Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Praktická implementace Vernamovy šifry

Bakalářská práce

Petr Kopecký

Vedoucí práce: Ing. Petr Břehovský

České Budějovice 2011

Bibliografické údaje

Kopecký, P., 2011: Praktická implementace Vernamovy šifry. [One-Time Pad implementation. Bc. Thesis, in Czech] - 33 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Abstract

The text is oriented to generation of random numbers of cryptographic quality and to performing statistical methods such as Diehard test to verify them. The data obtained this way are used in implementation of one-time pad for maximizing security of message encryption. The problems that may occur and need to be dealt with are also noted. The final part of the text includes suggestions for future one-time pad implementations and the final results that author came to.

Keywords: encryption, decryption, Vernam's cipher, One-Time Pad, random numbers

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích 12.12.2011

Poděkování

Děkuji svým rodičům za veškerou podporu, kterou mi v průběhu studia poskytli, vedoucímu této bakalářské práce, Ing. Petru Břehovskému, za pomoc, rady i připomínky a náměty, které dopomohly k sepsání tohoto textu. V neposlední řadě děkuji panu Josefu Vrbovi, za jazykovou korekturu.

Obsah

Slovníček pojmů

Šifrování – pozměnění daného textu podle určitých pravidel tak, aby nepovolaná osoba/strana nezjistila jeho původní obsah.

Dešifrování – aplikace pravidel, která převede zašifrovaný text do jeho původní podoby.

Klíč – řetězec, který je využit pro šifrování/dešifrování.

Symetrická šifra – pro šifrování/dešifrování je použit stejný klíč.

Substituční šifra – zašifrování daného textu záměnou každého znaku znakem jiným.

Vernamova šifra – neprolomitelná substituční šifra.

Náhodná čísla kryptografické kvality – náhodná čísla, která splňují „dostatečnou náhodnost“ pro efektivní aplikaci dané šifry.

Diehard Battery Test Suite – sada statistických testů k ověření kryptografické kvality náhodných čísel (dále jen „Diehard testy”).

CLI (Command Line Interface) – prostředí terminálu (Unix-like systémy)/příkazové řádky (DOS/Windows).

Perioda náhodného generátoru – délka sekvence, po které se začne výstup opakovat.

1 Úvod

Šifrování zpráv je nedílnou součástí lidské civilizace už od jejího počátku. Po vynálezu písma byl kladen velký důraz na zašifrování různých textů. Mohlo se jednat například o důležitá sdělení ve válečných konfliktech, která měla zůstat v případě jejich odchycení třetí stranou nečitelná, nebo odchycení obchodních tajemství konkurencí, či jenom o zamezení přečtení citivých údajů jednotlivce nepovolanou osobou. Šifrování se v průběhu let měnilo a mění v závislosti na dostupných technologiích a nových objevech v přírodních vědách (zvláště v matematice).

Šifrování nalezlo největší uplatnění s příchodem informačních technologií. Jelikož se informační technologie, včetně komunikací mezi nimi, staly neodlučitelnou součástí života většiny lidí, musí být kladen důraz také na jejich zabezpečení. Vernamova šifra je jedním ze způsobů, jak ochránit data tak, aby je nikdo, ať už bude mít dotyčný neomezený výpočetní výkon nebo čas, nemohl dešifrovat.

Tato práce se věnuje implementaci Vernamovy šifry pro účely šifrování elektronické korespondence včetně popisu problému generování skutečně náhodných čísel, která s implementací úzce souvisejí – tak, aby jí byl schopen aplikovat i člověk - laik.

Práce je rozdělena na tři části:

- Úvodní, která se zabývá krátkou historií šifrování a popisu, proč je použití Vernamovy šifry efektivním způsobem ochrany komunikace mezi dvěma subjekty, včetně popsání teoretického základu.
- Prostřední, věnující se vlastní realizaci implementace Vernamovy šifry, využití generátoru náhodných čísel a ověření jejich kryptografické kvality za pomoci statistických Diehard testů.
- Závěrečná pak podává souhrnné informace o implementaci, výsledcích statistických testů, možnostech jejího vylepšení a podání návrhů pro možné budoucí zpracování.

2 Cíle práce

- uvést krátkou historii šifrování a popsat princip Vernamovy šifry
- popsat 3 druhy generátorů náhodných čísel
- vygenerovat náhodná data a ověřit je pomocí Diehard testů
- analyzovat výsledky testů
- vytvořit software pro šifrování/dešifrování textu pomocí Vernamovy šifry
- navrhnout následná řešení

3 Metodika

3.1 Software

- Operační systém MS Windows XP Professional Service Pack 3 (32-bit)
- Python 2.7.2 (32-bit)
- Diehard Battery Test Suite v. DOS, Jan 7, 1997

Pro názorný příklad šifrování byl použit text vytvořený pomocí Lorem Ipsum generátoru (http://www.lipsum.com/).

4 Šifrování v komunikaci

4.1 Proč šifrovat?

Poněkud zavádějící otázka. Odpověď na ni je jednoduchá - právo na listovní tajemství, definované v Listině základních práv a svobod České republiky.

4.2 Historie šifrování v kostce

Šifrování provází lidstvo už od počátku, kdy k němu docházelo zprvu ústní formou a rozmachu se mu dostalo až s vynálezem písma. Dochované důkazy o šifrování textů pocházejí už ze starověkého Egypta či Izraele – a to ve formě nápisů a příkladů na kamenných deskách nebo papyrech.

Ve starověké Evropě pocházejí důkazy o šifrování ze Sparty, kdy bylo použito přístroje zvaného „Scytale“ pro šifrování korespondence mezi vojevůdci. Princip Scytale spočíval v namotání kusu pergamenu na šifrovací dřevěnou hůl a následném napsání zprávy. Po rozmotání pergamenu se jevila zpráva jako změť písmen. Pro dešifrovací postup byl stejný jako šifrovací - byla použita druhá dřevěná hůl o stejném průměru jako ta první, na niž se namotal pergamen se zašifrovanou zprávou. Jednalo se o první doložené použití transpoziční šifry.

