího. Případně se může jednat o totožný klíč.

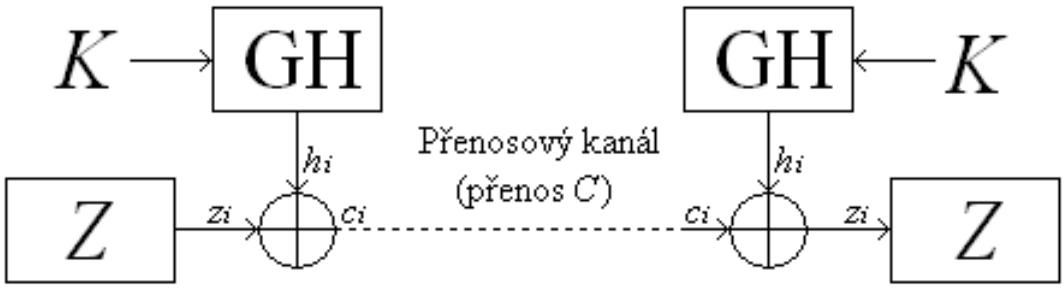
Oproti tomu je u asymetrické kryptografie využita dvojice klíčů (soukromý a veřejný), jejichž použití závisí na tom, zda provádíme šifrování nebo digitální podpis.

5.1 Symetrické kryptosystémy

Rozlišujeme dva typy symetrických kryptosystémů – proudové a blokové šifry. Pro šifrování se využívá substituce znaků zprávy, transpozice nebo kombinace obou těchto metod. Při substituci se jednotlivé znaky zprávy nahrazují jiným znakem použité abecedy. Transpozice představuje přeskládání znaků zprávy. Tyto metody nejsou příliš bezpečné, protože při kryptoanalýze prozrazují informace o původním textu. Proto se začala používat kombinace obou metod, kdy se provede substituce a následně transpozice zprávy. Tento kombinovaný způsob zajistí bezpečnější ukrytí informace.

5.1.1 Proudová šifra

Jedná se o jednoduchý příklad symetrického kryptosystému, kde zpráva je zpracovávána po bitech. Základní operací výpočtu je funkce XOR (sčítání modulo 2). Vstupem šifrátoru je bit zprávy z_i a bit hesla h_i, výstupem pak bit šifry c_i. Dešifrátor pak na základě c_i a stejné posloupnosti bitů hesla h_i získá bity zprávy, ze kterých složí přenášenou zprávu.



Obr. 5.2: Schéma proudové šifry.

Výpočet kryptogramu a následné dešifrování lze popsat pomocí dvojice rovnic

$$c_i = z_i \oplus h_i, \tag{5.4}$$

$$z_i = c_i \oplus h_i. \tag{5.5}$$

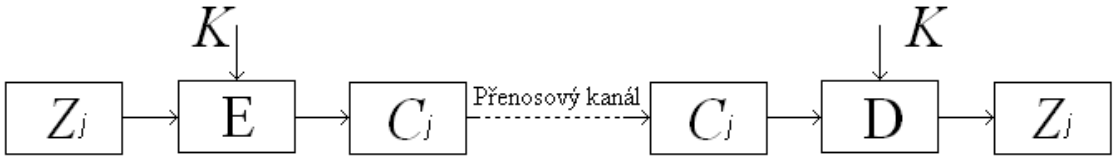
Důležitou součástí šifrátoru a dešifrátoru je generátor hesla GH. Jedná se o generátor náhodných čísel, který na základě inicializačního klíče K generuje jednotlivé bity hesla.

Tento generátor může být pseudonáhodný, tvořený obvodem provádějícím výpočet na základě počáteční inicializace klíčem K. Příkladem takového generátoru je lineární generátor s posuvným registrem a lineární zpětnou vazbou uvedený v [2].

Druhou možností je využít hodnoty získané z generátoru opravdu náhodných čísel, které jsou uloženy na záznamovém zařízení. Dané paměťové medium je třeba bezpečně předat oběma komunikujícím stranám. Příkladem takové šifry je dokonalá šifra (Vernamova šifra), která má na klíč (potažmo heslo) tři požadavky. Klíč musí být náhodný, stejně dlouhý jako zpráva a smí být použit pouze jednou.

5.1.2 Bloková šifra

Druhým typem symetrických kryptosystémů je bloková šifra. Ta zprávu dělí na bloky dat délky N. Využívají se bloky délky 64, 128 a 256 bitů.



Obr. 5.3: Schéma blokové šifry.

