Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Návrh a implementace generátoru náhodných čísel

Diplomová práce

Bc. Stanislav Pecka

Vedoucí práce: Ing. Petr Břehovský

České Budějovice 2018

Bibliografické údaje

Pecka, S. 2018: Návrh a implementace generátoru náhodných čísel. [Design and implementation of random number generator. Mgr. Thesis, in Czech.] 57 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Annotation

This diploma thesis deals with creation of several random number generators. The data from these prototypes are then compared according to various aspects and statistical methods.

The reader is familiar with the basic concepts, the existing random number generators and the technologies used.

Keywords

Random number comparison, TRNG, true random number generator

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL MAGISTERSKÉ PRÁCE

Student: Stanislav Pecka, Bc.
(jméno, příjmení, tituly)

Obor – zaměření studia: Aplikovaná informatika

Katedra/ústav, kde bude práce vypracovávána: Ústav aplikované informatiky

Školitel: Petr Břehovský, Ing.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF:
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant:
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma magisterské práce: Návrh a implementace generátoru náhodných čísel

Cíle práce:

Cílem práce je návrh a implementace několika druhů HW generátorů skutečně náhodných čísel a jejich porovnání. Porovnávacími kritérii jsou stanoveny míra náhodnosti generovaných dat, rychlost generování, náročnost implementace a masovost rozšíření. Práce bude zaměřena především na možnost využití moderních technologií, jako jsou smartphony, tablety apod.

Základní doporučená literatura:

Dice for statistical experiments. **Galton, Francis. 1890.** 13-14, místo neznámé : Nature, 1890. 10.1038/042013a0.

Gentle, James E. 1998. *Random number generation and Monte Carlo methods.* New York : Spronger, 1998. 9780387985220.

RAND Corporation. 1955. *A Million Random Digits with 100,000 Normal Deviates.* 1955. 0-8330-3047-7.

Financování práce: vlastní

Vedoucí práce..... podpis: .....

U externích vedoucích fakultní garant práce ..... podpis: .....

Garant oboru mag. studia ..... podpis: .....

Vedoucí katedry/ústavu, kde bude práce vypracovávána ..... podpis: .....

Případný souhlas vedoucího ústavu AV ..... podpis: .....

V Českých Budějovicích dne 23. 2. 2015 podpis studenta: .....

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47 b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové 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.

České Budějovice, 07.04.2018

Poděkování

Tímto bych chtěl poděkovat svým rodičům, Marii a Stanislavu Peckovým, za finanční a morální podporu v průběhu mého studia. Dále své ženě, Ing. Věře Peckové, která mi vytvářela domácí klidnou atmosféru ve chvílích, kdy jsem ji nejvíce potřeboval.

V neposlední řadě děkuji svému vedoucímu Ing. Petru Břehovskému za odborné rady při psaní této práce.

Děkuji také vyučujícím, kteří mě provázeli studiem a svými nároky mě donutili se studované problematice skutečně věnovat.

Velké poděkování patří i firmě Cleverlance Enterprise Solutions a.s., ve které jsem mohl s kolegy konzultovat problematiku řešenou v této práci a která mi svým ohleduplným přístupem umožnila diplomovou práci a studium dokončit.

Obsah

1. Úvod

V dnešní době, kdy lidé nahrazují dříve masovou komunikaci prostřednictvím SMS[1] komunikací internetovou, kdy využívají aplikace typu Messenger[2] nebo WhatsApp[3] od Facebooku nebo Skype[4] od Microsoftu, je zapotřebí klást velký důraz na šifrovanou komunikaci. Ta nám totiž umožní udržet soukromou komunikaci opravdu soukromou a citlivá data diskrétně dopravit z bodu A do bodu B.

Nejde ale jen o soukromou komunikaci. Jde i o bezpečné předávání dat mezi firmami, které chtějí utajit své obchodní tajemství, a především mezi státními orgány a mezinárodními organizacemi, které pracují s citlivými, třeba i tajnými daty nebo s osobními údaji obyvatel.

V době psaní této práce se v médiích často zmiňuje rostoucí trend kyberútoků[5] mezi jednotlivými státy, s cílem zjistit data tajných služeb, ovlivnit volby apod.

Nesmíme ale zapomenout ani na blížící se boom Internet of Things[6] technologií a rychlé rozšíření obrovského množství drobných zařízení připojených do veřejné sítě. Těmito zařízeními nemusí být jen měřící čidla apod., ale i lékařské přístroje – v případě přečtení či manipulace s daty by už nebylo ohroženo jen soukromí, ale i život.

Únikům dat se dá zabránit pouze šifrováním pomocí silných šifrovacích klíčů, které nejdou rozlomit v reálném čase prostřednictvím současných výpočetních zařízení. K tomu je potřeba, aby byl klíč náhodně vygenerován, byl dlouhý, ale zároveň byl vygenerován v co nejkratší době.

Tato práce se právě tímto generováním náhodných čísel zabývá. Je v ní popsán postup implementace několika prototypů generátorů náhodných čísel, získané hodnoty jsou statisticky zhodnoceny a následně vzájemně porovnány z hlediska náhodnosti, rychlosti generování a možnosti masového rozšíření k uživatelům v krátkém čase.

[1] Short message service - systém krátkých zpráv, jedna ze služeb buňkové sítě GSM (39)
[2] Messenger – dříve Facebook Chat – aplikace pro textovou komunikaci mezi uživateli sociální sítě Facebook
[3] WhatsApp – aplikace pro šifrovanou komunikaci mezi mobilními zařízeními
[4] Skype – aplikace pro textovou, hlasovou a video komunikaci mezi elektronickými zařízeními
[5] Kyberútok – útok vedený pomocí prostředků elektronické komunikace, zpravidla přes internet nebo vnitřní počítačovou síť, na jiný elektronický prostředek připojený do stejné sítě
[6] Internet of Things – tzv. internet věcí – umožňuje komunikovat běžnému elektronickému zařízení (např. lednice, automat na kávu, hodinky apod.) s jiným podobným zařízením nebo serverem prostřednictvím veřejné sítě k tomu určené

2. Cíle práce

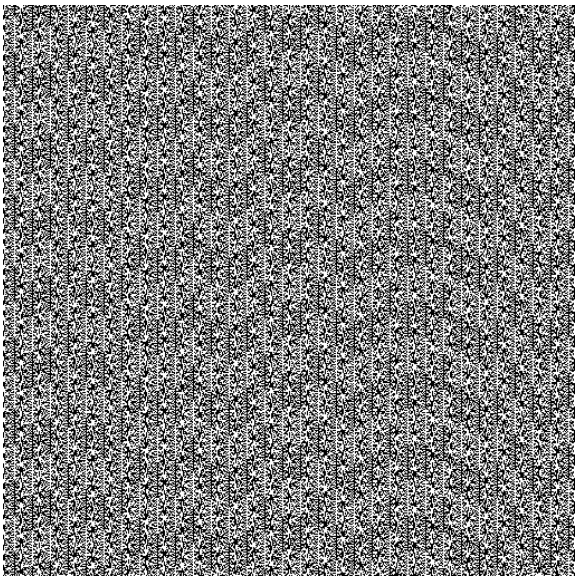
Hlavním cílem této práce je implementace prototypu generátoru skutečně náhodných čísel a následné zhodnocení získaných dat. Aby bylo tohoto cíle dosaženo, byly stanoveny dílčí cíle:

- Vytvoření několika prototypů generátoru náhodných čísel, a to:
  - Snímač pohybu smartphone
  - Snímač dráhy při pohybu prstu po displeji
  - Snímač bodů z kamery smartphone
  - Snímač pohybu myši na displeji počítače
  - Snímač prodlevy mezi stiskem kláves při psaní na počítači
- Návrh praktického užití vytvořených prototypů
- Vygenerování náhodných dat a změření množství vygenerovaných dat za časovou jednotku
- Statistické zhodnocení náhodnosti generovaných dat
- Porovnání prototypů podle různých hledisek

3. Úvod do problematiky

3.1. Pseudonáhodné číslo

Pseudonáhodné číslo je takové číslo, které je získáno algoritmem tak, aby generovaná posloupnost vypadala jako náhodná (1). Jak zdůrazňuje Makovička (2), každý algoritmus je deterministický[7], proto generovaná čísla nejsou zcela náhodná. Mohou se s určitou periodou opakovat, s dostatečným množstvím takto generovaných čísel můžeme číslo následující odhadnout. To dokazuje Obrázek 1, na kterém je zřetelný opakující se vzor. Naopak Obrázek 2 dokazuje, že pokud se použije náhodná externí složka, žádný opakující se vzor se neobjevuje. Výhodou pseudonáhodných čísel je nízká cena získání, než kdybychom generovali čísla opravdu náhodná.



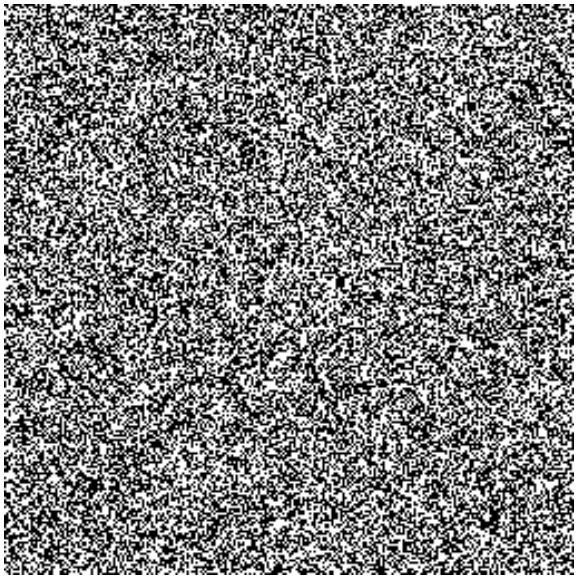
Obrázek 1 - Vizualizace výsledku generování pseudonáhodných čísel funkcí rand() jazyka PHP (3)

3.2. Náhodné číslo

Pro použití v kryptografii, která vyžaduje vysoký stupeň zabezpečení, se pseudonáhodná čísla nehodí. Používají se proto skutečně náhodná čísla generována generátory skutečně náhodných čísel, jejichž vstupem je vždy signál z vnějšího hardwarového zařízení, jehož výstup je nepředvídatelný. Lórencz (4) například zmiňuje fyzikální nebo vnější jevy, jako je radioaktivní rozpad (5), atmosférický šum (3), teplotní šum na analogových součástkách, bílý šum v elektronických obvodech, venkovní teplota, rychlost větru nebo chování uživatele – pohyb počítačovou myší nebo jiným polohovacím

[7] Deterministický je takový algoritmus, který má v každém svém kroku právě jednu možnost, jak pokračovat (42). Vždy ze stejných výchozích (vstupních) podmínek svým během vytvoří stejné výsledky (je tedy předvídatelný) (43).

zařízením, prodlevy při psaní na klávesnici apod. Jindy se použijí věci, které jsou pouhým vybavením domácnosti – např. fotografie lávových lamp (6).



Obrázek 2 - Vizualizace generovaných náhodných čísel službou random.org za použití atmosférického šumu (3)

3.3. Bezpečnost elektronické komunikace

Vzhledem k tomu, že elektronická komunikace většinou prochází prostředím, nad kterým nemáme vlastní kontrolu (např. internetová síť, mobilní síť apod.), je třeba zabezpečit komunikaci už na úrovni zařízení, které pro komunikaci používáme. Toho dosáhneme, pokud data na odesílající straně zašifrujeme a na straně druhé dešifrujeme. K tomu využíváme asymetrické šifrování.

Jsou případy, kdy data potřebujeme bezpečně přenést, ale nepotřebujeme využívat ověřovat identitu odesílatele, například zašifrovat text e-mailu. K tomu se využívá symetrické šifrování.

Stejně jako soukromé nebo důležité firemní dokumenty ukládáme do trezoru, měli bychom chránit i soubory uložené na elektronických nosičích.

3.4. Šifrování

Šifrování, neboli kryptografie, je nauka o metodách utajování obsahu zpráv převodem do podoby, která je čitelná jen se speciální znalostí.

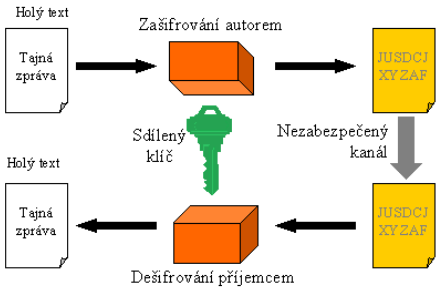
Základními pojmy kryptografie jsou:

- Šifra nebo šifrování – algoritmus, který převádí čitelnou zprávu (prostý text nebo také plain-text) na její nečitelnou podobu.
- Klíč – tajná informace, bez níž nejde šifrovaný text přečíst.
- Symetrická šifra – pro šifrování i dešifrování využívá stejný klíč.
- Asymetrická šifra – pro šifrování je použit veřejný klíč a pro dešifrování soukromý.
- Hashování – algoritmus, který z textu vytvoří krátký řetězec, který identifikuje původní text.
- Certifikáty a elektronický podpis – softwarové prostředky umožňující šifrování textu.

Rozlamováním šifrovaných dat, tedy získání obsahu bez speciální znalosti, se zabývá kryptoanalýza.

3.4.1. Symetrické šifrování

Symetrické šifrování je založeno na principu fungování společného klíče pro zašifrování a i odkódování dat. Nevýhodou je, že odesílatel i příjemce se musí dohodnout na jednom klíči. Problém tedy může nastat při předání tohoto sdíleného klíče.



Obrázek 3.1 - Symetrické šifrování (7)

Tímto klíčem mohou být kódové tabulky, šifrovací kotouče, nastavení šifrovacího stroje, domluvená abeceda (např. Morseova abeceda[8]) apod.

Obrázek 3.2 - Albertiho šifrovací disk – z doby americké občanské války (8)

[8] Morseova abeceda je sestava kódů interpretujících písmena, čísla a speciální znaky (45)

3.4.1.1. Blowfish

Šifra navržená Brucem Schneierem a jeho týmem jako alternativa k DES[9]. Jedná se o technologicky méně náročnou šifru. Základem je spolehlivý a ozkoušený algoritmus, navržena byla s ohledem na 32 bitové prostředí. Bohužel má komplexní vnitřní struktury, kvůli kterým se jen velmi těžko analyzují její slabá místa a nedostatky (9).

Velikost bloku šifry je 64 bitů a délka jejího klíče je maximálně 448 b, (tj. 56B). Tento počet bitů je pak pevně daný. Operace použité v Blowfish algoritmu jsou XOR[10] (plně disjunkční) a obsahují sčítání 32 bitových slov (10).