Řecký spisovatel Polybius vynalezl systém pro šifrování, který je dnes znám jako tzv. Polybiova šachovnice. Ta měla očíslované sloupce a řádky, do kterých byla vsazena písmena. Zašifrovaná zpráva měla podobu sledu po sobě jdoucích čísel.

Římané dali světu Cézarovu šifru – jeden z nejznámějších způsobů šifrování. Princip byl jednoduchý a na svou dobu efektivní. Jednalo se o zašifrování textu posunutím každého písmene o pevně daný počet míst v abecedě. Polybiova šachovnice a Cézarova šifra se řadí do kategorie tzv. substitučních šifer.

Ve středověku se používaly hlavně šifrovací postupy vytvořené v Itálii. Byly to šifrovací slovníky, tzv. „Nomenklátory”, které obsahovaly sadu klíčů, zástupné symboly pro písmena a několik dvoupísmenných ekvivalentů pro slova a jména.

Ve světových válkách byl asi nejznámějším šifrovacím strojem ENIGMA německého původu. Stroj pracoval na elektro-mechanickém principu s několika rotory s písmeny. Existoval v ně-

kolika provedeních, z nichž nejznámějšími jsou tří-rotorová verze, používaná Wehrmachtem a čtyř-rotorová, kterou používala Kriegsmarine.

Poválečná kryptografie zahrnuje postupy a šifrovací algoritmy, jako jsou například symetrický DES (Data Encryption Standard), který má 64-bitovou velikost bloku a používá 54-bitový klíč během šifrování nebo RSA (Rivest-Shamir-Adleman), pracující s modulem součinu dvou velkých prvočísel. V neposlední řadě určitě stojí za zmínku problematika asymetrické kryptografie – existují dva klíče (veřejný a tajný). Odesílatel zašifruje zprávu veřejným klíčem příjemce a odešle ji. Dešifrování zprávy je možné pouze tajným klíčem příjemce, takže pokud nedojde k jeho kompromitaci, může si zprávu přečíst pouze ten, jemuž je určena.

Nyní, po krátkém seznámení čtenáře s historií šifrování, se nabízejí otázky: Jaké jsou nevýhody šifrování? Má tedy vůbec smysl šifrovat?

4.3 Nevýhody

Pro běžného člověka, šifrujícího svoji korespondenci, je určitě největší nevýhodou jeho neznalost principu současných způsobů šifrování. Dnes je jedním z nejpoužívanějších PGP (Pretty Good Privacy), konkrétně jeho deriváty OpenPGP nebo GPG (GNU Privacy Guard). Pro šifrování obyčejné korespondence postačuje dostatečně. Chce-li člověk zaručit skutečně neprolomitelnou ochranu pro svou korespondenci, musí sáhnout k jednodušším, a zároveň obtížněji realizovatelným formám ochrany. Takovou ochranu skýtá jednoduchá Vernamova šifra – pod pojmem „jednoduchá“ se rozumí, jednoduchý postup šifrování (podobnost s Cézarovou šifrou, viz. Historie v kostce). Nevýhody plynou z obtížných přípravných postupů.

4.4 Má tedy vůbec smysl šifrovat?

Pro člověka píšícího obyčejné texty bez nějakého zásadnějšího významu určitě nemusí mít šifrování smysl. Dalším extrémem jsou různé vládní organizace, orgány veřejné správy nebo armáda. Pro ně je pro šifrování komunikace nutností. Lidé, pracující v oblasti počítačové bezpečnosti, ať už se jedná o „white hat“, „grey hat“ nebo „black hat“, musí nebo by aspoň měli mít zájem na šifrování své komunikace. Ochrana citlivých dat, ať už svých nebo někoho jiného, by tedy měla být jejich prvořadým zájmem.

5 Vernamova šifra

Nyní se dostáváme k popisu a teorii šifry, jejíž implementací se práce zabývá. Jak už bylo řečeno v předchozí sekci, je její princip podobný Cézarově šifře. Jedná se tedy o symetrickou substituční šifru. Proč se jedná o tzv. dokonalou šifru? Je to jediný způsob, o kterém je známo, že dokáže vytvořit matematicky neprolomitelné šifrování, pokud jsou splněna jistá pravidla:

1. Klíč použitý k šifrování je stejně dlouhý jako text, který chceme zašifrovat.
2. Klíč musí být vygenerován skutečně náhodně.
3. Klíč k šifrování se musí použít zásadně pouze jednou.

5.1 Historie

V roce 1917 vyvinul technik AT&T, Gilbert S. Vernam, způsob šifrování pro TTY (Teletypewriter), který matematicky napodobuje model Franka Millera z roku 1882. Sestrojil elektromechanický stroj, který pracoval se dvěma páskami. První páska obsahovala text v pětibitové mezinárodní telegrafní abecedě. Druhá pak sekvenci náhodných pětibitových hodnot o stejné délce, jako páska první. Zkombinování proběhlo přes relé pomocí booleovské funkce XOR (Exclusive OR). Tato metoda ale nebyla příliš bezpečná. Kapitán Joseph Mauborgne ukázal, že metoda nedokáže odolávat kryptoanalýze, a proto usoudil, že pro naprostou bezpečnost šifrování musí být splněna určitá pravidla (viz výše). V roce 1919 dostal Vernam na tuto metodu šifrování, zvanou OTT (One-Time Tape), patent.

Na začátku dvacátých let 20. století dešifrovali tři němečtí kryoptoanalytici francouzskou diplomatickou korespondenci. Ta využívala vytištěné číselné kódy z šifrovacího slovníku, aby převedla slova a výrazy na čísla. Na tomto základě vytvořili systém absolutní ochrany – na papírové archy byly naneseny řetězce náhodných čísel. Vždy přitom existovali pouze dvě kopie těchto archů – jedna pro odesílatele, druhá pro příjemce. Postupem času se začaly používat malé bločky s náhodnými čísly nebo písmeny, které byly zapsány na stránkách v blocích po pěti. Každá stránka byla použita pro zašifrování zprávy pouze jednou. Tyto bločky jsou označovány též jako OTP.