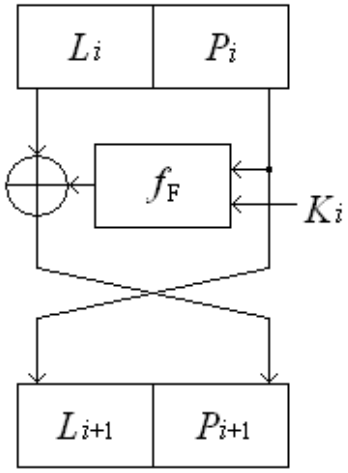
U blokové šifry získáme šifrováním j-tého bloku zprávy Z_j j-tý blok kryptogramu C_j. Samotné šifrování a dešifrování však není tak jednoduché jako v případě proudové šifry. Probíhá opakovaně, kdy blok zprávy je zašifrován a výsledný blok je přiveden na vstup dalšího šifrátoru. Celý proces šifrování se tak sestává z několika dílčích šifrátorů, které bývají označovány jako rundy a jejich propojení pak jako kaskáda rund. V praxi se pak z důvodu zvýšení bezpečnosti využívá nejméně 10 rund. Tento postup, zmíněný pro šifrování, platí také pro dešifrování, kde z bloku kryptogramu C_j získáme po provedení kaskády dešifrovacích rund původní blok zprávy Z_j.

Nejznámější a nejčastěji využívanou rundou je Feistelova runda (obr. 5.4), která vstupní blok rozděluje na levou (L_i) a pravou (P_i) polovinu. Hodnoty výstupního bloku jsou pak dány podle vztahů 5.6 a 5.7.

$$L_{i+1} = P_i, \tag{5.6}$$

$$P_{i+1} = L_i \oplus f_{\mathrm{F}}, \tag{5.7}$$

kde f_{F} představuje rundovní funkci.



Obr. 5.4: Schéma Feistelovy rundy.

5.1.3 Vlastnosti proudové a blokové šifry

Symetrické kryptosystémy jsou obecně rychlejší než asymetrické, přesto porovnáním proudové a blokové šifry najdeme i mezi oběma typy rozdíly v rychlosti. Ta je největší předností proudové šifry, spolu s její jednoduchostí. Každý znak zprávy je na základě hesla zašifrován do jiného znaku zprávy, tudíž případná chyba má vliv pouze na aktuální znak. Nevýhodou proudových šifer je jejich malá úroveň zabezpečení. V [2] je zmíněn útok pomocí modifikace kryptogramu, kdy útočník zná odesílanou zprávu a na základě zachyceného kryptogramu může získat heslo pomocí kterého přepošle modifikovanou zprávu.

Blokové šifry jsou konstrukčně složitější. Již z principu svého fungování jsou pomalejší, neboť u nich probíhá šifrování ve více rundách. Nevýhodou je také zpoždění vzniklé čekáním na načtení celého bloku. S využíváním bloků souvisí také šíření chyb, které ovlivní všechny znaky v bloku zprávy. Co se však týče bezpečnosti, jsou lepší než proudová šifra. Neumožňují útok modifikací kryptogramu, případná změna by byla objevena při dešifrování.

5.2 Asymetrické kryptosystémy

Asymetrické kryptosystémy využívají k zajištění bezpečnosti výpočetně náročné problémy, které by útočník musel spočítat, aby získal informace o kryprosystému. Využívá se jednocestnosti daných matematických problémů, kdy jsme schopni poměrně snadno provést určitý výpočet, ale případný útočník ze známých hodnot není schopen dopočítat vstupní hodnoty. Nejvýznamějším představitelem asymetrických kryptosystémů je algoritmus RSA, který využívá problému faktorizace velkých čísel. Druhým nejznámějším představitelem je Diffie-Hellmanův protokol založený na problematice diskrétního logaritmu.

5.2.1 RSA

Zkratka tohoto algoritmu je odvozena od jmen autorů (Rivest, Shamir a Adleman). RSA slouží k vytvoření páru veřejného a tajného klíče se snahou znemožnit řešení rovnice 5.3, a zabránit tak možnosti napadení komunikace útočníkem. Jak již bylo zmíněno výše, tento kryptosystém využívá problému faktorizace velkých čísel.

Postup výpočtu

1. Volba 2 velkých prvočísel p a q.
2. Výpočet modula n:
$$n = p * q. \tag{5.8}$$
3. Výpočet pomocného čísla r:
$$r = (p - 1) * (q - 1). \tag{5.9}$$
4. Volba veřejného klíče VK, který musí být nesoudělný s r.
5. Stanovení soukromého klíče SK na základě podmínky:
$$(VK * SK) \bmod r = 1. \tag{5.10}$$
6. Zveřejnění VK a n.