Blowfish patří mezi tzv. blokové šifry. Jejím výstupem je šifrovaný text, který může být šifrován i několikrát dokola, aby se zvýšila bezpečnost. To je také jeden z důvodů, proč se nehodí na šifrování velkých databází. Vhodná je proti slovníkovým útokům (10).

Tato šifra byla svým tvůrcem vytvořena jako neplacená a nelicencovaná.

3.4.1.2. Twofish

Také šifra Twofish vychází z „dílny“ Bruce Schneiera. Twofish navazuje na Blowfish, jedná se o její sofistikovanější verzi. Založena je na Feistelově síti, S-boxech, MDS matici a pseudo-Hardamanových transformacích. Aplikován zde byl požadavek na bezpečnou metodu šifrování americké organizace NIST (National institute of standards and technology). Nicméně i přesto Twofish nebyla standardizována AES (Advanced encryption standard – Standard pokročilého šifrování) (11).

Twofish využívá 128 až 256 bitovou délku šifrovacího klíče (10).

3.4.1.3. IDEA

Zkratka IDEA znamená International data encryption algorithm, česky nazýváno bloková šifra. Má délku bloku 64 bitů a pracuje s velikostí klíče 128 bitů, opět je bezpečnější než DES (12).

Publikována byla v roce 1991 jako nástupce předchozího algoritmu PES poté, co bylo oznámeno jeho prolomení.

---

[9] DES (Data/Digital Encryption Standard – Standard datového/digitálního šifrování) – šifra vyvinutá v roce 1975 – 1977 firmou IBM, používala pouze 54 bitové šifrování (po intervenci NSA (12)), dnes je považována za nedostatečnou, protože běžný počítač je schopen ji prolomit za 24 hodin, je zcela nevhodná např. pro použití v bankovnictví (10).

[10] XOR – bitová operace - exkluzivní součet vrací 1 v případě, že se sčítance liší, v ostatních případech vrací 0 (46).

3.4.1.4. Rijndael

Akronym navržený Joanem Daemenem a Vincentem Rijmenem. Rijndael se stal vítězem soutěže o budoucí standard AES v roce 2001. Validní je s klíči NIST (National institute of standards and technology) v délce 128, 192 a 256 bitů. Založen je na stejných principech jako DES, ale je vyváženější a zatím nebyl zpochybněn (9).

3.4.1.5. RC4

Licenčně placená bloková šifra RC4 navržena Donaldem Rivestem. Je jednou z nejpoužívanějších šifer v komerčním použití. Její klíč může mít maximálně 2048 bitů, v praxi se však používá 128 bitů. Šifra je postavena na míchání bitů a permutace klíče.

Donald Rivest postupně svoji šifru aktualizoval a publikoval ještě RC5 a RC6, které využívají rotace závislé na datech (13).

3.4.1.6. Vernamova šifra

Vernamova šifra je založena na jednoduchém principu – každý znak zprávy se posune o náhodně zvolený počet míst v abecedě na náhodnou pozici – prakticky se jedná o výměnu znaku za zcela náhodný znak jiný. Na každou komunikaci se použije nový klíč. A pokud se navíc klíč generuje skutečně náhodně, pak šifru nelze prolomit (14).

Pokud ale při šifrování použijeme klíč kratší, než je zpráva samotná, nebo použijeme generátor pseudonáhodných čísel[11] zvyšujeme tím pravděpodobnost prolomení (15). Stejně tak znovupoužití šifry snižuje bezpečnost.

V praxi se pro výměnu klíčů používá velký soubor vygenerovaných čísel předaný za bezpečných okolností mezi dvěma komunikujícími stranami a při zasílání šifrované zprávy se z tohoto souboru „ukrajuje“ podle délky zaslané zprávy.

3.4.2. Asymetrické šifrování

3.4.2.1. RSA

Algoritmus RSA je dlouhodobě považován za jedno z nejlepších schémat. Název je složen z iniciál příjmení svých tvůrců (R – R. **R**ivest, S – A. **S**hamir, A – L. **A**delman), sestaven byl již v roce 1978. Fungování klíče je zaručeno i s menší délkou než v případě jiných algoritmů. Výpočet je založen na součinu velkých prvočísel. Bezpečný bude do té doby, než bude nalezen algoritmus faktorizace velkých čísel v krátkém čase. Používá

[11] Nejlepší je použít generátor čísel založený např. na fyzikálních metodách či kvantových procesech.

šifrování s veřejným klíčem – jednosměrný hashovaný přenos, vhodný pro elektronické podpisy (12).

„Je snadné vynásobit dvě dlouhá (100 místná) prvočísla, ale bez jejich znalosti je prakticky nemožné zpětně provést rozklad výsledku na původní prvočísla.“ (13)

Tento algoritmus je velmi citlivý na implementaci, protože úspěšné útoky na tento algoritmus byly často založeny právě na implementačních chybách – výběr prvočísla, ověření prvočísla (9).

Běžná délka RSA je 1024 bitů, výjimkou však není ani 3072 bitů. Přičemž 1024 bitů RSA je ekvivalentem 80 bitům symetrického klíče (16).

3.4.2.2. AES

Advanced Encryption Standard – Standard pokročilého šifrování. AES měl původně nahradit DES (Data/Digital Encryption Standard – Standard datového/digitálního šifrování), který se v dnešní době již přestává používat (9).

AES byl původně vytvořen americkou vládou, dnes se běžně využívá pro šifrování Wi-Fi – zabezpečení WPA2. Prozatím tato šifra nebyla prolomena (10).

Délka klíče AES je podporována ve velikosti 128, 192 a 256 bitů.

3.4.2.3. SSL a TLS

SSL (Security socket layer) je předchůdce protokolu TSL (Transport security layer). Oba tyto protokoly zajišťují zabezpečený přenos dat v internetu (9).

Norma SSL byla vytvořena společností NETSCAPE, dnes se využívá jako bezpečný protokol pro zabezpečení https. Používá se tedy pro komunikaci, která nesmí být odposlechnuta. Uživatel potřebuje pro ověření důvěryhodnosti platný certifikát. Norma SSL je založena na symetrické šifře RC4 a asymetrické šifře RSA. Její velikost je 128 bitový klíč (13).

3.4.3. Elektronický podpis

Elektronický podpis nahrazuje autentičnost klasického podpisu v digitální formě. Díky tomu je možné podepsat i takové formáty dat, které by jinak fyzicky podepsat nešly. V českém právním systému se mu věnuje zákon č. 227/2000 Sb., o elektronickém podpisu (17).

Výše uvedený zákon hovoří o elektronickém podpisu jako o údajích v elektronické podobě, které jsou připojeny k datové zprávě a slouží jako k jednoznačnému ověření identity osoby, která data podepsala. Zaručený el. podpis pak ještě kromě výše uvedeného byl vytvořen a připojen ke zprávě pomocí prostředků, které může podepisující osoba držet výhradně pod svou kontrolou a další změny dat jsou jasně identifikovatelné.

Dalším pojmem spojeným s el. podpisem jsou např. elektronická značka – ta je připojena k datové zprávě a jednoznačně umožňuje identifikaci podepsané osoby pomocí kvalifikovaného systémového certifikátu.

Při podepisování dokumentů el. podpisem využíváme privátního a veřejného klíče. Privátní klíč slouží k vytvoření el. podpisu – je to jedinečný klíč generovaný u poskytovatele certifikačních služeb. Veřejný klíč pak slouží pro ověřování elektronického podpisu (17).

Při tvorbě el. podpisu se nešifruje celá zpráva, ale data nejdříve projdou tzv. hashovací funkcí. Zde je vytvořen hash – řetězec, otisk – vytvořený z libovolně dlouhého textu následně o konstantní délce. SHA-2 je hashovací funkce používaná od 31.12.2010 dle nařízení Národního bezpečnostního úřadu, dříve byly používána SHA-1 (17).

Elektronický podpis je datová struktura, která ověří původ zprávy a pravost podepsání dokumentu. Příjemce, který ověřuje pravost podpisu si vypočítá otisk dokumentu, ten následně dešifruje veřejnou částí klíče odesílatele a následně porovná s řetězcem, který si „sám“ vypočítal. Pokud se otisky rovnají, dokument nikdo nepozměnil (18).

3.5. Statistické hodnocení náhodnosti čísel

I přesto, že hardwarové generátory náhodných čísel jsou poměrně spolehlivá zařízení, je důležité výsledky ověřovat. Mezi faktory, které mohou ovlivnit výsledky generování jsou změny teploty, napájecího napětí nebo elektromagnetické rušení. Mohou se objevit i softwarové chyby v programu zpracovávajícím vstupní signály, případně hardwarové závady, které jsou těžko odhalitelné. Proto jsou generovaná čísla statisticky hodnocena, aby se ověřila náhodnost výstupu.

Podle Lórencze (4) je třeba mít na paměti, že statistickými testy lze ukázat, že generátor nejspíše není kvalitní, ale nelze prokázat, že je kvalitní. Skutečnost, že generátor „projde" všemi testy, ještě nezaručuje, že neobsahuje slabinu, jež testy nebyly schopny odhalit.

3.5.1. Pravděpodobnost

Pravděpodobnost jevu určuje, s jakou mírou jistoty se dá očekávat, že daný jev nastane. Nechť pro náhodný pokus platí:

- Všech možných výsledků je konečně mnoho (tj. ω je konečná množina).
- Nemohou nastat dva výsledky současně.
- Každý výsledek je stejně možný.

Pak platí, že pravděpodobnost jevu *A*, označena jako *P(A)* je rovna

$$P(A) = \frac{|A|}{|\omega|} \tag{3.3}$$

Pravděpodobnost se uvádí jako desetinné číslo z intervalu $\langle 0, 1 \rangle$ nebo pomocí procent (19).

3.5.2. Entropie informace

Pokud chceme zjistit informaci, kterou obdržíme, je třeba se zabývat entropií informace. Ta vyjadřuje míru nejistoty obsažené v náhodném ději. Pokud tedy máme konečný počet vzájemně se vylučujících jevů, jejichž pravděpodobnosti výskytu jsou $p_i(x), \ldots, p_n(x)$, entropii vyjádříme jako funkci těchto pravděpodobností (20).

Podle Lórencze se entropie vztahuje ke schopnosti útočníka předpovědět vygenerovanou hodnotu. Pokud útočník následující generovanou hodnotu s jistotou zná, entropie je nulová.

Dále uvádí, že entropie vyjadřuje průměrný počet bitů nutných k zakódování hodnoty při použití optimálního kódování – vyjadřuje obsažené množství informace uvedené v bitech (4).

Požadované vlastnosti funkce pro výpočet množství informace jsou:

- Jev[12] *X* má *n* realizací[13], množství informace je funkcí *n*.
- Je-li *n = 1*, jedná se o jev jistý, množství informace je rovno nule.
- Jevy *X* a *Y* probíhající současně a nezávisle, $p(x,y) = p(x) \cdot p(y)$: množství informace je dáno součtem množství jednotlivých jevů: $f(x,y) = f(x) + f(y)$
- Jev *X* má *n* realizací, jev *Y* má *m* realizací. Je-li $m > n$, pak musí i $f(m) > f(n)$

Funkce, která těmto podmínkám vyhovuje je logaritmus $I(x) = log\ n$ za předpokladu, že pravděpodobnost každé realizace je stejná. Má-li jev *n* realizací, lze psát $p(x) = 1/n$, odsud pak $n = 1/p(x)$.

Buď *X* množina výsledků náhodného děje, *x* výsledek realizace a $p(x)$ pravděpodobnost tohoto výsledku. Každému *x* z *X* pak lze přiřadit reálné číslo $I(x)$ nazývané vlastní informace o výsledku *x*, pro než platí $I(x) = - log\ p(x)$. Toto číslo $I(x)$ představuje množství informace obsažené ve výsledku *x*. Čím menší je pravděpodobnost výsledku, tím je větší množství informace obsažené ve výsledku.

Informační množství celého jevu X vyjadřuje entropie informace. Předpokládejme, že jev *X* má *n* realizací $X = x_1, x_{2, \ldots,} x_n$ s pravděpodobnostmi $p(x_1), p(x_2), \ldots, p(x_n)$. Entropie $H(X)$ je dána určitou střední hodnotou vlastních informací všech realizací jevů:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log p(x_i) \tag{3.4}$$

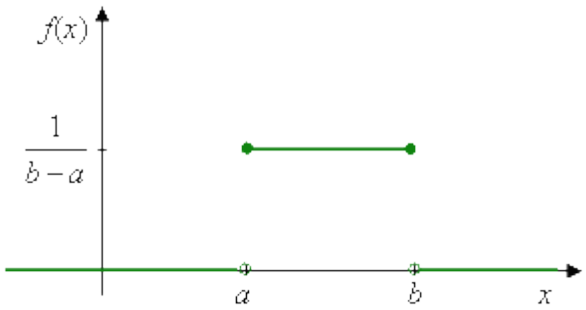
3.5.3. Kolmogorov-Smirnovův test

„Test je založen na porovnání teoretické a empirické distribuční funkce. Velké odchylky mezi těmito dvěma funkcemi svědčí o tom, že rozdíl mezi modelovými a vygenerovanými hodnotami není způsoben pouze náhodnými vlivy“ (21).

[12] Jev je náhodný proces s n možnými realizacemi, např. tah loterie
[13] Realizace jevu je jeden projev, získání výsledku

Otipka (22) rovnoměrné rozdělení a distribuční funkci definuje následovně: Náhodná veličina *X* má rovnoměrné rozdělení *R(a,b)* právě tehdy, když je hustota pravděpodobnosti určena vztahem

$$\Phi(\mathrm{x}) = \begin{cases} \dfrac{1}{b-a} & \text{pro } x \in \langle a, b\rangle \\ 0 & \text{pro } x \notin \langle a, b\rangle \end{cases} \tag{3.5}$$



Obrázek 3 - Graf hustoty pravděpodobnosti pro rovnoměrné rozložení (22)

Nechť $X_1, \ldots, X_n$ je náhodný výběr ze spojitého rozložení s distribuční funkcí

$$\Phi(\mathrm{x}) = \begin{cases} 0 & \text{pro } x \in (-\infty, a) \\ \dfrac{x-a}{b-a} & \text{pro } x \in \langle a, b\rangle \\ 1 & \text{pro } x \in (b, \infty) \end{cases} \tag{3.6}$$

„Výběrovou distribuční funkci označíme $F_n(x)$, tj. pro $\forall x \in R$:

$$F_n(x) = \frac{1}{n} card\{i; X_i \leq x\} \tag{3.7}$$

Na hladině významnosti α testujeme nulovou hypotézu

$$H_0: \Phi(x) = F_n(x) \text{ pro } \forall x \in R \tag{3.8}$$

proti alternativě

$$H_1: \Phi(x) \neq F_n(x) \tag{3.9}$$

pro aspoň jednu hodnotu x. Testová statistika má tvar

$$D_n = \max_{x \in R} |\phi(x) - F_n(x)| \tag{3.10}$$

.