5.2 Princip

Osobně si myslím, že největším problémem přípravy využití Vernamovy šifry je získání stejně dlouhého řetězce (klíče) náhodných čísel (nebo znaků), jako je délka textu, který chceme zašifrovat. Problematikou náhodných čísel se bude věnovat následující sekce.

Vlastní aplikace Vernamovy šifry je vcelku jednoduchá, ačkoliv je několik cest, jak se k danému výsledku dopracovat. Pro zvýšení bezpečnosti se zašifrovaný text rozdělí do bloků po (nejčastěji) 5ti číslicích. V několika následujících podsekcích je nastíněno několik způsobů, jak šifrovat pomocí Vernamovy šifry.

5.2.1 Převod na čísla

Jednou z metod je převod jednotlivých písmen textu na číslice. Na vstupu se použijí čísla převedeného textu a náhodně vygenerovaná čísla. Při šifrování se použije operace sčítání. Výsledným výstupem je text zašifrovaný Vernamovou šifrou. Tato metoda skýtá jistá úskalí – pro převod textu na číslice se musí nejdříve definovat abeceda, tzn. každému použitému znaku musí být přiřazena jeho číselná hodnota. Pro názornou ukázku jsem vytvořil abecedu znaků bez diakritických znamének. Druhá řádka obsahuje písmena, která se nejčastěji vyskytují v české abecedě. Čísla v prvním sloupci mohou být zvolena náhodně.

-	0	1	2	3	4	5	6	7	8	9
0	e	o	a	i	n	t	v	s	-	-
3	b	c	d	f	g	h	j	k	l	m
9	p	q	r	u	w	x	y	z	-	-

Tabulka 1: Definice abecedy

Následuje konverze textu na čísla podle abecedy. Výsledkem jsou dvoumístná čísla v pořadí sloupec-řádek.

Text	V	E	R	N	A	M	O	V	A	S	I	F	R	A
Konverze	06	00	92	04	02	39	01	06	02	07	03	33	92	02

Tabulka 2: Konverze

Před posledním krokem se rozčlení převedený text na bloky po 5ti číslicích a doplnění posledního bloku libovolnými čísly (zde použity číslice 9). Následně se přidá klíč a na každou dvojici čísel (převedený text, klíč) se aplikuje operace sčítání tak, aby výsledné číslo bylo vždy jednociferné, například 6 + 9 = 5.

Převedený text	06009	20402	39010	60207	03339	20299
Klíč (+)	82002	81587	07826	90099	52513	78972
Zašifrovaný text	88001	01989	36836	50296	55842	98161

Tabulka 3: Zašifrování textu

Pro dešifrování Vernamovy šifry se využije operace odečítání. Například se může použít odečítání (zašifrovaný text, zřevedený text) pro získání klíče, nebo (zašifrovaný text, klíč) pro získání původního textu.

Zašifrovaný text	88001	01989	36836	50296	55842	98161
Klíč (-)	82002	81587	07826	90099	52513	78972
Dešifrovaný text	06009	20402	39010	60207	03339	20299

Tabulka 4: Dešifrování textu

Posledním krokem po dešifrování je převedení znaků podle dané abecedy. Může se ještě využít alternativní přístup – pro šifrování použijeme operaci odečítání a pro dešifrování operaci sčítání.

5.2.2 Využití náhodných znaků

Tento způsob je další variantou šifrování textu. Při něm se pomocí znaků se využívá, jako jeden způsob, tzv. Vigenèrova tabulka nebo také Vigenèrův čtverec (viz. tabulka 5). Následující postup šifrování je podobný předchozímu.

-	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
c	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
d	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
e	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
f	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
g	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
h	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
i	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
j	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
k	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
l	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
m	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
n	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
o	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
p	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
r	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
s	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
t	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
u	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
v	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
w	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
x	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Tabulka 5: Vigenèrova tabulka

Pro získání zašifrovaného znaku se vezme aktuální znak v textu na vodorovné ose a znak hesla na ose svislé. Průsečík udává znak zašifrovaný.

Text	V	E	R	N	A	M	O	V	A	S	I	F	R	A
Klíč	N	O	A	X	R	X	A	E	K	P	W	B	U	G
Zašifrovaný text	I	S	R	K	R	J	O	Z	K	H	E	G	L	G

Tabulka 6: Zašifrování textu

Dešifrování textu probíhá následujícím způsobem – pro každý znak klíče se na vodorovné ose najde znak takový, který odpovídá zašifrovanému textu. Udělá se průsečík se svislou osou a na prvním řádku se objeví znak dešifrovaný.

Klíč	N	O	A	X	R	X	A	E	K	P	W	B	U	G
Zašifrovaný text	I	S	R	K	R	J	O	Z	K	H	E	G	L	G
Dešifrovaný text	V	E	R	N	A	M	O	V	A	S	I	F	R	A

Tabulka 7: Dešifrování textu

5.2.3 ASCII bitový převod

Poslední způsob, který zde budu popisovat, také využívá náhodných znaků. Oproti předchozímu příkladu se využívá ASCII tabulka jako abeceda. Každý znak textu i klíče se pomocí ASCII tabulky převede do bitové podoby.

Text	S	I	F	R	A
Binární podoba	01010011	01001001	01000110	01010010	01000001
Klíč	L	D	V	A	C
Binární podoba	01001100	01000100	01010110	01000001	01000011

Tabulka 8: Převod ASCII znaků do binární podoby

Následně se využije matematická funkce XOR, která má na vstupech bitové podoby textu a klíče. Výstupem je ASCII text zašifrovaný Vernamovou šifrou.