Šifrování a vlastnosti RSA

Délka šifrované zprávy musí být kratší než je n. Proto se zpráva Z rozdělí na bloky Z_j ($Z_j < n$). Výsledkem jsou pak bloky kryptogramu C_j, které příjemce dešifruje a ze získaných bloků zprávy opět složí přenášenou informaci. Šifrování a dešifrování pomocí RSA se provádí podle dvojice vzorců

$$C_j = {Z_j}^{VK} \bmod n, \tag{5.11}$$

$$Z_j = {C_j}^{SK} \bmod n. \tag{5.12}$$

RSA (a asymetrické systémy obecně) je pomalejší než symetrické kryptosystémy, neboť práce s velkými čísly zvyšuje výpočetní náročnost. Poskytuje však vyšší bezpečnost, která je zajištěna tím, že útočník není schopen faktorizovat velké číslo n, jehož doporučovaná velikost je 768, 1024, 2048 bitů.

RSA se proto využívá k přenášení krátkých zpráv (digitální podpis, klíče symetrického kryptosystému).

5.2.2 Diffie-Hellmanův protokol

Tento protokol je představitelem asymetrického kryptosystému založeného na problému řešení diskrétního logaritmu. Ty se nepoužívají k šifrování zpráv, protože kryptogram by byl dvakrát větší. Využívají se na ustavení klíče pro symetrickou kryptografii.

Postup výpočtu

Podobně jako RSA má i Diffie-Hellmanův protokol veřejné prvky. Tím je generátor g a velké prvočíslo p, které ve výpočtu přestavuje modulo.

1. Obě komunikující strany si zvolí náhodné číslo - jeden uživatel ($\mathrm{U_A}$) číslo a a druhý uživatel ($\mathrm{U_B}$) číslo b. Tato čísla jsou tajná a představují soukromé klíče uživatelů.
2. $\mathrm{U_A}$ vypočítá číslo

   $$X = g^a \bmod p, \tag{5.13}$$

   $\mathrm{U_B}$ vypočítá

   $$Y = g^b \bmod p. \tag{5.14}$$

   Tyto zprávy si komunikující strany vzájemně vymění.
3. Z informace získané od protější strany uživatelé spočítají klíč K. $\mathrm{U_A}$ podle vzorce

   $$K = Y^a \bmod p, \tag{5.15}$$

   $\mathrm{U_B}$ podle vzorce

   $$K = X^b \bmod p. \tag{5.16}$$

Bezpečnost tohoto protokolu vychází ze skutečnosti, že případný útočník zná veřejné prvky p, g a může odchytit zprávy X, Y. Na základě těchto hodnot však není schopen stanovit číslo a nebo b, a nemůže tak odhalit tajný klíč K. Velikost p by měla být podobně jako n v případě RSA 768, 1024, nebo 2048 bitů.

5.3 Porovnání symetrických a asymetrických kryptosystémů

Na závěr této kapitoly věnované základům kryptografie je vhodné shrnout vlastnosti symetrických a asymetrických kryptosystémů.

Symetrické kryptosystémy jsou rychlé, a proto jsou využívány k šifrování objemných dat. Jejich slabinou je bezpečná distribuce a správná volba klíčů. Důležitá je jejich délka, od níž se odvíjí počet klíčů, které mohou být použity. Klíče musí být uchovány v tajnosti.

Asymetrické kryptosystémy řeší problém bezpečného přenosu klíče symetrických kryptosystémů, oproti kterým jsou výrazně pomalejší. Využívají se také pro digitální podpis.

6 NÁHODNOST GENEROVANÝCH ČÍSEL A JEJICH VYŽITÍ V KRYPTOGRAFII

V této části se zaměříme na náš generátor náhodných čísel. Pokusíme se získat informace, které nám pomohou pro zařazení generátoru. Tedy zda je opravdu náhodný, nebo „jen“ pseudonáhodný. Jako nejvhodnější místo pro otestování námi vygenerovaných čísel se jeví proudová šifra zmiňovaná v kapitole 5.1.1.

6.1 Náhodnost čísel

Podíváme se nyní na testování náhodnosti čísel. Vyjdeme z pramenů [15], kde jsou uvedeny pouze klasické testy, a [19], kde jsou navíc zmíněny i testy teoretické. Ty se v podstatě zabývají metodou návrhu a konstrukcí generátoru náhodných čísel, ne čísly samotnými

Mezi klasické testy patří test frekvence, n-tic, sérií, rozložení sérií a autokorelační test. Často se potom vyskytují testovací baterie, což jsou vlastně programy provádějící více typů testů. Příklady baterií:

- FIPS 140 - 1
- FIPS 140 – 2
- NIST Statistical Test Suit
- DIEHARD
- ENT

Například NIST obsahuje 16 testů a DIEHARD 15.