Nulovou hypotézu zamítáme na hladině významnosti α, když

$D_n > D_{n,a}$, kde $D_{n,a}$ je tabelovaná kritická hodnota.

Pro větší *n* lze kritickou hodnotu aproximovat výrazem

$$D_{n,a} \approx \sqrt{\frac{1}{2n} ln \frac{2}{\alpha}}.\text{“ (21).} \tag{3.11}$$

3.5.4. Extremální body (body zvratu)

Test zkoumá, zda kolísání hodnot podle velikosti se v posloupnosti $x_i, \ldots, x_n$ mění dostatečně rychle. Není vhodný pro testování existence trendu, protože vychází pouze z lokálních vlastností posloupnosti.

Číslo x_i se nazývá bodem zvratu, když obě sousední čísla jsou současně buď větší než x_i, nebo menší než x_i, tj. platí-li buď $x_{i-1} > x_i < x_{i+1}$ nebo $x_{i-1} < x_i > x_{i+1}$.

Testovou statistiku zkonstruujeme následně: Označme *Y* celkový počet bodl zvratu v posloupnosti $x_i, \ldots, x_n$. Platí-li H_0, pak statistika má asymptoticky normální rozdělení se střední hodnotou $E(Y) = \frac{2(n-2)}{3}$ a rozptylem $D(Y) = \frac{16n-29}{90}$, tedy standardizovaná statistika

$$U = \frac{Y - \frac{2(n-2)}{3}}{\sqrt{\frac{16n-29}{90}}} \approx N(0,1) \tag{3.12}$$

Kritický obor $W = (-\infty, -u_{1-\frac{\alpha}{2}}\rangle \cup \langle -u_{1-\frac{\alpha}{2}}, \infty)$. Pokud $U \in W$, nulovou gypotézu zamítáme na asymtotické hladině významnosti α.

Nulová hypotéza H_0: posloupnost je náhodná proti alternativě H_1: posloupnost není náhodná.

3.5.5. Test znamének diferencí

Test zkoumá, zda posloupnost neobsahuje dlouhé řady čísel jdoucích za sebou vzestupně nebo sestupně. Používá se k ověření existence trendu.

Test je založen na počtu kladných 1. diferencí dané posloupnosti, tj. na počtu bodů růstu. Číslo x_i se nazývá bodem růstu, když $x_i < x_i+1$.

Konstrukce testové statistiky: Označme Y celkový počet bodů růstu v posloupnosti $x_1, \ldots, x_n$.

Platí-li H_0, pak statistika Y má asymptoticky normální rozložení se střední hodnotou $E(Y) = \frac{n-1}{2}$ a rozptylem $D(Y) = \frac{n+1}{12}$, tedy standardizovaná statistika

$$U = \frac{Y - \frac{n-1}{2}}{\sqrt{\frac{n+1}{12}}} \approx N(0,1) \tag{3.13}$$

Kritický obor: $W = (-\infty, -u_{1-\alpha/2}\rangle \cup \langle -u_{1-\alpha/2}, \infty)$. Pokud $U \in W$, nulovou hypotézu zamítáme na asymptotické hladině významnosti α.

Nulová hypotéza H_0: posloupnost je náhodná proti alternativě H_1: posloupnost není náhodná (21).

3.5.6. Autokorelační test

Test zkoumá sériovou korelaci po sobě jdoucích členů posloupnosti $x_1, \ldots, x_n$. Porovnávacím prvkem je koeficient R, který vyjadřuje, jakou měrou závisí člen x_{i+1} na členu x_i. Výpočet koeficientu realizujeme vztahem

$$R_k = \frac{\frac{1}{n-k}\sum_{i=1}^{n-k}(x_i - \bar{x})(x_{k+i} - \bar{x})}{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_{k+i} - \bar{x})^2} \tag{3.14}$$

kde $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Lze ukázat, že jsou-li $x_1, x_2, \ldots, x_n$ nezávislé, stejně rozdělené veličiny s konečnou disperzí $D(X) = \sigma^2$, potom má náhodná veličina $\sqrt{n}R_k$ pro $n \to \infty$ a k pevné limitní rozdělení $N(0,1)$. Pro testování shody rozdělení hodnot $\sqrt{n}R_k$ lze použít chi-kvadrát test dobré shody (23).

3.5.7. Chi-kvadrát test

Testuje hypotézu, která tvrdí, že náhodný výběr $X_1, \ldots, X_n$ pochází z rozložení s distribuční funkcí $\Phi(x)$.

Je-li distribuční funkce spojitá, data rozdělíme do r třídících intervalů $\langle u_j, u_{j+1}\rangle, j = 1, \ldots, r$. Zjistíme absolutní četnost n_j j-tého třídícího intervalu a vypočteme pravděpodobnost p_j, že náhodná veličina X s distribuční funkcí $\Phi(x)$ se bude realizovat v j-tém třídícím intervalu. Platí-li nulová hypotéza, pak $p_j = \Phi(u_{j+1}) - \Phi(u_j)$ (21).

3.5.8. Poker test

Aby se vyloučil chybně dobrý výsledek testu rovnoměrného rozdělení v případě posloupnosti, která není dostatečně náhodná, např. „111222333444555666“, v níž je patrné, že počet členů, které padnou do jednotlivých intervalů sice odpovídá rovnoměrnému rozložení, nikoli však náhodné posloupnosti, byl zaveden Poker[14] test.

Posloupnost generovaných čísel rozdělíme do pětic, následně zjistíme počet různých čísel v pětici, který označíme písmenem r. Poté použijeme kritérium s pravděpodobnostmi

$$p_r = \frac{d(d-1)\ldots(d-r+1)}{d^k}\binom{k}{r} \tag{3.15}$$

Kde p_r je pravděpodobnost, že v pětici bude právě r různých čísel, k je počet čísel ve skupině, d je maximální možná hodnota generované veličiny zvětšená o 1. Využívá se pro testování posloupností, jejichž členy jsou kladná celá čísla x, pro která platí $x \in \langle 0, d-1\rangle$ (24). Pravděpodobnosti výskytu kombinací zobrazuje Tabulka 1.

Kombinace	Slovní popis	Pravděpodobnost
abcde	jednice	0,3024
aabcd	dvojice	0,5040
aabbc	dvě dvojice	0,1080
aaabc	trojice	0,0720
aaabb	trojice a dvojice	0,0090
aaaab	čtveřice	0,0045
aaaaa	pětice	0,0001

Tabulka 1 - Pravděpodobnostní tabulka kombinací pro poker test

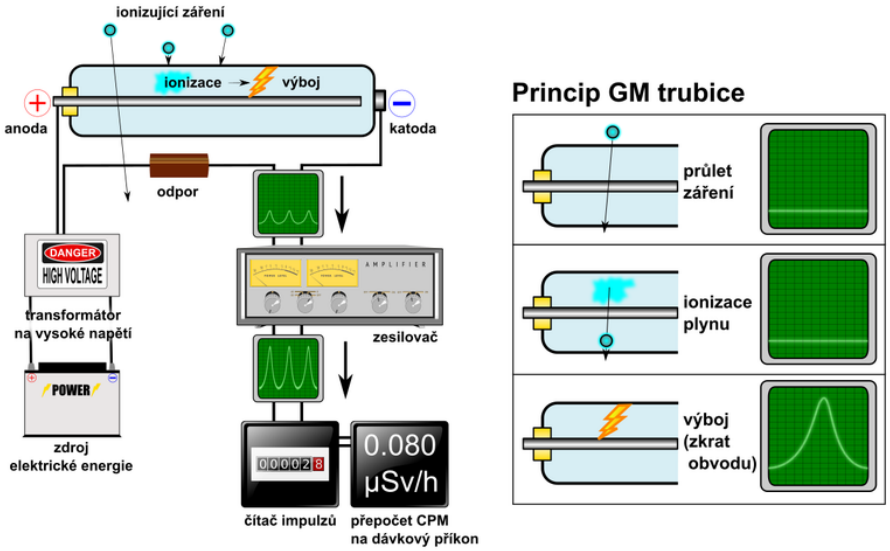
[14] Poker je karetní hra vyvinutá v USA. Hraje se na základě sázek na výherní kombinace 5 karet, které je možné složit ze 2 karet v hráčově ruce a 5 vyložených karet dealerem.

4. Stávající řešení generátorů skutečně náhodných čísel

4.1. HotBits – radioaktivní rozpad

Hotbits je generátor čísel, který nabízí skutečně náhodné sekvence. Řídí se inherentní[15] chybou kvantových mechanických přírodních zákonů při radioaktivním rozpadu. Generátor vytvořil John Walker.

Hotbits přenáší po sobě jdoucí dvojice radioaktivních rozpadů pomocí Geiger-Müllerovy trubice propojené s počítačem. Jakmile jsou data jednou poslána nějakému uživateli jsou okamžitě smazána a ta samá data již nikdy neobdrží jiný uživatel ani nejsou nikde uchovávána. Tudíž nikdy dva uživatelé nepoužijí stejnou sadu dat. Alternativa pro stažení dat z HotBits pro pozdější použití je poskytována balíčkem randomX pro Javu.



Obrázek 4 - Princip Geiger-Mülerovy trubice (25)

Pokud chce nějaká molekula, atom, hvězda či jiný fyzikální systém snížit svoji energii, může, ale vždy v mezích fyzikálních zákonů. Kvantová mechanika říká, co nastane, ale neříká kdy. A tato mezera je využita pro generování náhodných čísel.

Jako zdroj záření se pro tento generátor využívá cesium. Při beta[16] rozpadu jádra cesia-137 (radioaktivní izotop cesia 137Cs, který má poločas rozpadu 30, 17 let), vzhledem k uspořádání, neexistuje žádný způsob, jak zjistit, kdy se dané jádro

[15] Inherentní - daná, neoddělitelná, neodmyslitelná; charakterizuje atributy věcí, jež nejsou přidané a nahodilé, nýbrž plynou ze samé povahy příslušné věci – v tomto případě přírody a jejích zákonů
[16] Beta rozpad = rozpad beta částic (elektronů), kdy jsou všechny elektrony totožné. Název beta elektron vznikl ještě před tím, než fyzici přišli na to, že „beta paprsky“ a elektrony jsou jedna a ta samá věc

rozpadne. Jádro Caesia-137 (má 137 protonů a neutronů v atomu) se spontánně proměňuje v metastabilní jádro elementu barya (137Ba), jež má poločas rozpadu 156 sekund. Když elektron vyletí z jádra začne excitované jádro barya vyzařovat gama paprsek.

Skutečný čas rozpadu je náhodný a jestliže je náhodný samotný rozpad, pak i interval mezi dvěma po sobě jdoucími rozpady je také náhodný. Stačí změřit dvojici těchto intervalů a udělit jim nulu nebo jeden bit na základě relativní délky obou intervalů. Při počítání je zaznamenán čas prvního impulzu (jako T1), poté se vyčká na druhou dvojici impulzů (T2) a změří se interval i mezi nimi. Pokud jsou stejné, výsledky měření jsou zahozeny a provádí se nový pokus. Jinak je-li T1 < T2 udělíme jim nulový bit, pokud je T1 > T2, obdrží jeden bit (26).

Tento generátor není možné otestovat sadou Diehard, protože tato sada vyžaduje 10-11 MB vstupní soubor, kdežto server poskytuje maximálně 4 KB, protože náhodné bity jsou generované pomalu – 100 B/sek., a po zaslání uživateli jsou smazané ze serveru (27).

4.2. Random.org – atmosférický šum

Random.org nabízí generátor náhodných čísel, kdy náhodnost pochází z atmosférického šumu, který je pro mnoho účelů lepší než algoritmicky pseudonáhodná čísla. V současné době jsou data poskytovaná touto službou využívána pro loterijní, vědecké účely, pro online hry atd. Služba existuje od roku 1998, kdy byla zprovozněna doktorem Madsem Haadrem z Trinity College v Dublinu (28).

Za atmosférický šum mohou elektromagnetické poruchy např. bouřkového původu, které mohou být například slyšet při rádiovém příjmu jako praskot. Atmosférický šum se může šířit na tisíce kilometrů, je velmi snadné ho získat, ale je nutné se vyvarovat zdrojů, které by do něj vnášely prvky periodičnosti.

Šum rozlišujeme vnější a vnitřní/uměle vytvořený (29):

- vnější se vyskytuje v přírodě z různých zdrojů (již zmíněné bouřky, ale i například vítr, moře, paprsky slunce, vesmírný prach dopadající na Zemi aj.),
- vnitřní/umělý – je nechtěně vytvořený např. v elektronických obvodech – tepelný, při přenosu signálu, šum z letadel, automobilů, elektrických motorů, spínacích zařízení, vysokého napětí, zářivek atd.

Další dělení šumu:

- bílý – náhodný s rovnoměrnou výkonovou spektrální hustotou, nejlépe využitelný ke generování náhodných čísel, využití bílého šumu v praxi:
    - testování kmitočtové odezvy zesilovačů, elektrických filtrů a podobných obvodů
    - generování náhodných čísel
    - automatická equalizace
- barevný – s nerovnoměrným rozložením hustoty kmitočtových složek ve spektru,
- praskavý šum – tento je vytvářen znečistěným přechodem mezi bází a emitorem iontů těžkých kovů, hustota jejich výkonu má mnoho frekvencí a klesá s rostoucím počtem kmitočtů,
- blikavý – blikavý šum způsobují poruchy v krystalové mřížce, tento jev se projevuje především v nižších kmitočtech,
- tepelný – velké množství volných iontů a elektronů ve vodičích vytváří vibrace a teplo, jejich fluktuace vytváří proud, zdroj odporu ve vodiči je tedy nejlepším pro zcela náhodný jev.

4.3. Teplotní šum na analogových součástkách

Šum je náhodný signál, který narušuje zpracování a přenos užitečného/správného signálu. Základní rozdělení šumu je externí (mimo analyzovaný obvod) a interní (uvnitř analyzovaného obvodu).