Text	01010011	01001001	01000110	01010010	01000001
Klíč	01001100	01000100	01010110	01000001	01000011
XOR výstup	00011111	00001101	00010000	00010011	00000010

Tabulka 9: Zašifrování textu pomocí funkce XOR

Pro dešifrování XOR výstupu se opět využije klíč použitý pro šifrování. I zde platí, že každý takový musí být náhodný a musí být použit pouze jednou, resp. dvakrát – poprvé na zašifrování textu, podruhé na dešifrování textu.

Zašifrovaný text	00011111	00001101	00010000	00010011	00000010
Klíč	01001100	01000100	01010110	01000001	01000011
Dešifrovaný text	01010011	01001001	01000110	01010010	01000001
ASCII převod	S	I	F	R	A

Tabulka 10: Dešifrování textu pomocí funkce XOR

S využíváním Vernamovy šifry se objevuje docela podstatný problém. Když se 2 strany rozhodnou využívat tohoto způsobu šifrování, musí také myslet na distribuční kanály pro své klíče. Je možné distribuovat klíč pomocí internetu – například přes zašifrovaný e-mail nebo přes nějaký zabezpečený protokol. Je však nutné podotknout, že neprolomitelnost takto distribuovaného klíče závisí na síle šifry použité k jeho zašifrovaní. Nejspolehlivějším způsobem distribuce je osobní předání, například na CD nebo DVD nosiči, což však není dobře realizovatelné, pokud se například obě strany nalézají na opačných stranách Země.

6 Náhodná čísla

Na otázku: „Co jsou náhodná čísla?“, může být vícero odpovědí. Nejdříve je nutné si uvědomit: „Co je to náhodnost?“, nebo možná ještě lépe: „Co znamená vlastnost být náhodný?“.

Je číslo 3 náhodné? Pokud je dané číslo vygenerováno generátorem skutečně náhodných čísel, pak odpověď je: „Ano, číslo 3 je náhodné“. Nás však bude více, než zda daný generátor umí vygenerovat náhodná čísla, zajímat, jestli vygenerovaná sekvence na sobě nezávislých náhodných čísel je skutečně náhodná, tzn. pokud každá jednotlivá číslice nemá nic společného s žádnou další číslicí v sekvenci, a že je pevně daná pravděpodobnost jejich výskytu.

V dřívějších dobách, když chtěl někdo získat náhodná čísla, musel se uchýlit k jednoduchým metodám. Asi nejlepším příkladem manuálního generování náhodných čísel v rozmezí 1-6 byl hod kostkou (za předpokladu, že kostka byla perfektně vyvážená na všech stranách). Jako druhý příklad může posloužit losování karet s čísly z urny. I když jsou tyto způsoby relativně efektivní, jsou značně náročné z pohledu času. Každý si asi umí představit, kdyby měl pomocí kostky vegenerovat náhodnou sekvenci čísel o délce 500 000 číslic. S příchodem moderních technologií se generování náhodných čísel značně zjednodušilo. Příkladem může být počítač Ferranti Mark I z roku 1951, který vkládal 20 náhodných bitů do zásobníku za pomoci generátoru šumu.

John von Neumann se v roce 1946 pokusil najít řešení za pomoci jednoduchých aritmetických operací, které mohly počítače zpracovat. Princip spočíval ve výpočtu kvadrátu předchozího náhodného čísla a vybrání několika prostředních číslic jako dalšího náhodného čísla. Problém této metody je, že takto vygenerovaná čísla nejsou náhodná – každý další potomek je de facto závislý na svém rodiči. Takto (deterministicky) vygenerovaným číslům se říká *pseudo-náhodná*. Co z toho vyplývá? Podle testů z padesátých let 20. století se různá čísla na vstupech po určitém čase dopracují ke stejným výstupům. V tomto případě záleží na velikosti čísel.

Detailnější popis náhodných čísel je nad rámec této práce. Pro podrobnější studium problematiky náhodných čísel doporučuji knihu od pana profesora Donalda E. Knutha - The Art Of Computer Programming. Je jedním z mých cílů porovnat několik generátorů a z nich využít ten nejvhodnější pro potřeby implementace Vernamovy šifry.

6.1 Python random module

Modul „random" v jazyce Python obsahuje implementace pseudo-náhodných generátorů, z nichž jádro tvoří Mersenne Twister generátor. Tento generátor vytvořili a popsali v roce 1998 Makoto Matsumoto a Takuji Nishimura. Podle autorů má jeho algoritmus astronomickou periodu $2^{19937} - 1$, což je velice dobrý výsledek. Pro podrobnější studium Mersenne Twister generátoru doporučuji publikaci „Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator" od výše zmíněných autorů.

Pro testování byly pomocí modulu „random" vygenerovány 3 miliony pseudonáhodných čísel (cca 3 miliony je potřeba pro Diehard testy). Tato čísla (32-bitové integery) musela být upravena do určitého formátu. Ten musí ctít následující formuli: čísla se převedou do hexadecimální podoby a uloží se do souboru – tak, že každá řádka obsahuje 80 znaků; ve výsledku tak připadá 10 32-bitových integerů na řádku. Formule byla použita pro vytvoření souboru „data.raw".

Datový soubor, obsahující hexovou podobu vygenerovaných čísel, byl následně použit jako vstup programu „ASC2BIN.EXE", který je součástí Diehard Battery Test Suite balíku pro systémy MS Windows. Tento program převede datový soubor do binární podoby, která je nutná pro spuštění Diehard testů.

6.2 KISS generátor

KISS (Keep It Simple Stupid) je kombinačním generátorem náhodných čísel, jehož autorem je George Marsaglia (mimo jiné autor Diehard Battery Test Suite – viz následující sekce). Jedná se o kombinaci několika jednoduchých a rychlých pseudo-náhodných generátorů, které dohromady vytvoří generátor s dostatečně dlouhou periodou a schopností projít Diehard testy. Jedná se o následující generátory:

- Kongruenční generátor
- Generátor využívající posuvný registr
- MWC (Multiply-With-Carry) generátor

Dle autora je KISS generátor velice vhodný pro svou rychlost a jednoduchost (pro generátory s tak vysokou periodou). Součástí Diehard Battery Test Suite pro systémy MS Windows je program „MAKEWHAT.EXE", který dokáže vytvořit binární soubory pro Diehard testy, podle druhu zadaného generátoru. Pro inicializaci počátečních hodnot musí uživatel zadat 4 počáteční (nenulová) čísla typu integer. Výstupem tohoto procesu je binární soubor „kiss.32". Zadal jsem tyto počáteční hodnoty: **58964**, **21**, **789** a **244568**.