6.1.1 DIEHARD

DIEHARD [8] jsme měli možnost vyzkoušet. Bylo to však bohužel pouze na ukázkovém souboru, který jsme si vytvořili podle návodu, který je součástí programu. Baterie totiž pracuje s velkým blokem dat (v manuálu se uvádí, že vyžaduje binární soubor velikosti 10–11 MB, což pro náš program představuje nutnost vygenerovat 10 milionů hodnot). V případě malého souboru je testování ukončeno předčasně v jeho průběhu. Jak udává [15], tato baterie je oproti NIST méně vhodná pro testování v oblasti kryptografie.

6.1.2 ENT

Druhou testovací baterií, kterou jsme vyzkoušeli, je ENT [17]. Autorem programu je John Walker, tvůrce generátoru náhodných čísel na základě rozpadu radioaktivního

materiálu (HotBits) [16]. Jak sám autor uvádí, jedná se o test sekvencí pseudonáhodných čísel.

Výsledkem je stanovení entropie, možné komprese, testu chí-kvadrát, aritmetického průměru, Monte Carlo hodnoty pro π a koeficientu sériové korelace. Při popisu vyjdeme z informací uvedených na stránce programu [17].

Program umožňuje volbu zpracování podle zadaných parametrů, my si zde uvedeme možnosti, které jsme využili. Volba -b značí, že vstupní soubor bude brán po bitech a ne bajtech, tedy zpracovávat a vyhodnocovat se budou hodnoty 0 a 1 ve zdroji. Druhá použitá volba -c zajistí vypsání četnosti jednotlivých znaků ve vstupním souboru. Výstupem kromě výše zmíněných hodnot je tak tabulka, která obsahuje hodnotu vyjádřenou dekadicky, odpovídající znak, počet jeho výskytů ve zdrojovém textu a odpovídající podíl v celém textu.

Entropie vyjadřuje, jaké množství informace je obsaženo v daném znaku. Tedy v případě bajtových znaků je maximem 8 bitů/znak.

Komprese udává procentuální hodnotu, o kterou bychom mohli zmenšit velikost daného souboru při jejím optimálním provedení.

Test rozložení chí-kvadrát je citlivý na chyby v sekvenci pseudonáhodných čísel. Výsledkem je jednak absolutní hodnota, ale nás bude zajímat především druhý údaj, který vyjadřuje pravděpodobnost, že skutečně náhodné hodnoty budou vyšší než vypočítaná hodnota. Nenáhodná data by měla dosahovat výsledku kolem 50%. Tabulka 6.1 udává, jak jsou získané procentuální hodnoty interpretovány z hlediska náhodnosti. Podle této tabulky pak budeme posuzovat získané výsledky.

Tab. 6.1: Vyhodnocení chí-kvadrát testu programem ENT.

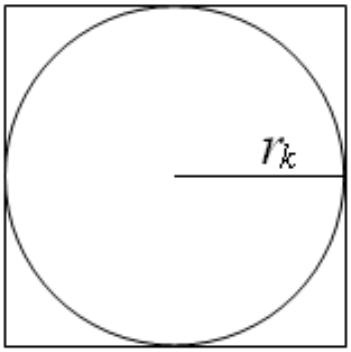
Rozsah hodnot	Náhodnost sekvence
<1% nebo >99%	nenáhodná
1–5% nebo 95–99%	podezřelá
5–10% nebo 90–95%	mírně podezřelá
10–90%	náhodná

Aritmetický průměr určí průměrnou hodnotu bajtu, která by měla být kolem hodnoty 127,5. Byla-li použita volba -b, pak by se tato hodnota měla pohybovat kolem 0,5.

Metoda Monte Carlo pro π. Základem tohoto testu je čtverec a do něj vepsaná kružnice (případně čtvrtkružnice). Každých 6 bajtů je využito jako 24-bitová hod-

nota souřadnice x a y ve čtverci. Bod, jehož vzdálenost je menší než poloměr kružnice (padne do kružnice), je označen jako zásah. Na základě úspěšnosti zásahů se spočítá π' a zjistí se procentuální odchylka od π.

Pro podrobnější popis využijeme zdroj [14]. Základem je zmiňovaný čtverec a v něm vepsaná kružnice poloměru r_k obr. 6.1.



Obr. 6.1: Čtverec a vepsaná kružnice pro metodu Monte Carlo.