Vlastnosti tepelného šumu (29):

- vyskytuje se jak ve vodičích, tak polovodičích,
- vzniká v důsledku náhodného pohybu elektronů (pokud je teplota vyšší než 0 K),
- má konstantní spektrální výkon až do kmitočtů cca 100THz,
- šumové napětí rezistoru.

Šumové číslo říká, kolikrát se zhorší poměr signálu na výstupu proti původní hodnotě vstupu obvodu při oboustranném výkonovém přizpůsobení.

Jedná se o teplotní šum generovaný pohybem elektronů v elektrických zařízeních. Velikost šumu je úměrná teplotě a nezávisí na frekvenci, napětí ani proudu.

Rozlišujeme:

- výstřelový šum – ten je způsobený velkou rychlostí pohybu elektronů v zařízení, které je napájeno,
- inverzní šum – výsledek nečistot ve vodivostním pásu polovodičů, lze ho potlačovat zvýšením frekvence,
- okolní šum – neomezený počet okolních zdrojů – využití kombinace např. teploty, zvuků a dalších šumů,
- kvantizační šum – vzniká při zpracování analogového signálu a jeho převedení do digitální formy.

Veškerý šum lze softwarově i hardwarově redukovat, např. výpočtem průměrného signálu (SW), výpočtem průměrného signálu v určité oblasti (SW), uzeměním (HW), analogovým filtrováním (HW), stíněním (HW), modulací (HW) apod.

4.4. Veracrypt - chování uživatele

Pro generování náhodných čísel můžeme využít i softwarová řešení, která spoléhají na náhodnost zainteresovaných komponent – pohyb myši po monitoru PC uživatele nebo závisející na stisku kláves.

Generátor skutečně náhodných čísel je k počítači připojené (či v procesoru obsažené) zařízení generující náhodná čísla z fyzikálního procesu. Nicméně toto zařízení je náchylné na změnu prostředí – tzn. v případě stejného uživatele je možné časem získávat data pseudonáhodná. Pokud však bude obsluha PC různorodá, jde na základě nepředvídatelnosti jejího chování získávat data zcela náhodná např. při pohybu myši po podložce, při stisku kláves.

Při pohybu myši např. program Randomgen využívá informace o poloze myši – souřadnice pohybu (velikost volné paměti a swapovací oblasti), z kterých algoritmem získává číslo.

Při použití klávesnice pro generování náhodných čísel můžeme např. využít jako vstupní hodnotu frekvenci stisku kláves, nicméně stisky kláves bývají operačními systémy průběžně ukládány do bufferu, tudíž nejdříve dochází k jejich nahromadění a až posléze k odeslání a vyhodnocení, čímž dochází k jejich zkreslení.

5. Použité technologie

5.1. Programovací jazyk Java

V roce 1991, kdy nejpoužívanějšími moderními programovacími jazyky byly C a C++ vznikl požadavek na platformě nezávislý programovací jazyk. Platformě nezávislým programovacím jazykem se rozumí jazyk nezávislý na architektuře počítače, který je možné kromě standardních PC použít i ve spotřební elektronice, jako je např. mikrovlnná trouba, dálkové ovladače atd. Pracovníci společnosti Sun Microsystems, konkrétně James Gosling, Patrick Naughton, Chris Warth, Ed Frank a Mike Sheridan, tedy začali pracovat na jazyku, který nazvaly Oak. Od počáteční implementace, která trvala 18 měsíců, do prvního oznámení jazyka Java v roce 1995, na kterou byl jazyk Oak přejmenován, se na rozvoji podílelo mnoho dalších programátorů.

Problémem ostatních jazyků, které byly do roku 1991 dostupné bylo, že byly navrženy tak, aby pro každý procesor musel být zvláštní kompilátor. I když například pro jazyk C++ existovaly kompilátory pro téměř každý typ procesoru, vždy při vzniku nového procesoru bylo třeba vytvořit nový kompilátor, což bylo velmi drahé a časově náročné.

Druhý důvod, který podle Schildta (30) jazyku Java zajistil tak důležité postavení a přežití do dnešní doby, byl vývoj webu a opětovný požadavek na platformní nezávislost programů. Zatímco do této doby byl počítačový svět rozdělený na 3 konkurenční tábory zastánců (OS na platformě Intel, Mac OS a UNIX), se světem internetu se architektura roztříštila na mnoho dalších táborů.

Klíč, který řeší výše zmíněné problémy je bajt kód – výstupem kompilátoru Javy totiž není spustitelný kód, ale optimalizovaná sada instrukcí určená pro virtuální stroj Javy (JVM – Java Virtual Machine). Znamená to tedy, že pro každý systém je třeba vytvořit pouze tento interpreter, který je schopný zpracovat tentýž bajt kód.

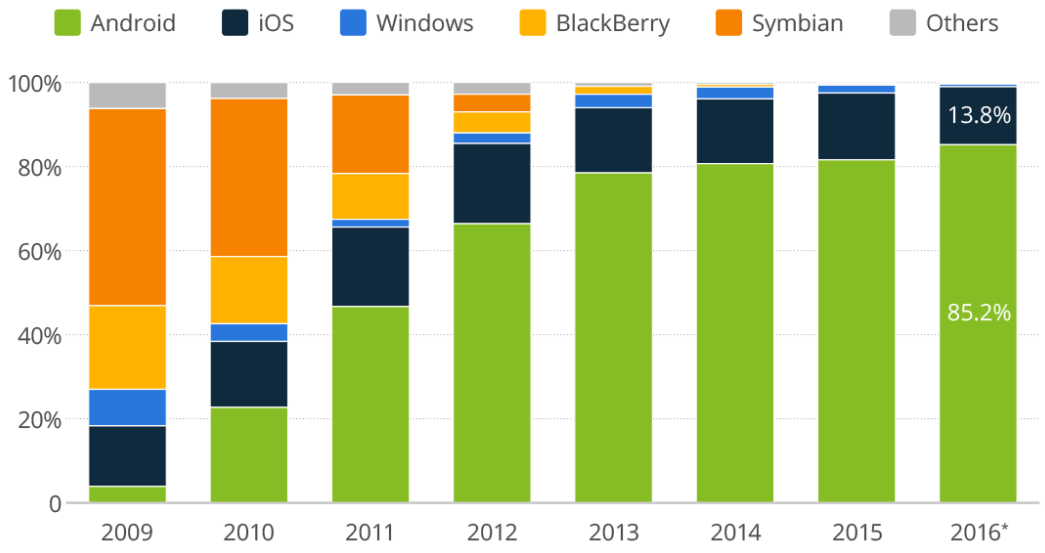
Mezi další výhody Javy podle Schildta (30) patří:

- Jednoduchost – tvůrci využili a jen lehce upravili syntaxi jazyka C, pro programátory tedy bylo a je jednoduché se Javu rychle naučit. Stejně takpřevzali principy objektového programování.

- Objektová orientace – díky implementaci tří základních OOP přístupů (zapouzdření, dědičnost, polymorfismus) si Java zachovává velkou přehlednost napsaného kódu a neuvěřitelnou rychlost a variabilitu vývoje.
- Robustnost – díky silnému typování probíhá kontrola kódu už při kompilaci. Další kontrola probíhá za běhu programu a vzhledem k objektové práci s výjimkami je nemožné v kódu vytvořit těžko dohledatelné chyby – všechny chyby a reakce na ně dokáže řídit samotný program.
- Podpora multithreadingu – práce s více vlákny – pokud je procesor schopný pracovat s více vlákny, je možné provádět více operací najednou, čímž sevýrazně zrychluje běh aplikace.
- Distribuovanost – protože je Java navržena pro prostředí internetu, standardně podporuje protokol TCP/IP. Přístup k síťovému zdroji je tedy stejně jednoduchý, jako práce se souborem.

5.2. Operační systém Android

Android je mobilní operační systém založený na jádře operačního systému Linux. Ješířen jako opensource[17] a v současné době je výrazně nejrozšířenějším na trhu (viz Obrázek 5). Programovacím jazykem, který je použit pro psaní aplikací pro Android je jazyk Java.



Obrázek 5 - Poměr mobilních operačních systémů na trhu (31)

[17] Opensource - Software s otevřeným kódem. U programů typu open source dostupné zdrojové kódy, které lze za podmínek popsaných v licenci upravovat. (47)

Operační systém Android vznikl v roce 2003, kdy stejnojmennou společnost založili Andy Rubin, Rich Miner, Nick Sears a Chris White. Cílem bylo vytvořit operační systém pro moderní chytré přístroje. Z počátku byli jen jedni z mnoha s tímto cílem, avšak roku 2005 Android koupila společnost Google.

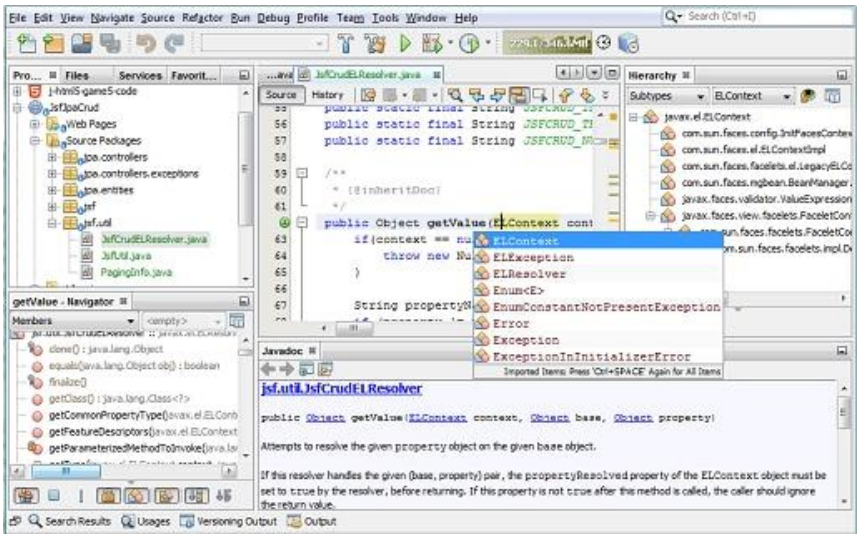
Google v roce 2007 založil spolu se společnostmi Nvidia, Samsung, LG, HTC, Motorola, Intel, Qualcomm, Ebay, T-Mobile a Telefonica konsorcium Open Handset Alliance, jehož cílem bylo podporovat a vyvíjet otevřený operační systém Android. První telefon s OS Android se dostal do prodeje v roce 2008 (32).

Aplikace pro OS Android jdou distribuovat buďto napřímo jako APK instalační soubory, nebo přes oficiální obchod Android Market.

5.3. Netbeans IDE

Netbeans IDE[18] je nástroj, pomocí kterého mohou vývojáři psát, překládat a ladit aplikace. Netbeans je napsaný v jazyce Java, podporuje ale prakticky jakýkoliv jazyk.

Netbeans je šířený jako opensource, je tedy možné jej bezplatné používat v komerčním i nekomerčním prostředí, s velkou komunitou vývojářů. Díky tomu lze základní prostředí Netbeans IDE jednoduše rozšířit pomocí modulů Projekt byl založen v roce 2000 společností Sun Microsystems, která je zároveň (nyní prostřednictvím mateřské společnosti Oracle) hlavním sponzorem.



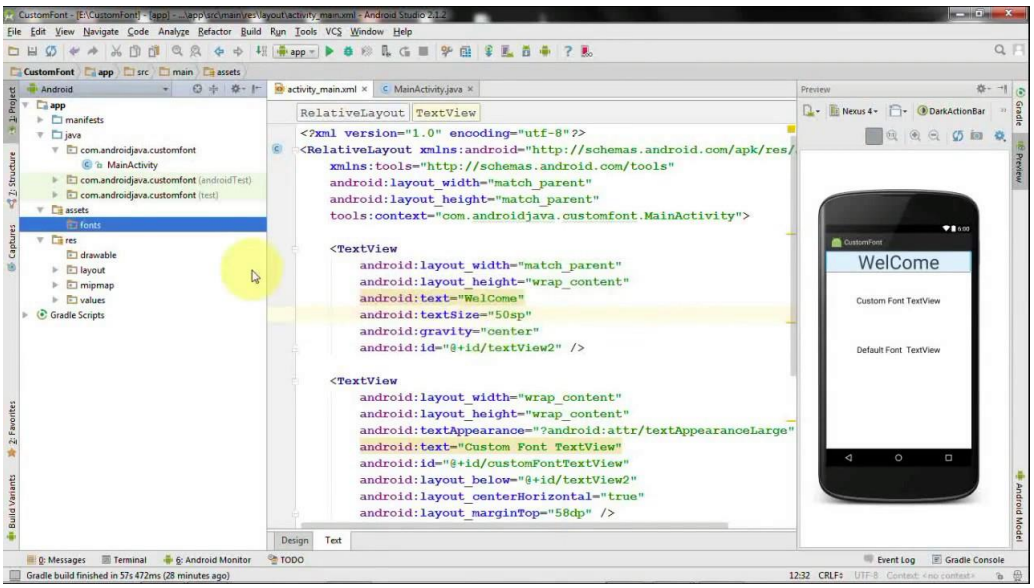
Obrázek 6- Netbeans IDE (33)

[18] IDE – Integrated development environment – integrované programové vybavení pro vývojáře, zaměřené většinou na jeden programovací jazyk (48).

5.4. Android Studio

Android studio je oficiální IDE pro vývoj aplikací pro Android. Oproti klasickým IDE, jako je např. Netbeans, umožňuje spouštět emulátor různých zařízení a různých verzí OS Android – vývojář tedy může svou aplikaci testovat prakticky na všech možných zařízeních (různé velikosti obrazovek, HW konfigurace), která mohou jeho aplikaci využívat, aniž by tato zařízení fyzicky vlastnil.

Do roku 2013 bylo oficiálním prostředím pro vývoj Android aplikací IDE Eclipse s pluginem Android Developer Tools, od tohoto roku ale společnost Google distribuuje vlastní IDE postavené na IntelliJ IDEA.



Obrázek 7 - Android Studio (34)

6. Implementace generátoru náhodných čísel

Jelikož důležitým požadavkem na generátor náhodných čísel pro tuto práci byla jednoduchá rozšiřitelnost mezi velké množství uživatelů, bylo třeba zvolit takovou formu, která by se dala jednoduše, a bez náročných požadavků na instalaci, implementovat do zařízení, která se běžně používají.

Je třeba zohlednit, že běžný uživatel spotřební elektroniky nemá podrobnou znalost elektronické komunikace. Z toho vyplývá, že je málo pravděpodobné, že bude sám vyhledávat možnosti, jak svou komunikaci zabezpečit a ochránit před ostatními. Proto generátor musí být implementován tak, aby co nejméně zatěžoval uživatele, ideálně aby o něm ani nevěděl.