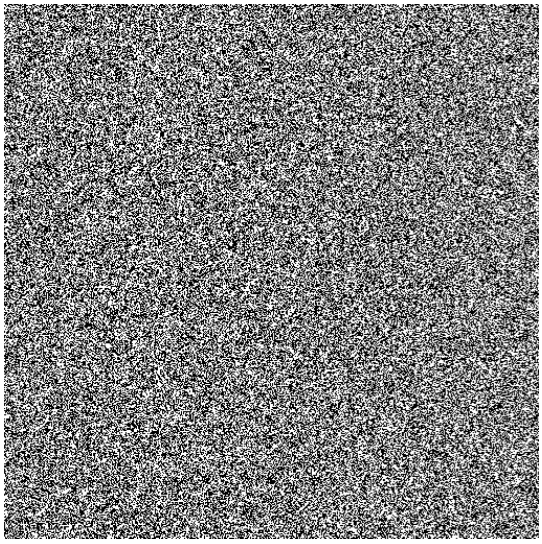
6.3 Atmosférický šum

Při hledání zdrojů pro tuto práci jsem nalezl internetovou stránku „www.random.org", poskytující binární soubory náhodných čísel, vygenerovaných za pomoci propojení počítače a rádia, které snímá atmosférický šum. Jelikož fyzikální jevy jsou považovány jako jediný zdroj „skutečně" náhodných čísel, je tento způsob ze tří uvedených jedním z největších aspirantů na podání skvělých výstupů u Diehard testů a následném využití při implementaci Vernamovy šifry.

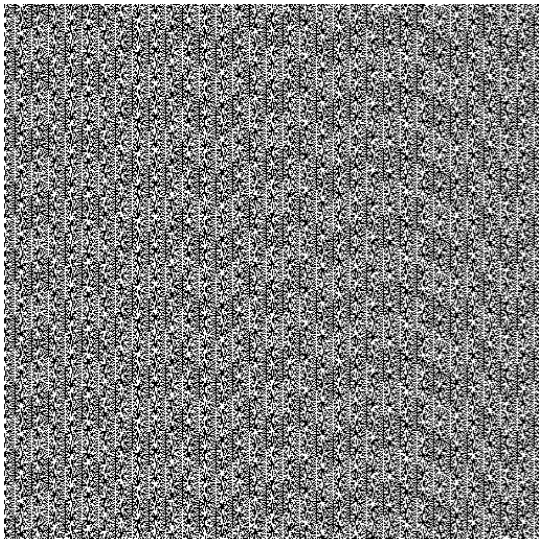
Služba „www.random.org" nabízí k binární soubory o velikosti cca 1.0 MB, které poskytuje od roku 2006 k volnému použití. Tyto soubory jsou vytvářeny každý den vždy po půlnoci. Pro mé potřeby bylo nutné stáhnout 11 binárních souborů (prvních 11 dní v listopadu 2011), které se po uložení na disk spojily do jednoho cca 11.0 MB velkého souboru za pomocí CLI příkazu:

copy /B *.bin data.bin

Pro srovnání jsou přiloženy obrázky, které byly vytvořeny z náhodných čísel generovaných za pomoci atmosférického šumu a PHP funkce rand(). Jak je vidět na obrázku č. 2, je perioda opakování počítačem generované sekvence pseudo-náhodných čísel častější než u sekvence vygenerované zachycením atmosférického šumu. Tím pádem je podstatně méně kvalitní a pro potřeby šifrování nepoužitelná.



Obrázek 1: Náhodnost atmosférického šumu



Obrázek 2: Náhodnost PHP funkce rand()

6.4 Alternativy

Existuje mnohem více kvalitních generátorů, využívajících fyzikální jevy, které by dokázaly generovat dostatečně rychle sekvenci čísel pro praktické účely v kryptografii. Může se jednat

například o Geiger-Müllerův čítač, měřící ionizační záření v plynném prostředí svého vnitřku (nejčastěji se jedná o vodík nebo argon), nebo hardwarové generátory.

Z hardwarových generátorů stojí za zmínku využití audio/video vstupu. Pokud není připojen do zvukové karty mikrofon, může se na unixových systémech získat šum jako náhodná sekvence bitů – pomocí čtení /dev/audio. Tyto bity se mohou ještě zkompresovat pomocí unixového nástroje „compress".

7 Diehard testy

Diehard testy poskytl v roce 1995 George Marsaglia k volnému užití pro veřejnost. Jedná se o soubor statistických testů, které testují určité vlastnosti vygenerových čísel, převedených do binární podoby. Výstupem těchto testů jsou tzv. p-hodnoty. Pro pochopení p-hodnoty musíme definovat několik pojmů.

Nulová hypotéza (H0) – je statistická hypotéza, kterou se ve své podstatě snažíme vyvrátit, abysme mohli něco prokázat.

Chyba 1. druhu – když dojde k zamítnutí platné hypotézy H0.

Hladina testu – α je maximální přípustná pravděpodobnost chyby 1. druhu.