Pro plochu čtverce $S_{\check{c}}$ platí vztah

$$S_{\check{c}} = 4r_{\mathrm{k}}^2, \tag{6.1}$$

pro obsah kružnice S_{k}

$$S_{\mathrm{k}} = \pi r_{\mathrm{k}}^2. \tag{6.2}$$

Vyjádříme-li z obou rovnic r_{k}^2, dostaneme porovnáním druhých stran rovnic vztah

$$\frac{S_{\check{c}}}{4} = \frac{S_{\check{c}}}{\pi}, \tag{6.3}$$

ze kterého vyjádříme π a získáme tak vztah daný poměrem obou ploch

$$\pi = 4\frac{S_{\mathrm{k}}}{S_{\check{c}}}. \tag{6.4}$$

Úpravou vztahu 6.4 pak získáme vzorec pro výpočet hodnoty π metodou Monte Carlo (π')

$$\pi' = 4\frac{n_{\mathrm{k}}}{n_{\check{c}}}, \tag{6.5}$$

kde n_{k} je počet zásahů do oblasti kruhu a $n_{\check{c}}$ je počet zásahů do oblasti čtverce (počítá se plocha celého čtverce, tudíž i zásahy do kružnice).

Koeficient sériové korelace (autokorelace) vyjadřuje, do jaké míry závisí každý bajt textu na předchozím bajtu. Pro opravdu náhodná data se hodnota blíží nule, sekvence s hodnotou kolem 0,5 jsou již považovány za nenáhodné.

6.2 Použití vygenerovaných čísel

Jak bylo uvedeno v úvodu této kapitoly, rozhodli jsme se otestovat hodnoty našeho generátoru při šifrování zprávy pomocí proudové šifry. Pro každý z dvanácti obrázků jsme vygenerovali 1000 hodnot, které budou použity jako heslo. Soubor s klíči a obrázky, které jsou součástí programu, jsou uloženy na přiloženém CD.

Pro šifrování jsme využili program CrypTool [3] a jeho možnost klasické symetrické šifry XOR. CrypTool neprovádí operaci XOR po bitech, jak bylo zmiňováno při popisu proudové šifry, ale po bajtech. K zašifrování jednoho znaku zprávy (textu) tak potřebujeme jeden bajt hesla, tedy dva znaky hexadecimální soustavy. Pro 1000 vygenerovaných čísel tudíž může mít zpráva maximálně 500 znaků, aby nedošlo k opakování hesla.[10]

6.2.1 Použití testů ENT na vygenerovaná čísla

Nyní se podíváme na výsledky získané pro vygenerovaná čísla pomocí programu ENT. Otestovali jsme naše čísla v desítkové soustavě (pro kterou byl původně program vytvořený) a v šestnáctkové soustavě (vygenerována čísla z důvodu potřeb CrypToolu). Pro porovnání jsme přidali náhodná čísla HotBits [16]. Využili jsme maximální velikost dat, kterou program poskytuje (2048 B) a vygenerovali 4096 znaků hexadecimální soustavy a binární soubor obsahující 2048 znaků.

Pouze poslední uvedený soubor má výsledek testu chí-kvadrát jiný než 0,01 %, a to 75,47 % pro zpracování po bitech a 36 % při zpracování bajtů. Podobně dopadla metoda Monte Carlo, jejímž výsledkem pro první tři řetězce je $\pi' = 4{,}00$, což představuje chybu 27,32 %. Pouze binární soubor aplikace HotBits dosáhl reálného výsledku $\pi' = 3{,}014662757$, což odpovídá chybě 4,04 %. Hodnota $\pi' = 4{,}00$ značí, že všechny zásahy padly do plochy kružnice, a tudíž i čtverce. V rovnici 6.5 je tedy počet zásahů do kružnice stejný jako zásahy do čtverce (n_{k}=n_{k}).

Ostatní parametry (entropie, aritmetický průměr (AP) a koeficient sériové korelace (Koef.)) jsou uvedeny v tabulce 6.2 – nejdříve vždy pro zpracování po bajtech a následně při bitovém zpracování (parametr -b).

V případě binárních dat HotBits vidíme, že i zbývající testy zvládne na výbornou. Tedy vysoká informační hustota, hodnota aritmetického průměru přibližně uprostřed rozsahu a koeficient sériové korelace je nízký.

Zbylé tři sekvence čísel mají při bajtovém vyhodnocení přibližně 50% nadbytečnost. Průměrná hodnota bajtu je u nich nízká, protože jsou zpracovávány hodnoty

[10] Zde máme důkaz, že náš generátor není vhodný pro proudovou šifru, neboť získat 1000 znaků je náročné. Spíše je vhodný pro generování klíčů, podobně jako generátory založené na pohybu myši.

Tab. 6.2: Porovnání výsledků baterie ENT pro vygenerovaná čísla.