Z tohoto důvodu byl pro tvorbu prototypů zvolen programovací jazyk Java, a to jak ve své desktop verzi, kompatibilní napříč operačními systémy, tak i v mobilní verzi pro operační systém Android.

Druhy prototypů byly zvoleny následně:

- Snímač pohybu smartphone
- Snímač dráhy při ježdění prstem po displeji
- Snímač bodů z kamery smartphone
- Snímač pohybu myši na displeji počítače
- Snímač prodlevy mezi stiskem kláves při psaní na počítači

Tyto prototypy mohou být implementovány pro běh na pozadí, tudíž uživatel není nijak omezován nebo nucen dělat činnosti nad rámec požadovaných, zatímco jsou generována náhodná čísla, která mohou být ve chvíli, kdy jsou potřeba, použita.

6.1. Obecná funkce prototypů

Spolu s jednoduchou rozšiřitelností mezi uživatele je dalším důležitým požadavkem na prototyp i rychlé generování velkého množství dat. Z tohoto důvodu prototypy negenerují pouze jedno číslo pro každý běh algoritmu, ale vždy sled číslic od 0 do 9 tak, že jsou brána data z externího hardwarového zdroje a upravena do potřebné podoby. Jak bude diskutováno především v kapitole Sběr a hodnocení dat, v některých případech tento postup přinesl navýšení množství generovaných čísel, avšak za cenu menší náhodnosti.

S ohledem na skutečnost, že prototypy byly vytvářeny jen za cílem vygenerování čísel, nemají žádné GUI pro uživatele – jen nejnutnější minimum pro správnou funkčnost generátoru.

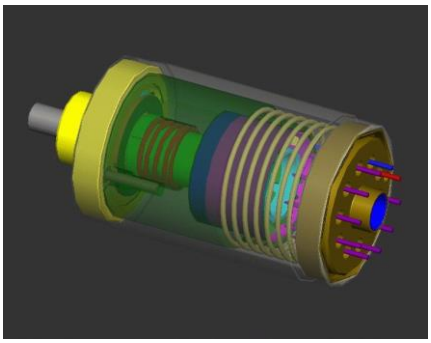
Kvůli možnosti porovnat, jaké množství dat je prototyp schopen vygenerovat za určitou časovou jednotku, všechny prototypy jsou nastaveny tak, aby měřily přesně 60 sekund. V průběhu měření jsou data shromažďována v paměti zařízení. Po uplynutí 60 sekund jsou data uložena do souboru, který je v názvu opatřen časem začátku generování a typem prototypu.

Na skončení měření je uživatel v mobilní aplikaci upozorněn zprávou, desktopová aplikace se ukončí.

6.2. Snímač pohybu smartphone

Tento prototyp byl zvolen z důvodu, že valná většina smartphonů[19], které jsou rozšířeny mezi jejich uživateli, je vybavena senzorem pohybu – takzvaným akcelerometrem[20].

Akcelerometr je součástka, která měří zrychlení. Je navržena tak, že při změně z konstantní nebo nulové rychlosti tuto změnu zaznamená. Akcelerometr využívá mikroskopické krystaly, na kterých se prostřednictvím piezoelektrického jevu z vibrací, způsobených změnou rychlosti, generuje napětí (35).



Obrázek 8 – Akcelerometr (35)

Pomocí akcelerometru tedy poznáme, že je s telefonem pohybováno. Vzhledem k tomu, že pohyb telefonu vykazuje určitou náhodnost – pohybuje se jinak, když ho máme

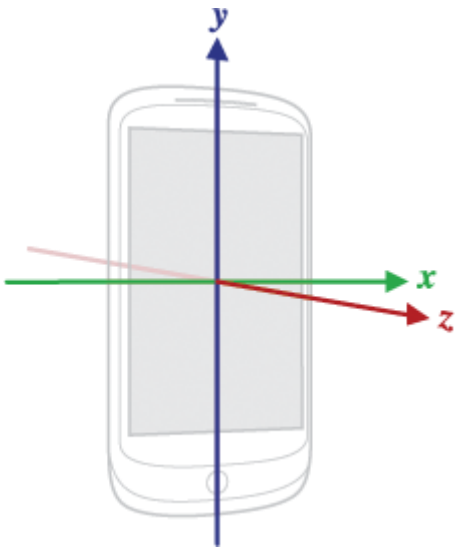
[19] Smartphone nebo také chytrý telefon, je telefon s pokročilými funkcemi. Mezi charakteristické znaky patří otevřený operační systém a možnost instalovat aplikace (40). Smartphone umožňuje kromě volání např. používat mapy, sledovat video, prohlížet internet apod.

[20] Akcelerometr také bývá, často z marketingových důvodů, nazýván jako gravitační čip, gravity sensor, G-senzor apod.

za chůze v kapse, když ho držíme v ruce apod. – jsou data získaná z akcelerometru vhodná k použití v generátoru náhodných čísel jako náhodná složka. Navíc generování čísel může probíhat na pozadí, bez aktivního zapojení uživatele – stačí když uživatel nechá aplikaci generovat čísla, když má telefon v kapse nebo pouzdře. Tím může být nasbírán velký objem náhodných dat pro následné použití – při šifrování již není třeba dělat nic navíc pro to, aby data byla zašifrována. Pouze se použije část již dříve vygenerovaných dat.

Prototyp využívá třídy android.hardware.SensorManager, která nám umožní přístup k senzoru typu akcelerometr. Tento senzor je v Androidu podporován od verze 1.5. Třída SensorManager, zajišťující přístup k senzorům zařízení, nám pomocí zaregistrovaného SensorEventListeneru[21], jemuž předává notifikace o změnách na senzorech, vrací SensorEvent[22], kterou odchytáváme funkcí onSensorChanged. Z instance třídy SensorEvent si pomocí metody values() načteme pole hodnot získaných z akcelerometru ve formátu

- *values[0]* – akcelerace snížená o gravitaci na ose x
- *values[1]* – akcelerace snížená o gravitaci na ose y
- *values[2]* – akcelerace snížená o gravitaci na ose z



Obrázek 9 - Osy akcelerometru (36)

Hodnoty jsou sníženy o gravitaci z toho důvodu, že senzor měří akceleraci vztaženou k zařízení. Z toho důvodu zařízení bez pohybu, např. ležící na stole, měří akceleraci $g = 9{,}81\,\frac{m}{s^2}$. Z toho je zřejmé, že při volném pádu by zařízení měřilo zrychlení o hodnotě $0\,\frac{m}{s^2}$.

[21] Listener – třída, která je jakýmsi „posluchačem“ – hlídá změny hodnot, nebo zachytí akci uživatele a zareaguje tím, že vrátí „událost“ – event.
[22] Event – objekt, který říká, jakou událost, nebo změnu zaznamenal listener. Na tuto událost poté vývojář reaguje nějakou svojí akcí.

Pokud tedy chceme zjistit reálnou akceleraci zařízení, je potřeba gravitační zrychlení zohlednit (37).

Výsledná hodnota, která se zaznamenává jako vygenerované číslo jsou hodnoty z těchto tří indexů pole *values* spojené do řetězce. Protože jsou hodnoty i s desetinnými čísly a zápornými hodnotami, jsou z řetězce odebrány znaky „.“ a „-“.

```
@Override
    public void onSensorChanged(SensorEvent event) {
        if(System.currentTimeMillis() >= startTime + MEASURING_TIME){
            String path = writeToFile(finalValue, "generated_values_move_" +
startTime + ".txt");
            Toast.makeText(MoveActivity.this, "Měření bylo dokončeno. Soubor
s daty najdete zde: " + path, Toast.LENGTH_LONG).show();
        }else{
            float[] values = event.values;
            String value = String.valueOf(values[0]) +
String.valueOf(values[1]) + String.valueOf(values[2]);
            value = value.replace(".", "");
            value = value.replace("-", "");

            finalValue += value;
        }
    }
```

Ukázka kódu 1 – Zpracování hodnot z akcelerometru

6.3. Snímač dráhy při pohybu prstu po displeji

Ačkoliv mají lidé některá gesta při práci se smartphone naučená, je velmi malá pravděpodobnost, že by délka tahu a umístění gesta na displeji a tvar gesta byla vždy stejná.

Tohoto předpokladu je využito u prototypu pro snímání pohybu prstu na displeji. Při každém dotyku jsou snímány souřadnice umístění prstu a zaznamenány do souboru.

K souřadnicím umístění prstu na displeji se dostaneme pomocí registrovaného View.OnTouchListeneru a implementací jeho metody onTouch, která umožňuje přístup k datům prostřednictvím MotionEvent. MotionEvent zprostředkovává přístup k informacím typu ACTION_DOWN (souřadnice, kde bylo gesto zahájeno), ACTION_UP (souřadnice, kde bylo gesto ukončeno) nebo ACTION_CANCEL (souřadnice, kde bylo gesto přerušeno).

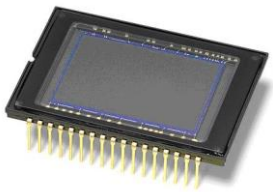
Jelikož pro generování náhodných čísel není potřeba rozlišovat různá stádia gesta, je využito metod getRawX a getRawY, které vracejí hodnoty proměnné AXIS_X, resp. AXIS_Y. Hodnoty těchno proměnných pro dotykové displeje reprezentují absolutní souřadnici X, resp. Y pozice prostředku dotyku na dotykové vrstvě. Jednotky jsou pixely displeje.

```
public boolean onTouch(View view, MotionEvent event){
                if(System.currentTimeMillis() >= startTime +
MEASURING_TIME){
                    String path = writeToFile(finalValue,
"generated_values_touch_" + startTime + ".txt");
                    Toast.makeText(DotekActivity.this, "Měření bylo
dokončeno. Soubor s daty najdete zde: " + path,
Toast.LENGTH_LONG).show();
                }else{
                    String generatedValue =
String.valueOf(event.getRawX()).replace(".", "") +
String.valueOf(event.getRawY()).replace(".", "");
                    finalValue += generatedValue;
                }
                return true;
                  }
```

Ukázka kódu 2 – Zpracování hodnot z dotykového displeje

6.4. Snímač bodů z kamery smartphone

Většina fotoaparátů obsažených v současných smartphonech je osazena CMOS nebo CCD snímači pro snímání obrazu. Ačkoliv se liší cenou, způsobem zpracování výsledného obrazu a výrobními technologiemi, pro potřeby generátoru náhodných čísel je možno tyto rozdíly ignorovat.



Obrázek 10 - CCD snímač (38)

Každý snímač se skládá z velkého množství světlocitlivých křemíkových buněk. Při osvětlení těchto buněk se uvolní z krystalové mřížky volný elektron. Množství uvolněných elektronů je odpovídající intenzitě dopadajícího světla.

Pro rozlišení barvy dopadajícího světla je před buňku je před ní umístěn filtr příslušné barvy. Ve výsledku je tedy získána informace, jaké světlo a jaké intenzity dopadá na buňku snímače.

Tyto informace jsou vyvedeny do elektronických obvodů, kde se elektrický signál pomocí A/D převodníků převede do digitální podoby.

S touto digitální podobou už je schopen pracovat operační systém. Přístup k datům z kamery smartphone umožňuje třída CameraManager, která zprostředkovává komunikaci se všemi kamerami připojenými k mobilnímu telefonu.

Pomocí třídy CaptureRequest.Builder zachytíme aktuální obraz, která nám předává snímač ve fotoaparátu a předáme je třídě ImageReader. Tato třída umožňuje přímý přístup aplikace k binárním datům zachyceného obrázku. Následně z těchto dat vytvoříme hashkód, který využijeme jako náhodná čísla.

Obrázek 11 - CMOS snímač (38)

Je vycházeno z předpokladu, že žádné 2 obrazy zachycené bezprostředně po sobě nebudou stejné. I kdyby se totiž snímaná scéna nezměnila, technická nedokonalost snímačů vnáší do obrazu šum a ten samotný zajistí, že obraz nebude stejný. Proto i kdyby byl obraz snímán v naprosté tmě, výsledná data budou šumem ovlivněna.

```
public void onImageAvailable(ImageReader reader) {
                    Image = null;

                    image = reader.acquireLatestImage();
                    ByteBuffer buffer = image.getPlanes()[0].getBuffer();
                    byte[] bytes = new byte[buffer.capacity()];
                    buffer.get(bytes);

                    String data = String.valueOf(bytes.hashCode());
                    finalValue += data;

                    Log.i("FINALVALUE", finalValue);
                      }
```

Ukázka kódu 3 – Zpracování hodnot z kamery

6.5. Snímač pohybu myši na displeji počítače

Stejně jako je pohyb prstu na dotykovém displeji nezopakovatelný, také pohyb myši se nedá přesně replikovat. A protože naprostá většina počítačů má připojenou myš anebo trackball k ovládání, dá se tento prototyp snadno implementovat a použít.

A stejně jako má Android podporu pro snímání dotyku, je samozřejmé, že i jazyk Java pro desktop má podporu pro snímání polohy myši.

Prostřednictvím registrovaného MouseMotionListener je implementována metoda mouseMoved v které se přistupuje k datům MouseEvent. Stejně jako u MotionEvent v Android SDK, je i zde možnost zjistit, kdy a jaké tlačítko bylo stlačeno, puštěno apod.,

pro potřeby prototypu ale postačí informace o souřadnicích ukazatele myši, které se zjistí zavoláním getPoint (třída java.awt.Point) a přečtením veřejné proměnné x, resp. y.

```
public void mouseMoved(MouseEvent e) {
                        if (System.currentTimeMillis() >= startTime +
MEASURING_TIME) {
                            String path;
                            try {
                                path = writeToFile(finalValue,
"generated_values_mouse_" + startTime + ".txt");
                            } catch (Exception ex) {

Logger.getLogger(Frame.class.getName()).log(Level.SEVERE, null, ex);
                            }


Logger.getLogger(Frame.class.getName()).log(Level.INFO, "Mereni
dokonceno");
                            System.exit(1);
                        } else {
                            finalValue += e.getPoint().x +
e.getPoint().y;

Logger.getLogger(Frame.class.getName()).log(Level.INFO, "Finalvalue " +
finalValue);
                        }
                          }
```

Ukázka kódu 4 – Zpracování hodnot z pohybu myši

6.6. Snímač prodlevy mezi stiskem kláves při psaní na počítači

Obdobně, jako je u většiny počítačů připojena myš, je k nim také připojena klávesnice pro zadávání znaků. Pro účely této práce byla vybrána metoda snímání prodlevy mezi stiskem kláves – ať už je psáno pomalu několika prsty, nebo deseti, není prodleva mezí jednotlivými stisky stejná a vykazuje určitou míru náhodnosti.