P-hodnota – je taková nejmenší α, při které ještě dochází k zamítání H0 z dané množny dat... neboli pokud platí H0, je p-hodnota pravděpodobností výsledků stejných nebo méně příznivých pro dané H0.

$$\alpha = P(T \varepsilon W \mid H_0)$$

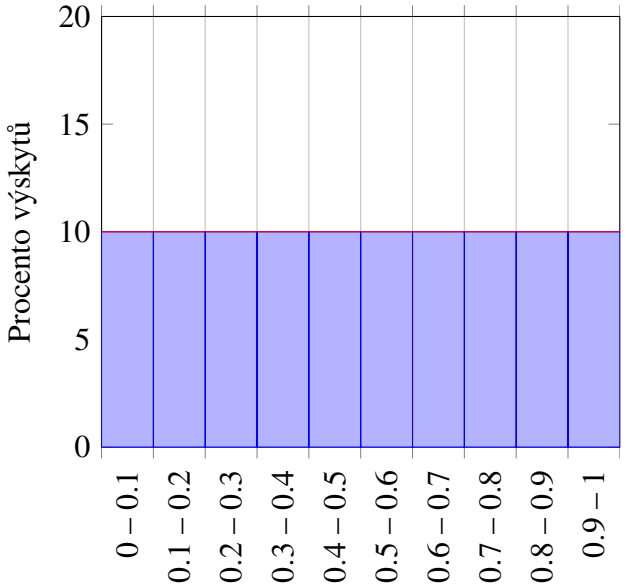
Písmeno T ve vzorci je testovací kriterium a W kritickým oborem.

Ve výsledku, pokud daná sekvence čísel prošla úspěšně Diehard testy, by měly být všechny p-hodnoty rovnoměrně rozděleny v intervalu [0,1). Může také dojít k tomu, že p-hodnoty spadnou do extrémních hodnot jako jsou 0 a 1. Pokud k takovým případům dojde na 6ti a více místech, daná sekvence bitů neprošla úspěšně danými testy. Pro nejpřesnější výsledky bylo použito všech 15 testů:

- Birthday Spacings
- Overlapping Permutations
- Ranks of 31x31 and 32x32 Matrices
- Ranks of 6x8 Matrices
- Monkey Tests on 20-bit Words
- Monkey Tests OPSO, OQSO, DNA

- Count the 1's in a Stream of Bytes
- Count the 1's in Specific Bytes
- Parking Lot Test
- Minimum Distance Test
- Random Sphere Test
- The Squeeze Test
- Overlapping Sums Test
- Runs Test
- The Craps Test

Pokud jde o skutečně náhodnou sekvenci, jsou ve výsledku všechny p-hodnoty vyrovnané. Pro ilustraci zde uvedu graf takového stavu.



Obrázek 3: Graf ideálního rozvržení p-hodnot

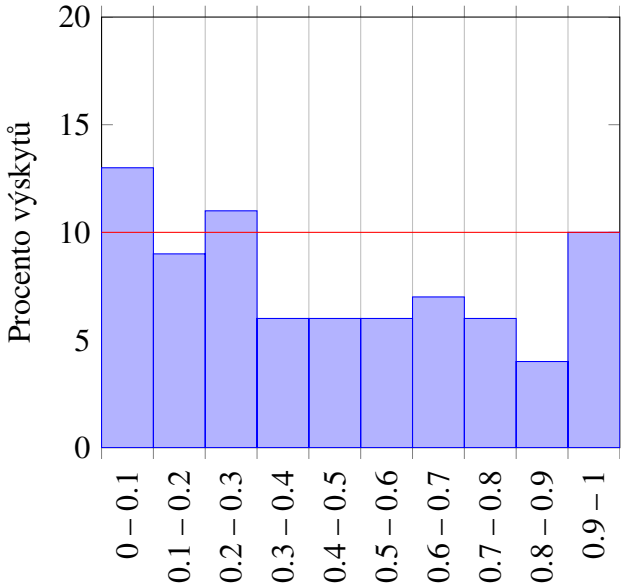
Data získaná ze 3 generátorů – dva z nich byly pseudo-náhodné (Python random modul, KISS) a jeden „skutečně" náhodný, který využíval atmosférického šumu – jsou otestovaná podle výše uvedených testů.

Pro testy všech uvedených generátorů bylo použito následujícího postupu:

Nejprve byl spuštěn předkompilovaný soubor „DIEHARD.EXE". Po spuštění byl zadán název testovaného vstupního binárního souboru. Následně byl zadán název výstupního souboru, který obsahuje dump výpisu výsledků (všechny testované soubory mají podobu: „název_generátoru.out". Z tohoto výstupního souboru byly odfiltrovány pomocí programu „DIEHARDCSV.EXE" (viz. Reference) p-hodnoty a zapsány do souboru „název_generátoru.res". Kvůli možnému zaokrouhlení mohou procentuální výsledky dávat dohromady více než 100 %.

7.1 Python random modul test

Při testování jsem zjistil, že test č. 12 (The Squeeze Test) nelze aplikovat, i když velikost testovaného souboru vyhovuje potřebné velikosti pro Diehard testy. Při spuštění testu dojde k vyhození chyby „run-time error F6501 - end of file encountered". Po podrobnějším studiu problému mělo být řešení v podobě vlastního zkompilování Diehard testů. Nicméně ani tato možnost nebyla úspěšná. K vyvození závěrů bylo tedy použito 14 ostatních testů.



Obrázek 4: Graf výsledných p-hodnot random modulu

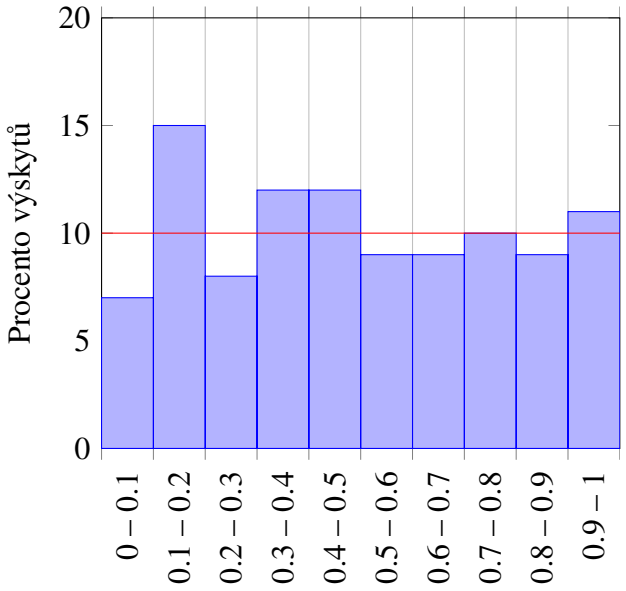
Kromě hodnot znázorněných Obrázkem 4 byl zjištěn následující počet p-hodnot v extrémech 0 a 1:

- 0 – 5 %
- 1 – 16 %

Jak je vidět, tento typ generátoru není vůbec vhodný pro potřeby šifrování.