	Entropie	Entropie -b	AP	AP -b	Koef.	Koef. -b
Zdroj čísel	[bit/bajt]	[bit/bit]	[-]	[-]	[-]	[-]
HotBits bin	7,901126	0,999996	128,4443	0,4988	-0,023169	-0,006354
HotBits hex	4,102535	0,964014	56,7320	0,3888	0,251375	-0,055532
Náš dec	3,314372	0,988195	52,4070	0,4361	0,022937	0,030681
Náš hex	3,990519	0,966473	58,2200	0,3926	0,006903	-0,048871

0–9 a A–F, což v dekadickém vyjádření představuje kódy hodnot 48–57 a 65–70. Při bitovém vyhodnocení těchto dvou testů dosahují hodnot srovnatelných s binárním souborem. Koeficient autokorelace je asi jediným faktorem, podle kterého můžeme jednotlivé sekvence plnohodnotně porovnat. Na základě tohoto testu lze usoudit, že výstupem našeho generátoru jsou skutečně náhodná data. Trochu překvapivý je pak vysoký koeficient 0,251375 u sekvence hexadecimálních znaků Hotbits.

6.2.2 Použití testů ENT na zašifrovaný text

V této kapitole se zaměříme na náhodnost zašifrovaného textu pomocí vygenerovaných hodnot. Jako text určený k šifrování byl použit úryvek z díla Babička od Boženy Němcové [9]. Pro porovnání opět využijeme sekvence získané aplikací HotBits. Zaměříme se na hodnoty rozložení chí-kvadrát, aritmetický průměr, metodu Monte Carlo a koeficient autokorelace.

Pro vyhodnocení prvního z uvedených testů využijeme tabulku 6.1. Nenáhodný je kryptogram získaný pomocí sekvence hexadecimálních znaků HotBits. Podezřelé pak jsou výsledky získané pomocí obrázku 7 a 12, mírně podezřelé obrázky 4 a 5. Při zpracování po bitech jako nenáhodný vychází pouze kryptogram získaný sekvencí čísel od obrázku 4. Ostatní šifrované texty jsou náhodné.

V tabulce 6.3 jsou uvedeny výsledky aritmetického průměru pro zpracování po bitech (parametr -b) i bajtech a chyba hodnoty π' metody Monte Carlo. Aritmetický průměr se ve všech případech pohybuje kolem požadované hodnoty 0,5 respektive 127,5. Při bitovém zpracování se hodnoty liší v jednotkách tisícin, trochu vyšší odchylku má kryptogram získaný pomocí řady čísel třetího obrázku. Ve výsledcích bajtového zpracování se objevují větší odchylky pro hesla získaná programem HotBits a náš obrázek číslo 8. U poloviny kryptogramů se chyba hodnoty π' vešla pod 1 %. Největší chybu (blížící se 10 %) vykazuje obrázek 6.

Z hlediska koeficientu sériové korelace lze všechny výsledky považovat za náhodné, neboť tato hodnota se ve všech případech pohybuje v řádu setin a tisícin. Výjimkou je opět sekvence hexadecimálních znaků Hotbits, s hodnotou 0,175217.

Tab. 6.3: Porovnání výsledků baterie ENT pro kryptogramy.

	Aritmetický průměr -b	Aritmetický průměr	Chyba π'
zdroj hesla	[-]	[-]	[%]
HotBits	0,5055	138,9074	0,80
obrázek 1	0,4951	126,7269	0,97
obrázek 2	0,4986	131,6829	0,80
obrázek 3	0,5130	128,4745	0,97
obrázek 4	0,5000	129,4514	4,51
obrázek 5	0,4945	124,1736	2,57
obrázek 6	0,4965	131,2454	9,81
obrázek 7	0,4974	122,7477	2,74
obrázek 8	0,5058	136,9977	0,97
obrázek 9	0,4939	129,2477	4,33
obrázek 10	0,4974	126,8009	2,57
obrázek 11	0,5032	127,9097	6,10
obrázek 12	0,5064	131,5231	0,80

Posloupnost hodnot z aplikace HotBits nás nemile překvapila, neboť jsme očekávali, že bude dosahovat lepších výsledků. O to více nás těší úspěch hodnot získaných naším generátorem náhodných čísel.

7 ZÁVĚR

Rozlišujeme dvě skupiny generátorů náhodných čísel. Setkáme-li se se softwarovými generátory náhodných čísel, obvykle se jedná o pseudonáhodné generátory. Ty jsou vhodné zejména pro případy, kdy potřebujeme v poměrně krátké době získat mnoho náhodných hodnot. Jejich typické využití je v simulačních programech.

Generátor skutečně náhodných čísel má většinou hardwarovou část, která slouží k měření určité veličiny. Důležitou část návrhu představuje způsob převodu získaných dat do počítače. Co se týče generátorů pravých náhodných čísel, asi nejvíce nás zaujala možnost generovat čísla na základě „tmy“. Tedy přesněji šumu snímaného prostřednictvím webové kamery umístěné v neprůhledné krabičce. Další možností je použití atmosférického šumu, případně rozpadu radioaktivního zdroje.