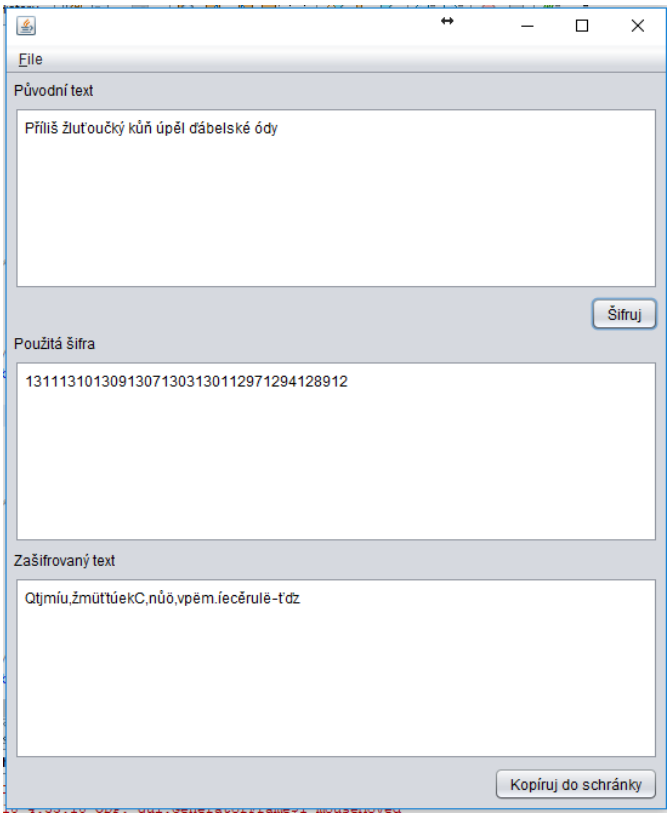
Java s klávesnicí pracuje na podobném principu jako s myší, tzn. Prostřednictvím KeyListener je implementována metoda keyPressed. V této metodě je zjištěn čas stisku a od něj je odečten čas stisku předchozího. Rozdíl je zapsán jako náhodné číslo.

```
public void keyPressed(KeyEvent e) {
                        if (System.currentTimeMillis() >= startTime +
MEASURING_TIME) {
                            String path;
                            try {
                                path = writeToFile(finalValue,
"generated_values_keyboard_" + startTime + ".txt");
                            } catch (Exception ex) {

Logger.getLogger(Frame.class.getName()).log(Level.SEVERE, null, ex);
                            }


Logger.getLogger(Frame.class.getName()).log(Level.INFO, "Mereni
dokonceno");
                            System.exit(1);
                        } else {
                            long time = System.currentTimeMillis();
                            long timeBetweenPress = time -
lastPressedTime;
                            lastPressedTime = time;

                            finalValue += timeBetweenPress;

Logger.getLogger(Frame.class.getName()).log(Level.INFO, "Finalvalue " +
finalValue);
                        }
                      }
```

Ukázka kódu 5 – Zpracování hodnot ze stisku kláves

7. Implementace Vernamovy šifry

Pro názornou ukázku využití generátoru náhodných čísel byla zvolena Vernamova šifra (viz kapitola 3.4.1.6) pro implementaci společně s generátorem náhodných čísel pomocí pohybu myší po obrazovce.



Obrázek 12 - Hlavní okno aplikace

Do hlavního okna se do pole Původní text zadá text, který má být šifrován. Po kliknutí na tlačítko Šifruj se otevře celoobrazovkové okno, které snímá pohyb ukazatele myši. Ve chvíli, kdy je vygenerován dostatek náhodných dat pro šifrování se okno zavře a v hlavním okně programu se vyplní pole Použitá šifra a Zašifrovaný text.

Protože je Vernamova šifra šifrou symetrickou, je třeba distribuovat jak zašifrovaný text, tak i samotnou šifru. Volba vhodného způsobu distribuce šifry závisí na podmínkách, v kterých se šifrování používá, to však není obsahem této implementace, stejně tak jako dešifrování zprávy.

Pro implementaci šifry byla zvolena abeceda

```
char[] alphabet =
"aáäbcčdďeéěëfghiíjklmnňoóöpqrřsštťuúůüvwxyýzžAÁÄBCČDĎEÉĚËFGHIÍJKLMNŇOÓÖP
QRŘSŠTŤUÚŮÜVWXYÝZŽ1234567890 ,.?!_+--
*:/=%@#&\\[]()°;|\n„“\"\'".toCharArray();
```

Ukázka kódu 6 – Abeceda pro Vernamovu šifru

a samotné šifrování textu probíhá tak, že se vezme první znak šifrovaného textu a první znak šifry, tedy podle Obrázek 12 písmeno P a číslo 1 a na referenční abecedě alphabet se vyhledá odpovídající písmeno (tedy na pozici P + 1 znak = Q), které je již šifrovaným textem. Takto se postupuje až do posledního znaku šifrovaného textu.

```
public void encodeText(String cipher, String originalText) throws
Exception {
        if (cipher.length() == originalText.length()) {
            char[] alphabet = createAlphabet();
            char[] originalTextArr = originalText.toCharArray();
            String encodedText = "";
            for (int i = 0; i < originalTextArr.length; i++) {
                char character = originalTextArr[i];
                int alphIndex = getIndexOf(character, alphabet);
                int encIndex = alphIndex +
Character.getNumericValue(cipher.charAt(i));
                if (encIndex > (alphabet.length - 1)) {
                    encIndex -= (alphabet.length);
                }
                String newChar = Character.toString(alphabet[encIndex]);
                encodedText += newChar;
            }

            cipherArea.setText(cipher);
            encodedTextArea.setText(encodedText);

            copyButton.setEnabled(true);
            cipherArea.setEnabled(true);
            encodedTextArea.setEnabled(true);
        } else {
            throw new Exception("Cipher must be as long as original
text");
        }
    }
```

Ukázka kódu 7 – Šifrování pomocí Vernamovy šifry

Dešifrování textu při znalosti cifry by bylo obdobné, akorát by posun polí v abecedě nebyl doprava, ale doleva.

8. Sběr a hodnocení dat

8.1. Metodika sběru a hodnocení

Na každém prototypu generátoru náhodných čísel byla provedena tři měření (získaná data jsou na přiloženém disku v elektronické podobě) po dobu 60 sekund, aby bylo možno porovnat, zda se data generují skutečně náhodně a nedochází k předpověditelnému generování posloupností a pro zjištění množství generovaných dat.

Generátory využívající klávesnici a myš byly spuštěny na běžném PC (procesor Intel Core i5-3470 3,20 GHz, paměť RAM 8 GB, 64 bitový operační systém Windows 10, běžná bezdrátová myš a klávesnice). Generátory využívající dotykovou vrstvu, pohyb mobilního telefonu a kameru byly spuštěny na telefonu Vodafone Smart Platinum 7 (procesor Qualcomm Snapdragon 652, paměť RAM 3 GB, fotoaparát 16 MPx, 64 bitový operační systém Android 6).

Následně na datech bylo provedeno hodnocení z hlediska rychlosti generování dat, entropie informace získané z generovaných dat a náhodnosti pomocí Kolmogorov-Smirnovova testu.

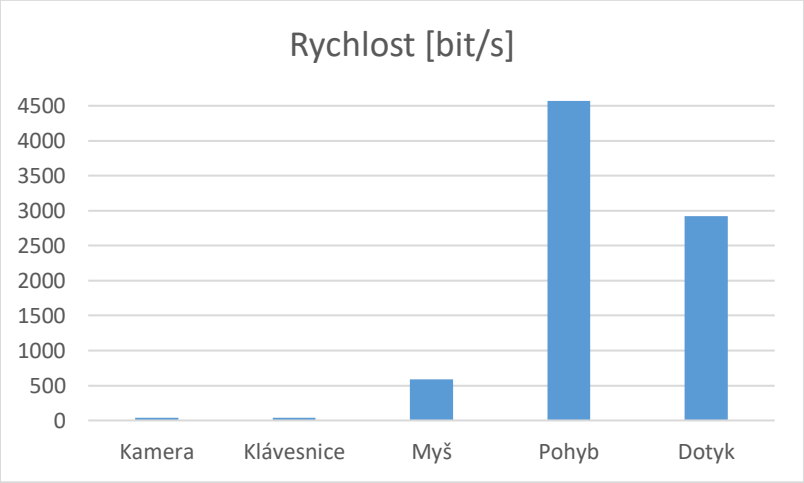
8.2. Rychlost generování dat

Tabulka 2 obsahuje množství vygenerovaných čísel za 1 minutu a počet bitů vygenerovaných za 1 sekundu. Rychlost byla vypočtena na základě převodu dekadického čísla na binární – na zakódování nejvyšší číslice 9 je jsou potřeba 4 bity.

Z prvního pohledu na Graf 1 je zřejmé, že zdaleka nejvíce, a to průměrně 68501 znaků, generuje prototyp generátoru založený na pohybu mobilního telefonu. Zajímavou rychlost má i generátor založený na dotyku displeje – průměrně 4567 bit/s. Naopak nejhorší z hlediska rychlosti generování je generátor využívající kameru telefonu. Průměrná rychlost 35 bit/s je velmi pravděpodobně způsobena nedostatečným výpočetním výkonem procesoru mobilního telefonu, jelikož se v reálném čase vypočítává hash obrazu, což je samozřejmě velmi náročné na zmíněný výkon procesoru.

Generátor	Číslo měření	Čísel za 1 min	bit/s
Kamera	1	564	38
	2	466	31
	3	567	38
	průměr	**532**	**35**
Klávesnice	1	915	61
	2	577	38
	3	389	26
	průměr	**627**	**42**
Myš	1	8266	551
	2	9060	604
	3	9027	602
	průměr	**8784**	**586**
Pohyb	1	68299	4553
	2	68423	4562
	3	68782	4585
	průměr	**68501**	**4567**
Dotyk	1	36653	2444
	2	47597	3173
	3	47251	3150
	průměr	**43833**	**2922**

Tabulka 2 – Rychlost generování dat



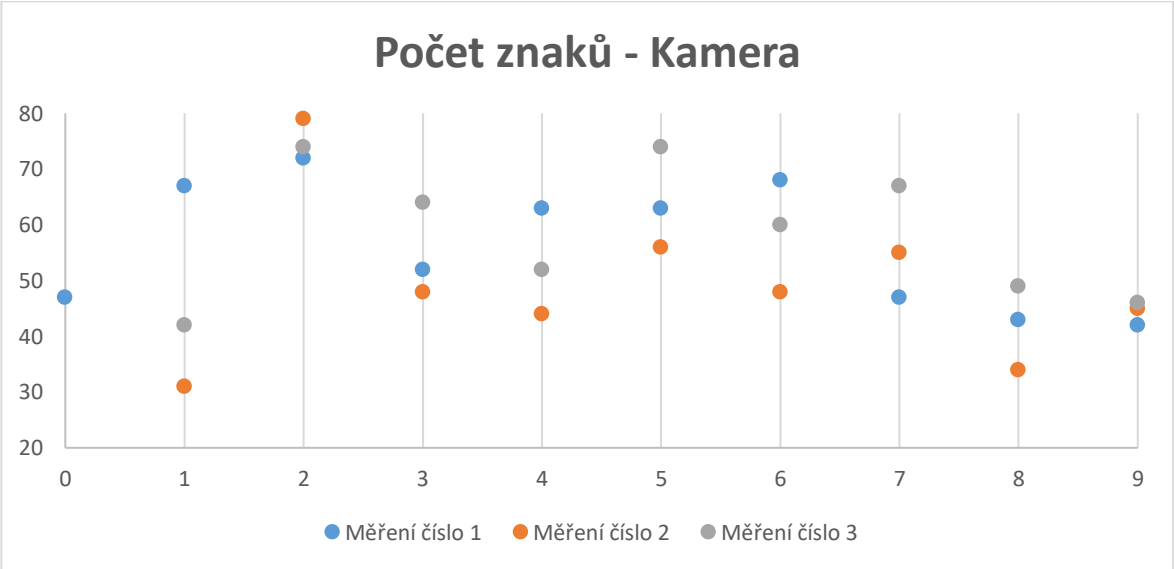
Graf 1 - Rychlost generování dat

8.3. Počet jednotlivých znaků

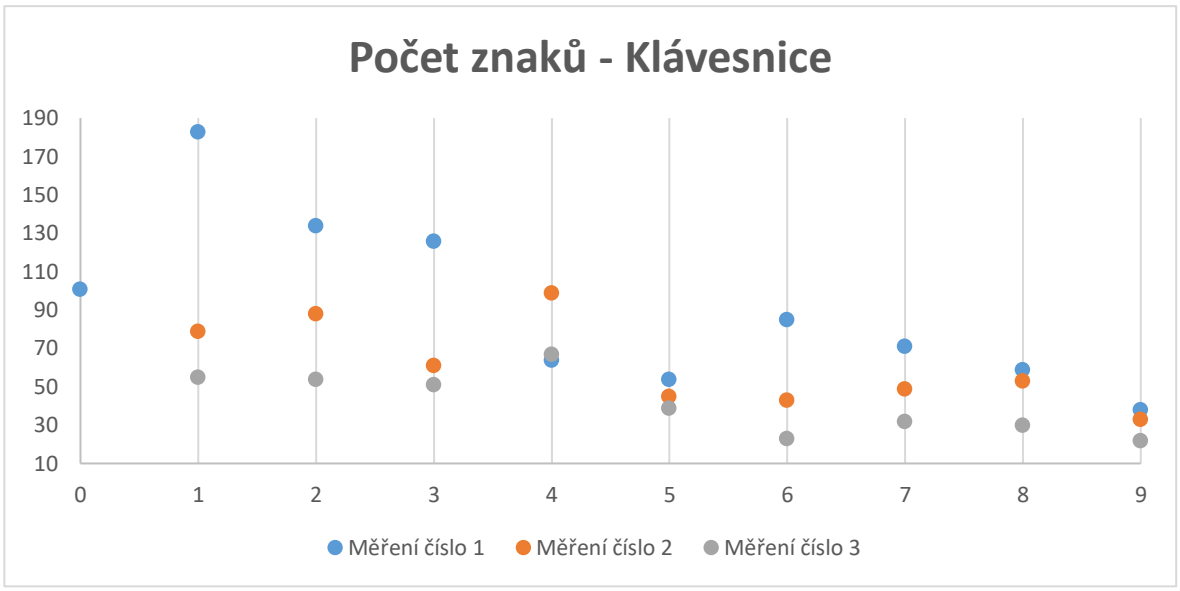
Grafy v této kapitole zobrazují porovnání jednotlivých měření na generátorech náhodných čísel. Na těchto grafech je možné pozorovat, zda jsou na sobě výsledky měření nezávislé, nebo jestli se objevuje určitá pravidla v množství vygenerovaných čísel.