7.2 KISS test

KISS generátor má, díky své povaze, předpoklady na lepší výsledky Diehard testů. Obrázek 5 znázorňuje naměřené p-hodnoty.

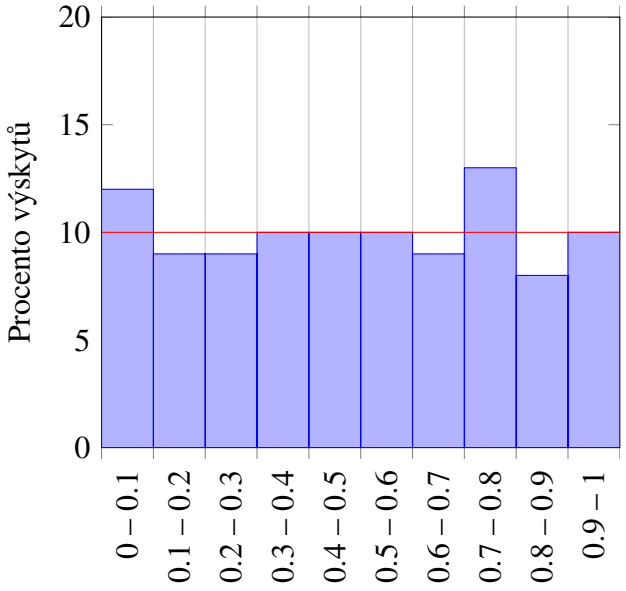


Obrázek 5: Graf výsledných p-hodnot KISS generátoru

KISS generátor si tedy vede ve výsledku podstatně lépe než generátor z modulu „random" programovacího jazyka Python.

7.3 Test atmosférického šumu

Tento druh generátoru má ze všech uvedených předpoklady na nejlepší výsledky Diehard testů. Zkombinovaný soubor „at_noise.bin" obsahuje náhodná binární data, která mají nejvíce p-hodnot ve vyhovujícím kritériu 10 %.



Obrázek 6: Graf výsledných p-hodnot atmosférického šumu

Jak je na grafech výsledných p-hodnot vidět, má test atmosférického šumu nejlepší výsledky, což je shodné s předpoklady. Z grafů lze též vyčíst, že získat opravdu skutečně náhodná čísla je velice obtížný úkol.

V následující sekci bude využit binární soubor „at_noise.bin” v praxi při šifrování pomocí Vernamovy šifry. Tento soubor je pro přehlednost přejmenován na „key.bin”.

8 Implementace

8.1 Popis

Původní myšlenkou bylo, aby vlastní implementace byla jednoduchá a napsaná v jazyce Python, který je rozšířený a multiplatformní. Implementace je zajištěna jednoduchým skriptem „otp.py".

Skript se skládá z několika funkcí. Mezi nejdůležitější části patří metoda:

process(plaintext, key)

Jak je ze signatury funkce vidět, jsou zde 2 parametry, z nichž jsou oba povinné. Parametr „plaintext" je text (řetězec), který budeme chtít šifrovat/dešifrovat, a „key" je klíč, který využijeme pro šifrování/dešifrování. K vlastnímu šifrování/dešifrování ASCII znaků na stejné pozici v řetězci „plaintext" a „key", je využita logická funkce XOR .

Je nutné definovat několik funkcí, které budou sloužit k ověření:

check_length(plaintext, key)

Tato funkce má jednoduchý úkol – vezme délky řetězců „plaintext" a „key" a porovná je. Délka se musí rovnat; je to jeden ze základních předpokladů Vernamovy šifry.

check_ascii(plaintext)

Jelikož je program navržen tak, že text, který chceme zašifrovat, musí obsahovat pouze ASCII znaky, je nutné, aby také obsahoval funkci, která to ověří. Výše uvedená funkce využívá regulárních výrazů pro kontrolu řetězce „plaintext", splňuje-li dané kritérium.

Poslední funkcí, kterou skript obsahuje, je:

key_strip(k_file, length)

Funkce zajišťuje poslední podmínku, kterou musí obsahovat Vernamova šifra – každý klíč je použit pouze jedenkrát. Funkce otevře soubor, jehož název je definovaný parametrem „k_file" a odstraní ze začátku přesně daný počet bytů. Tento počet bytů je definován parametrem „length", který je roven délce textu, který chceme zašifrovat/dešifrovat. V těle zdrojového kódu se funkce spustí pouze za předpokladu, že proběhne úspěšně funkce ***check_length()***.

8.2 Spuštění

Pro spuštění programu je potřeba mít nainstalovaný Python 2.x.x. a mít nastavenou systémovou proměnnou „PATH", která obsahuje cestu do instalačního adresáře. Po spuštění CLI se zadá příkaz:

python otp.py -h

pro zobrazení nápovědy ohledně způsobu použití programu. Výpis tohoto příkazu je následující:

Program tedy bere 2 povinné parametry – plaintext_file (soubor, který chceme zašifrovat/dešifrovat) a key_file (soubor obsahující klíč k zašifrovaní/dešifrování).

Nepovinný parametr „-d" určuje, že daný soubor chceme dešifrovat. Standardně je skript nastaven k šifrování souboru.

Posledním nepovinným parametrem „-e" se zajišťuje, že dojde ze souboru obsahující klíč k odmazání počtu bytů odpovídající velikosti plaintext_file souboru.