Námi navržený program využívá náhodného chování obsluhy, a nejvíce se tak podobá metodě generování náhodných čísel založené na pohybu myši ve vymezeném okně. Při spuštění se na pozadí načte jeden ze 12 obrázků. Podle souřadnic kliku je pak vybrán příslušný pixel, který se podle zadaných parametrů zpracuje. Původně byl program určen pouze pro hodnoty desítkové soustavy. U použitých obrázků nás překvapila rovnoměrnost generovaných hodnot.

Pro aplikaci čísel jsme zvolili proudovou šifru a za tím účelem jsme použili program CrypTool. Z důvodů jeho požadavků na heslo složené ze znaků hexadecimální soustavy jsme upravili náš program a vygenerovali klíče o délce 1000 znaků pro každý obrázek. Pro srovnání jsme použili sekvenci znaků generátoru náhodných čísel HotBits, který je založen na radioaktivním rozpadu.

Pro testování jsme využili program ENT. Na základě testů jsme rozhodli, že náš program je generátorem náhodných čísel. Rychlost generování je však nízká, o čemž jsme se přesvědčili při požadavku baterie DIEHARD na vstup velikosti 10 MB. Program je tedy vhodný pro generování klíčů podobně jako zmiňovaný generátor založený na pohybu myši.

Text zašifrovaný pomocí vygenerovaných posloupností náhodných čísel je pak v rámci možných hodnot bajtů rovnoměrně rozložen.

LITERATURA

[1] BrotherSoft.com. *Download True random number generator, True random number generator 1.0 Download* [online]. c2002-2009 [cit. 2009-11-21]. Dostupné z WWW: <http://www.brothersoft.com/true-random-number-generator-169743.html>.

[2] BURDA, Karel. *Bezpečnost informačních systémů.* [online]. Brno : FEKT VUT v Brně, 1.11.2005 [cit. 2010-05-21]. Dostupné z WWW: <https://www.vutbr.cz/www_base/priloha.php?dpid=23579&lang=0>.

[3] Deutsche Bank / Contributors. *CrypTool* [počítačový program]. Ver. 1.4.21. c1998–2010 [cit. 2010-05-21]. Dostupné z WWW:<http://www.cryptool.org/>.

[4] GLOMBÍKOVÁ, Viera *Generování náhodných čísel : Pseudonáhodná čísla* Přednáška [online]. 2008 [cit. 2009-11-21]. Dostupné z WWW: <http://www.kod.tul.cz/info_predmety/Psi/prednasky_2007/prednaska_08/Generovan_08.pdf>.

[5] HAAHR, Mads. *Introduction to Randomness and Random Numbers* [online]. c1998-2009 [cit. 2009-11-03]. Dostupné z WWW: <http://www.random.org/>.

[6] HÁJEK, Jan. *Elektronické hledače* [s.l.]: BEN - technická literatura, 2002. 112 s. Ukázka knihy. Dostupné z WWW: <http://www.ben.cz/_d/ukazka/121047u.pdf>.

[7] KASKA, Stanislav, KAPINUS, Lukáš. *Charakteristika a mrtvá doba Geiger-Müllerova počítače* [online]. Jihočeská univerzita v Českých Budějovicích. Pedagogická fakulta. Katedra fyziky. 2006 [cit. 2009-12-08]. Dostupné z WWW: <http://mvt.ic.cz/tri/spm/01.pdf>.

[8] MARSAGLIA, George. *DIEHARD* [počítačový program]. c1995 [cit. 2010-05-21]. Dostupné z WWW: <http://www.stat.fsu.edu/pub/diehard/>.

[9] NĚMCOVÁ, Božena. *Babička.* 1. vyd. ve Světové četbě. Praha : SNKLHU, 1955. 291 s, s.4. Dostupné z WWW: <http://www.bibliohelp.cz/sites/default/files/fulltext/Babicka.pdf>.

[10] NOLL, Landon Curt, COOPER, Simon, PLEASANT, Mel. *LavaRnd* [online]. c2001-2003 [cit. 209-11-22]. Dostupné z WWW: <http://www.lavarnd.org/>.

[11] SCHNEIER, Bruce. *Applied cryptography.* 2nd edition. [s.l.] : John Wiley & Sons, 1996. 784 s. ISBN 0-471-11709-9 .

[12] *Srand - C++ Reference* [online]. c2000-2009 [cit. 2009-11-21]. Dostupné z WWW: <http://www.cplusplus.com/reference/clibrary/cstdlib/srand/>.

[13] Sun Microsystems. *Random (Java 2 Platform SE v1.4.2)* [online]. c2003 [cit. 209-11-22]. Dostupné z WWW: <http://java.sun.com/j2se/1.4.2/docs/api/java/util/Random.html>.