Například Graf 4 jasně ukazuje, že generátor využívající pohyb myši na monitoru generuje podstatně více znaků 1, 2 a 3 než ostatních. Generátor využívající pohyb telefonu (Graf 5) zase generuje výrazně méně znak 9, generátor využívající dotyk displeje (Graf 6) zase často opakuje znak 2.

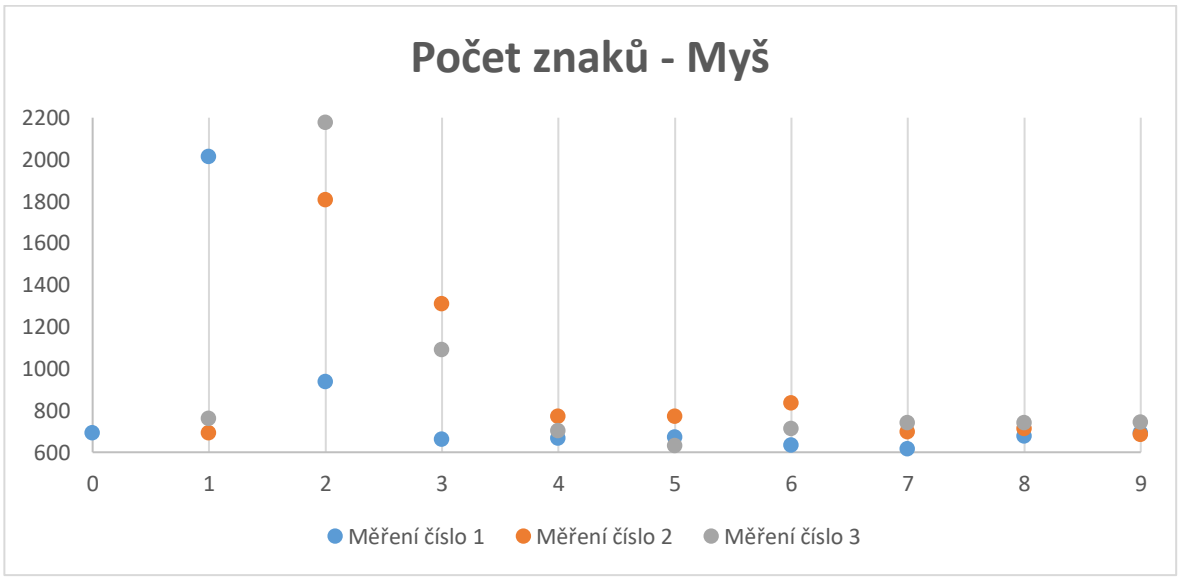
Tyto anomálie budou zřejmě způsobeny špatnou volbou odečtu hodnot využitých pro generování čísel, kdy se například u generátoru založeném na pohybu myši často objevuje téměř vždy na začátku znak 1 nebo 2. Stejně tak generátor využívající dotyk displeje – zde je použita stejná metoda, jako u myši.



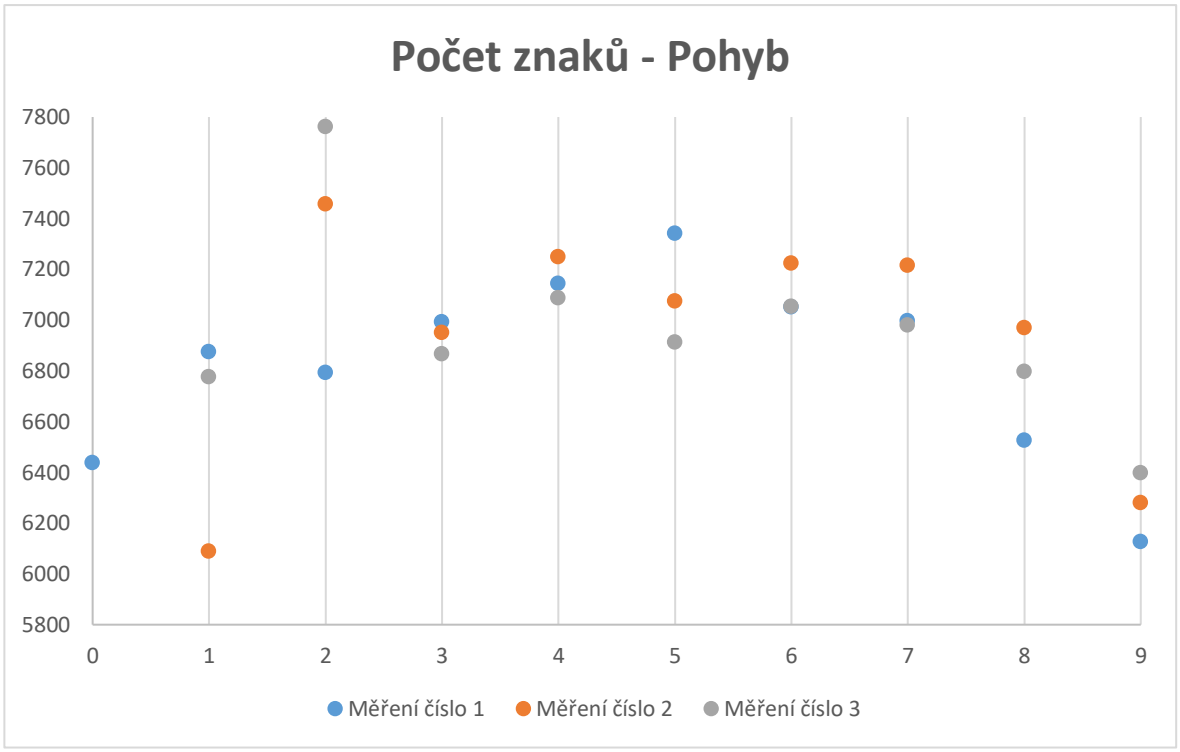
Graf 2 - Počet znaků – Kamera



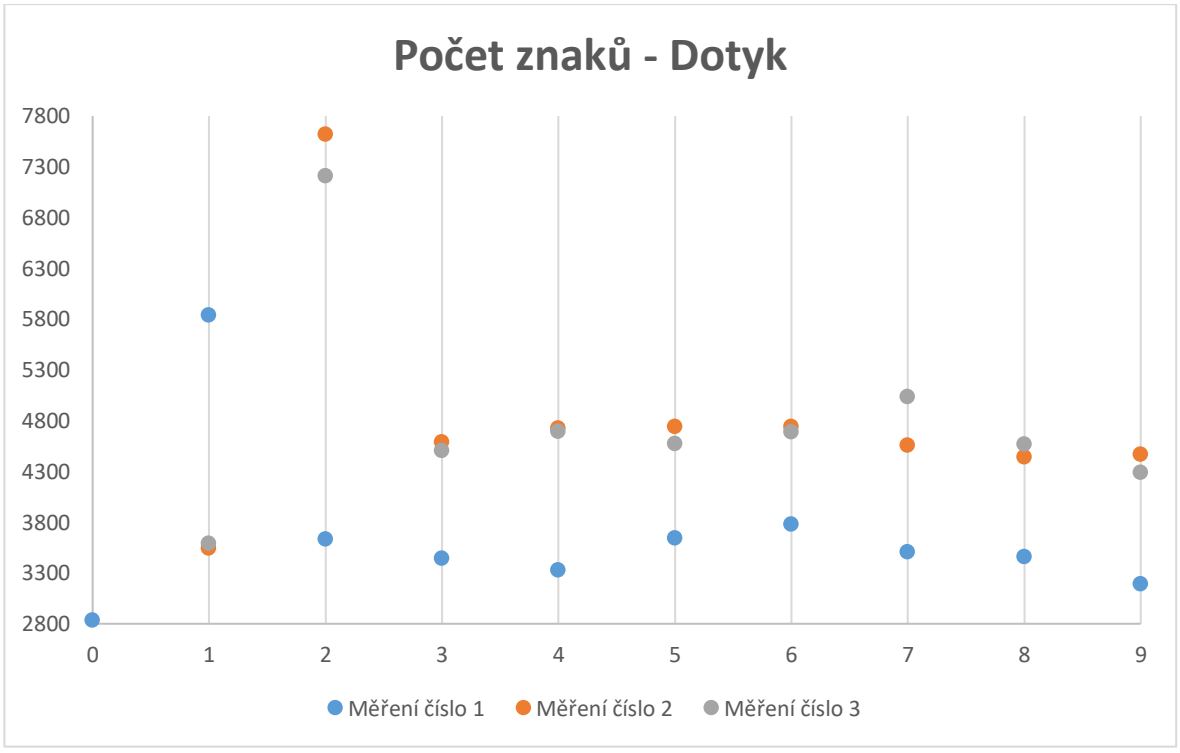
Graf 3 - Počet znaků – Klávesnice



Graf 4 - Počet znaků – Myš



Graf 5 – Počet znaků - Pohyb



Graf 6 - Počet znaků – Dotyk

8.4. Entropie informace

Entropie informace byla vybrána z důvodu možnosti číselně vyjádřit, jak náročné je odhadnout číslo, které bude v řetězci následovat. Pokud je složité další znak odhadnout, dodává nám jeho zjištění větší informaci, než kdybychom věděli, jaké další číslo přijde. Míru této informace vyjadřuje právě entropie informace. Znamená to tedy, že čím vyšší nám entropie vyjde, tím větší míru informace vygenerované znaky přenáší a tím je tedy složitější odhadnout, jaké bude další číslo v naší vygenerované posloupnosti.

Pro výpočet entropie byl využit vzorec (3.4) z kapitoly 3.5.2 Entropie informace a v následujících tabulkách je rozdělen na více částí pro přehlednost.

Jak zobrazuje Graf 7, nejvyšší průměrnou entropii informace má generátor založený na měření časových prodlev mezi stisky kláves. Druhá v pořadí je posloupnost znaků vygenerovaná prototypem využívajícím pohyb myši. Naopak nejnižší entropii má generátor využívající pohyb mobilního telefonu.

Spolu s pohybem telefonu má generátor využívající kameru v rámci jednotlivých měření stabilní entropii informace, zatímco u jiných prototypů entropie různě kolísá.

znak		0	1	2	3	4	5	6	7	8	9
počet	1	47	67	72	52	63	63	68	47	43	42
	2	31	79	48	44	56	48	55	34	45	26
	3	42	74	64	52	74	60	67	49	46	39
P(x) [%]	1	8,33	11,88	12,77	9,22	11,17	11,17	12,06	8,33	7,62	7,45
	2	6,65	16,95	10,30	9,44	12,02	10,30	11,80	7,30	9,66	5,58
	3	7,41	13,05	11,29	9,17	13,05	10,58	11,82	8,64	8,11	6,88
log P(x)	1	-1,08	-0,93	-0,89	-1,04	-0,95	-0,95	-0,92	-1,08	-1,12	-1,13
	2	-1,18	-0,77	-0,99	-1,02	-0,92	-0,99	-0,93	-1,14	-1,02	-1,25
	3	-1,13	-0,88	-0,95	-1,04	-0,88	-0,98	-0,93	-1,06	-1,09	-1,16
H(X)	**1**	**10,08**									
	2	**10,20**									
	3	**10,10**									

Tabulka 3 - Entropie informace - Kamera

znak		0	1	2	3	4	5	6	7	8	9
počet	1	101	183	134	126	64	54	85	71	59	38
	2	79	88	61	99	45	43	49	53	33	27
	3	55	54	51	67	39	23	32	30	22	16

P(x) [%]	1	11,04	20,00	14,64	13,77	6,99	5,90	9,29	7,76	6,45	4,15
	2	13,69	15,25	10,57	17,16	7,80	7,45	8,49	9,19	5,72	4,68
	3	14,14	13,88	13,11	17,22	10,03	5,91	8,23	7,71	5,66	4,11
log P(x)	1	-0,96	-0,70	-0,83	-0,86	-1,16	-1,23	-1,03	-1,11	-1,19	-1,38
	2	-0,86	-0,82	-0,98	-0,77	-1,11	-1,13	-1,07	-1,04	-1,24	-1,33
	3	-0,85	-0,86	-0,88	-0,76	-1,00	-1,23	-1,08	-1,11	-1,25	-1,39
H(X)	**1**	**10,45**									
	2	**10,34**									
	3	**10,41**									

Tabulka 4 - Entropie informace - Klávesnice

znak		0	1	2	3	4	5	6	7	8	9
počet	1	692	2015	938	661	667	672	634	616	678	693
	2	693	1807	1310	771	772	836	698	714	686	773
	3	762	2177	1091	702	632	712	741	741	745	724
P(x) [%]	1	8,37	24,38	11,35	8,00	8,07	8,13	7,67	7,45	8,20	8,38
	2	7,65	19,94	14,46	8,51	8,52	9,23	7,70	7,88	7,57	8,53
	3	8,44	24,12	12,09	7,78	7,00	7,89	8,21	8,21	8,25	8,02
log P(x)	1	-1,08	-0,61	-0,95	-1,10	-1,09	-1,09	-1,12	-1,13	-1,09	-1,08
	2	-1,12	-0,70	-0,84	-1,07	-1,07	-1,03	-1,11	-1,10	-1,12	-1,07
	3	-1,07	-0,62	-0,92	-1,11	-1,15	-1,10	-1,09	-1,09	-1,08	-1,10
H(X)	**1**	**10,32**									
	2	**10,24**									
	3	**10,33**									