Pro demonstraci fungování skriptu je využit následující text o délce 500 bytů:

> „Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque id eros turpis, pulvinar pharetra orci. Fusce consectetur euismod ullamcorper. Cras tincidunt justo eu erat suscipit laoreet. Sed eget turpis orci, a vehicula lorem. In hac habitasse platea dictumst. Duis porttitor laoreet enim, malesuada malesuada massa rhoncus ut. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla vitae neque eu eros tincidunt vulputate congue nec arcu. Aenean eget nullam."

Tento text je uložen do souboru „input.txt". Jako klíče je využito binárního souboru „key.bin" (viz. předchozí sekce). Tyto soubory jsou uloženy ve stejném adresáři, jako skript „otp.py". Pomocí příkazu:

python otp.py input.txt key.bin

je vytvořen zašifrovaný soubor „encryted.txt". Pro opětovné dešifrování souboru je použit následující příkaz:

python otp.py -d encrypted.txt key.bin

Ten vytvoří soubor „decrypted.txt” s dešifrovaným textem.

Jak je zde ilustrováno, je použití skriptu k implementaci Vernamovy šifry jednoduché. Byl tedy splněn můj cíl, aby implementace byla jednoduchá a splnila účel.

9 Návrhy

Pro budoucí řešení této problematiky stojí za zmínku vytvoření vlastního generátoru náhodných čísel. Například sestrojení Geiger-Müllerova čítače a vlastního generátoru náhodných čísel za pomoci atmosférického šumu s lepší kalibrací a následně jejich porovnáním by mohlo vnést více světla do dané problematiky generování kvalitních náhodných čísel mimo akademické prostředí.

Dalším možným řešením je přepsání zde popsaného skriptu na šifrování/dešifrování textu Vernamovou šifrou do podoby Python modulu, který by se dal využívat při různých kryptografických řešeních. Ovšem na vstupu by se musel zavést takový zdroj náhodných čísel, který by prošel Diehard testy.

10 Závěr

Závěrem stačí říci, že tato práce splnila všechny dané cíle. Byla popsána krátká historie šifrování s vysvětlením, proč je Vernamova šifra – jak co do jednoduchosti implementace, tak do obtížnosti celkové realizace – ideálním řešením pro šifrování (nejen) textových zpráv. Následně byl vysvětlen princip fungování Vernamovy šifry a podán výčet přístupů, jakými ji lze dosáhnout.

Také byla nastíněna problematika generování náhodných čísel s praktickými ukázkami využití 3 generátorů , včetně jejich popisu. Pomocí těchto generátorů byl vytvořen soubor, který byl následně využit pro testování pomocí Diehard testů. Bylo vysvětleno, co jsou Diehard testy, co je jejich výstupem a bylo řečeno, jaké hodnoty by měla splňovat ideální náhodnost. Testovaná data byla porovnána a ta s nejlepšími výsledky byl využita jako klíč v implementaci.

Implementace Vernamovy šifry byla zhotovena jako skript pomocí programovacího jazyka Python.

Poslední část tvoří úvaha nad možným budoucím řešením dané problematiky.

Reference

[1] RIJMENANTS, Dirk. Dirk Rijmenants' Cipher Machines and Cryptology : Historical and Technical Information about Crypto Machines, Cryptology and Free Software Simulations [online]. c2004, last updated 26 Jul 11 [cit. 2011-09-11]. One-time-pad. Dostupné z WWW: <http://users.telenet.be/d.rijmenants/en/onetimepad.htm>.

[2] VONDRUŠKA, Pavel. Cesta kryptografie do nového tisíciletí: Od Kámasutry k osobním zápiskum K. H. Máchy. ComputerWorld [online]. 2000, 37/2000–40/2000, [cit. 2011-09-11]. Dostupný z WWW: <http://www.math.muni.cz/~bulik/vyuka/aplikace/vondruska-cesta.pdf>.

[3] WAGNER, Neal. R. Perfect Cryptography: The One-Time Pad [online]. c2001, Revision date: 2001-09-27. [cit. 2011-09-11]. The Laws of Cryptography:. Dostupné z WWW: <http://www.cs.utsa.edu/~wagner/laws/pad.html>.

[4] BROWN, Robert G. Duke Physics [online]. Version 3.31.0. c2004 [cit. 2011-09-14]. Robert G. Brown's General Tools Page. Dostupné z WWW: <http://www.phy.duke.edu/~rgb/General/dieharder.php>.

[5] Počet-znaků.cz [online]. c2010 [cit. 2011-09-14]. Četnost znaků. Dostupné z WWW: <http://www.pocet-znaku.cz/cetnost-znaku>.

[6] KNUTH, Donald E. The Art Of Computer Programming : Volume 2 / Seminumerical Algorithms. Third Edition. [s.l.] : Addison-Wesley Professional, 1997. 784 s. ISBN 0201896842.

[7] EASTLAKE 3RD, Donald E.; SCHILLER, Jeffrey I.; CROCKER, Steve. IETF Tools [online]. Request for Comments: 4086. Červen 2005 [cit. 2011-11-21]. Randomness Requirements for Security. Dostupné z WWW: <http://tools.ietf.org/html/rfc4086>.

[8] Tonbandstimmen [online]. [cit. 2011-12-11]. A Random Bit Generator for EVPmaker. Dostupné z WWW: <http://www.tonbandstimmen.de/evpmaker/random-bit-generator/index_e.htm

[9] Stata [online]. 1999 [cit. 2011-12-11]. Certification results. Dostupné z WWW: <http://www.stata.com/support/cert/diehard/index.html>.

[10] Random.org [online]. c2011 [cit. 2011-12-11]. Statistical Analysis. Dostupné z WWW: <http://www.random.org/analysis/>.

[11] Wikipedia [online]. 15. 9. 2011 [cit. 2011-12-12]. Testování statistických hypotéz. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Testování_statistických_hypotéz>.