[14] ŠLÉGR, Jan. *Metoda Monte Carlo, Generátory náhodných čísel* [online]. Hradec Králové : Univerzita Hradec Králové, Pedagogická fakulta, Katedra fyziky a informatiky, 10.3.2007 [cit. 2010-05-21]. Dostupné z WWW: <http://www.black-hole.cz/soubory/mc.pdf>.

[15] TESAŘ, Petr. *Generátory náhodných bitů* [online]. 24.5.2006 [cit. 2010-05-21]. Dostupné z WWW: <http://www.comtel.cz/files/download.php?id=2439>.

[16] WALKER, John. *HotBits: Genuine Random Numbers* [online]. May, 1996, last update: September 2006 [cit. 2009-12-08]. Dostupné z WWW: <http://www.fourmilab.ch/hotbits/>.

[17] WALKER, John. *Pseudorandom Number Sequence Test Program* [online]. January 28th, 2008 [cit. 2010-05-21]. Dostupné z WWW: <http://www.fourmilab.ch/random/>.

[18] ZEMAN, Václav. *Přednáška z předmětu Kryptografie v informatice: Asymetrická kryptografie*. [E-learning VUT Brno] 13.4.2010 [21.5.2010]. Dostupné z WWW: <https://www.vutbr.cz/elearning/file.php/86882/MKRI_6_2010_AS.pdf>.

[19] ZEMAN, Václav. *Přednáška z předmětu Kryptografie v informatice: Generátory náhodných čísel*. [E-learning VUT Brno] 13.4.2010 [21.5.2010]. Dostupné z WWW: <https://www.vutbr.cz/elearning/file.php/86882/MKRI_7_2010_2.pdf>.

[20] ZEMAN, Václav. *Přednáška z předmětu Kryptografie v informatice: Symetrické šifry*. [E-learning VUT Brno] 13.4.2010 [21.5.2010]. Dostupné z WWW: <https://www.vutbr.cz/elearning/file.php/86882/MKRI_4_2010_sym.pdf>.

[21] ZOUHAR, Petr. *Generátor náhodných čísel*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 24 s. Vedoucí semestrální práce Ing. Jiří Sobotka.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

a	tajný klíč uživatele $\mathrm{U_A}$
b	tajný klíč uživatele $\mathrm{U_B}$
B	hodnota modré složky pixelu
C	kryptogram (zašifrovaná zpráva)
c_i	i-tý bit kryptogramu
C_j	j-tý blok kryptogramu
D	dešifrování (decryption)
DM	dolní mez rozsahu generovaných čísel
E	šifrování (encryption)
f_F	rundovní funkce
g	generátor protokolu Diffie-Hellman
G	hodnota zelené složky pixelu
GH	generátor hesla (náhodných/pseudonáhodných čísel)
h_i	i-tý bit hesla
HM	horní mez rozsahu generovaných čísel
K	klíč (v symetrických kryptosystémech)
K_D	dešifrovací klíč
K_E	šifrovací klíč
K_i	rundovní klíč i-té rundy
L_i	levá polovina vstupního bloku i-té rundy
n	modulo RSA
N	délka bloku blokové šifry
$n_\check{\mathrm{c}}$	počet zásahů do čtverce
n_k	počet zásahů do kružnice

p	prvočíslo pro výpočet RSA a Diffie-Hellman(modulo)
π'	hodnota π vypočtená metodou Monte Carlo
P_i	pravá polovina vstupního bloku i-té rundy
PRNG	Pseudo-Random Number Generator
q	prvočíslo pro výpočet RSA
r	pomocná hodnota ve výpočtu RSA
R	hodnota červené složky pixelu
r_k	poloměr kružnice
$S_{\check{c}}$	obsah čtverce
S_{k}	obsah kružnice
SK	soukromý klíč
TRNG	True Random Number Generator
$\mathrm{U_A}$	účastník komunikace
$\mathrm{U_B}$	účastník komunikace
V	vygenerované číslo námi navrženým programem
VK	veřejný klíč
X	zpráva od uživatele $\mathrm{U_A}$ Diffie-Hellman protokolu
Y	zpráva od uživatele $\mathrm{U_B}$ Diffie-Hellman protokolu
Z	zpráva v nešifrované podobě (prostý text)
z_i	i-tý bit zprávy
Z_j	j-tý blok zprávy

SEZNAM PŘÍLOH

A CD

A.1 Obsah CD

- Text.pdf - text práce ve formátu pdf
- Generátor - složka s počítačovým programem vytvořeným v rámci diplomové práce, včetně použitých fotek
- klíče.txt - sekvence vygenerované programem a využité pro testování a šifrování