Tabulka 5 - Entropie informace - Myš

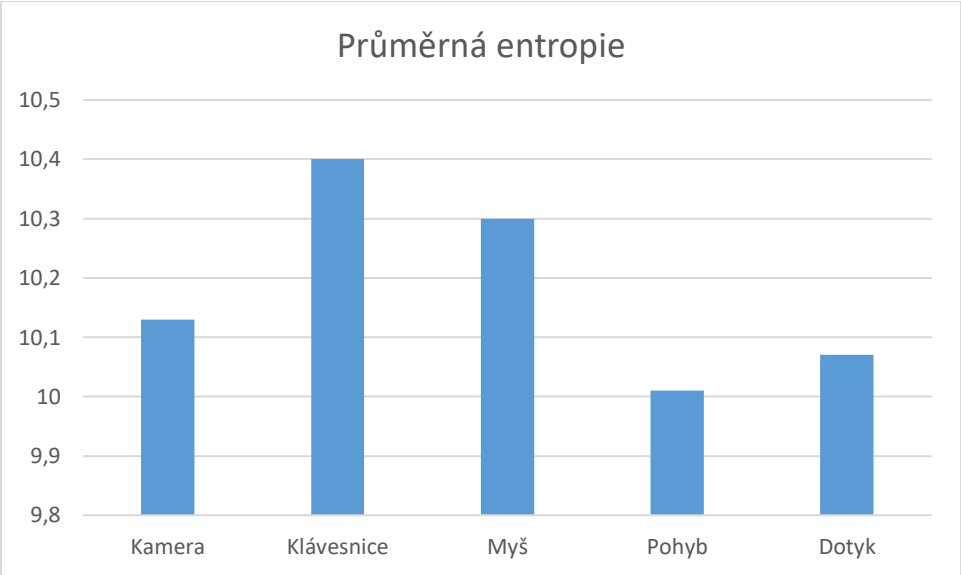
znak		0	1	2	3	4	5	6	7	8	9
počet	1	6440	6876	6795	6993	7146	7342	7053	6998	6528	6128
	2	6090	7458	6952	7251	7076	7225	7217	6970	6281	5903
	3	6778	7762	6867	7088	6914	7055	6981	6798	6399	6140
P(x) [%]	1	9,43	10,07	9,95	10,24	10,46	10,75	10,33	10,25	9,56	8,97
	2	8,90	10,90	10,16	10,60	10,34	10,56	10,55	10,19	9,18	8,63
	3	9,85	11,28	9,98	10,31	10,05	10,26	10,15	9,88	9,30	8,93
log P(x)	1	-1,03	-1,00	-1,00	-0,99	-0,98	-0,97	-0,99	-0,99	-1,02	-1,05
	2	-1,05	-0,96	-0,99	-0,97	-0,99	-0,98	-0,98	-0,99	-1,04	-1,06
	3	-1,01	-0,95	-1,00	-0,99	-1,00	-0,99	-0,99	-1,01	-1,03	-1,05
H(X)	**1**	**10,01**									
	2	**10,01**									
	3	**10,01**									

Tabulka 6 - Entropie informace - Pohyb

znak		0	1	2	3	4	5	6	7	8	9
počet	1	2834	5840	3633	3444	3329	3642	3780	3505	3457	3189
	2	3542	7618	4589	4726	4742	4743	4558	4441	4465	4173

	3	3589	7212	4503	4693	4575	4689	5033	4569	4291	4097
P(x) [%]	1	7,73	15,93	9,91	9,40	9,08	9,94	10,31	9,56	9,43	8,70
	2	7,44	16,01	9,64	9,93	9,96	9,96	9,58	9,33	9,38	8,77
	3	7,60	15,26	9,53	9,93	9,68	9,92	10,65	9,67	9,08	8,67
log P(x)	1	-1,11	-0,80	-1,00	-1,03	-1,04	-1,00	-0,99	-1,02	-1,03	-1,06
	2	-1,13	-0,80	-1,02	-1,00	-1,00	-1,00	-1,02	-1,03	-1,03	-1,06
	3	-1,12	-0,82	-1,02	-1,00	-1,01	-1,00	-0,97	-1,01	-1,04	-1,06
H(X)	**1**	**10,08**									
	2	**10,08**									
	3	**10,07**									

Tabulka 7 - Entropie informace – Dotyk



Graf 7 - Průměrná entropie informace

8.5. Náhodnost

Pro ověření náhodnosti čísel v generovaných posloupnostech byl vybrán Kolmogorov-Smirnovův test. Ten porovnává teoretickou a naměřenou distribuční funkci. V našem případě tedy porovnává rovnoměrné rozdělení definované vzorcem (3.7) v kapitole 3.5.3 Kolmogorov-Smirnovův test a náhodná výběr definovaný vzorcem (3.6) téže kapitoly.

Velké odchylky mezi těmito funkcemi svědčí o tom, že rozdíl mezi modelovými a generovanými hodnotami není způsoben jen náhodnými vlivy.

Testujeme tedy nulovou hypotézu, že se tyto funkce rovnají, a to na hladině významnosti 0,05. Kritickou hodnotu jsme spočetli pomocí vzorce (3.11) zmíněné kapitoly.

Nulová hypotéza nebyla zamítnuta jenom u 3. měření na generátoru s pohybem mobilního telefonu. Pro všechna ostatní byla nulová hypotéza zamítnuta.

Porovnání kritické hodnoty s testovou statistikou zobrazuje Graf 8, na kterém je zřejmé několikanásobné překročení kritické hodnoty pro generátory využívající pohyb myši a prodlevu mezi stiskem kláves.

Srovnání průměrných hodnot překročení kritické hodnoty zobrazuje Graf 9. Tento graf potvrzuje téměř 8 násobné překročení mezní hodnoty pro pohyb myší, naopak minimální překročení metodou snímání obrazu kamerou. Také pohyb telefonu vykazuje menší překročení, než ostatní metody.

$<u_j;u_{j+1})$		(-1;0>	(0;1>	(1;2>	(2;3>	(3;4>	(4;5>	(5;6>	(6;7>	(7;8>	(8;9>
x		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
n_j	1	47	67	72	52	63	63	68	47	43	42
	2	31	79	48	44	56	48	55	34	45	26
	3	42	74	64	52	74	60	67	49	46	39
N_j	1	47	114	186	238	301	364	432	479	522	564
	2	31	110	158	202	258	306	361	395	440	466
	3	42	116	180	232	306	366	433	482	528	567
$\Phi(x)$	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	2	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	3	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$F_n(x)$	1	0,0833	0,2021	0,3298	0,4220	0,5337	0,6454	0,7660	0,8493	0,9255	0,0833
	2	0,0665	0,2361	0,3391	0,4335	0,5536	0,6567	0,7747	0,8476	0,9442	0,0665
	3	0,0741	0,2046	0,3175	0,4092	0,5397	0,6455	0,7637	0,8501	0,9312	0,0741
$\lvert\Phi(x)-F_n(x)\rvert$	1	0,0167	0,0021	0,0298	0,0220	0,0337	0,0454	0,0660	0,0493	0,0255	0,0167
	2	0,0335	0,0361	0,0391	0,0335	0,0536	0,0567	0,0747	0,0476	0,0442	0,0335
	3	0,0259	0,0046	0,0175	0,0092	0,0397	0,0455	0,0637	0,0501	0,0312	0,0259

$D_{n,\alpha}$	1	0,0572	D_n	0,0660	**zamít. H_0?**	ANO
	2	0,0618		0,0747		ANO
	3	0,0554		0,0637		ANO

Tabulka 8 - Kolmogorov-Smirnovův test – Kamera

$<u_j;u_{j+1})$		(-1;0>	(0;1>	(1;2>	(2;3>	(3;4>	(4;5>	(5;6>	(6;7>	(7;8>	(8;9>
x		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
n_j	1	101	183	134	126	64	54	85	71	59	38
	2	79	88	61	99	45	43	49	53	33	27
	3	55	54	51	67	39	23	32	30	22	16
N_j	1	101	284	418	544	608	662	747	818	877	915
	2	79	167	228	327	372	415	464	517	550	577
	3	55	109	160	227	266	289	321	351	373	389
$\Phi(x)$	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	2	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1

	3	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$F_n(x)$	1	0,1104	0,3104	0,4568	0,5945	0,6645	0,7235	0,8164	0,8940	0,9585	1
	2	0,1369	0,2894	0,3951	0,5667	0,6447	0,7192	0,8042	0,8960	0,9532	1
	3	0,1414	0,2802	0,4113	0,5835	0,6838	0,7429	0,8252	0,9023	0,9589	1
$\|\Phi(x)-F_n(x)\|$	1	0,0104	0,1104	0,1568	0,1945	0,1645	0,1235	0,1164	0,0940	0,0585	0
	2	0,0369	0,0894	0,0951	0,1667	0,1447	0,1192	0,1042	0,0960	0,0532	0
	3	0,0414	0,0802	0,1113	0,1835	0,1838	0,1429	0,1252	0,1023	0,0589	0

$D_{n,\alpha}$	1	0,0449	**D_n**	0,1945	**zamít. H_0?**	ANO
	2	0,0574		0,1667		ANO
	3	0,0676		0,1838		ANO

Tabulka 9 - Kolmogorov-Smirnovův test – Klávesnice

$<u_j;u_{j+1})$		**(-1;0>**	**(0;1>**	**(1;2>**	**(2;3>**	**(3;4>**	**(4;5>**	**(5;6>**	**(6;7>**	**(7;8>**	**(8;9>**
x		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
n_j	1	692	2015	938	661	667	672	634	616	678	693
	2	693	1807	1310	771	772	836	698	714	686	773
	3	762	2177	1091	702	632	712	741	741	745	724
N_j	1	692	2707	3645	4306	4973	5645	6279	6895	7573	8266
	2	693	2500	3810	4581	5353	6189	6887	7601	8287	9060
	3	762	2939	4030	4732	5364	6076	6817	7558	8303	9027
$\Phi(x)$	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	2	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	3	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$F_n(x)$	1	0,0837	0,3275	0,4410	0,5209	0,6016	0,6829	0,7596	0,8341	0,9162	1
	2	0,0765	0,2759	0,4205	0,5056	0,5908	0,6831	0,7602	0,8390	0,9147	1
	3	0,0844	0,3256	0,4464	0,5242	0,5942	0,6731	0,7552	0,8373	0,9198	1
$\|\Phi(x)-F_n(x)\|$	1	0,0163	0,1275	0,1410	0,1209	0,1016	0,0829	0,0596	0,0341	0,0162	0
	2	0,0235	0,0759	0,1205	0,1056	0,0908	0,0831	0,0602	0,0390	0,0147	0
	3	0,0156	0,1256	0,1464	0,1242	0,0942	0,0731	0,0552	0,0373	0,0198	0

$D_{n,\alpha}$	1	0,0449	**D_n**	0,1945	**zamít. H_0?**	ANO
	2	0,0574		0,1667		ANO
	3	0,0676		0,1838		ANO

Tabulka 10 - Kolmogorov-Smirnovův test – Myš

$<u_j;u_{j+1})$		**(-1;0>**	**(0;1>**	**(1;2>**	**(2;3>**	**(3;4>**	**(4;5>**	**(5;6>**	**(6;7>**	**(7;8>**	**(8;9>**
x		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
n_j	1	6440	6876	6795	6993	7146	7342	7053	6998	6528	6128
	2	6090	7458	6952	7251	7076	7225	7217	6970	6281	5903
	3	6778	7762	6867	7088	6914	7055	6981	6798	6399	6140
N_j	1	6440	13316	20111	27104	34250	41592	48645	55643	62171	68299
	2	6090	13548	20500	27751	34827	42052	49269	56239	62520	68423
	3	6778	14540	21407	28495	35409	42464	49445	56243	62642	68782

Φ(x)	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	2	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	3	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$F_n(x)$	1	0,0943	0,1950	0,2945	0,3968	0,5015	0,6090	0,7122	0,8147	0,9103	1
	2	0,0890	0,1980	0,2996	0,4056	0,5090	0,6146	0,7201	0,8219	0,9137	1
	3	0,0985	0,2114	0,3112	0,4143	0,5148	0,6174	0,7189	0,8177	0,9107	1
$\lvert\Phi(x)-F_n(x)\rvert$	1	0,0057	0,0050	0,0055	0,0032	0,0015	0,0090	0,0122	0,0147	0,0103	0
	2	0,0110	0,0020	0,0004	0,0056	0,0090	0,0146	0,0201	0,0219	0,0137	0
	3	0,0015	0,0114	0,0112	0,0143	0,0148	0,0174	0,0189	0,0177	0,0107	0

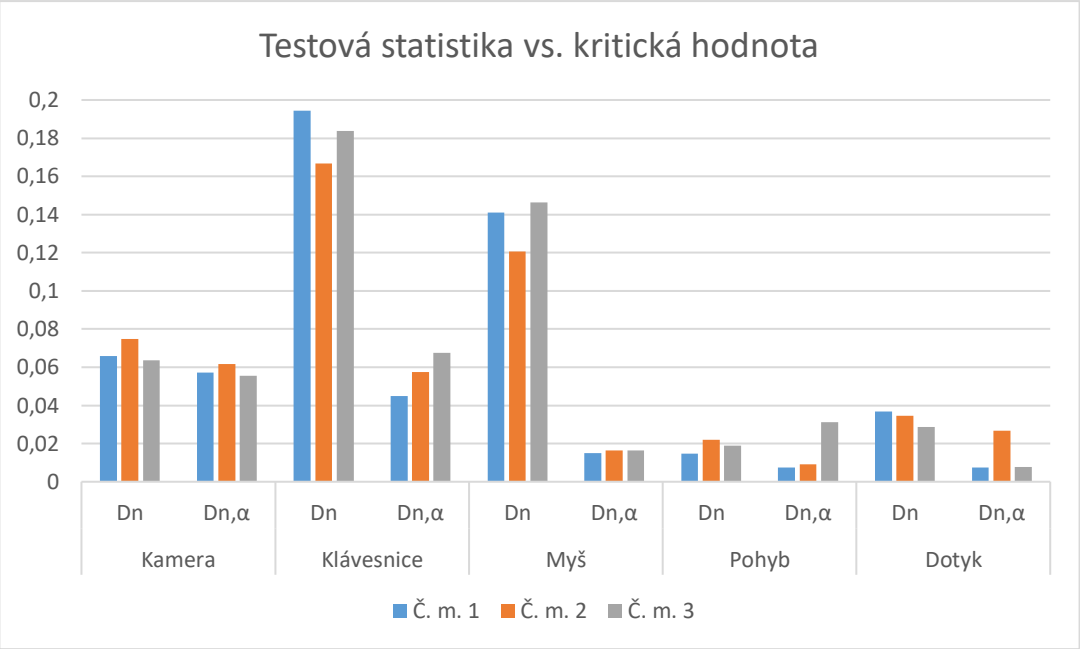
$D_{n,\alpha}$	1	0,0075	**D_n**	0,0147	**zamít. H_0?**	ANO
	2	0,0092		0,0219		ANO
	3	0,0311		0,0189		NE

Tabulka 11 - Kolmogorov-Smirnovův test – Pohyb

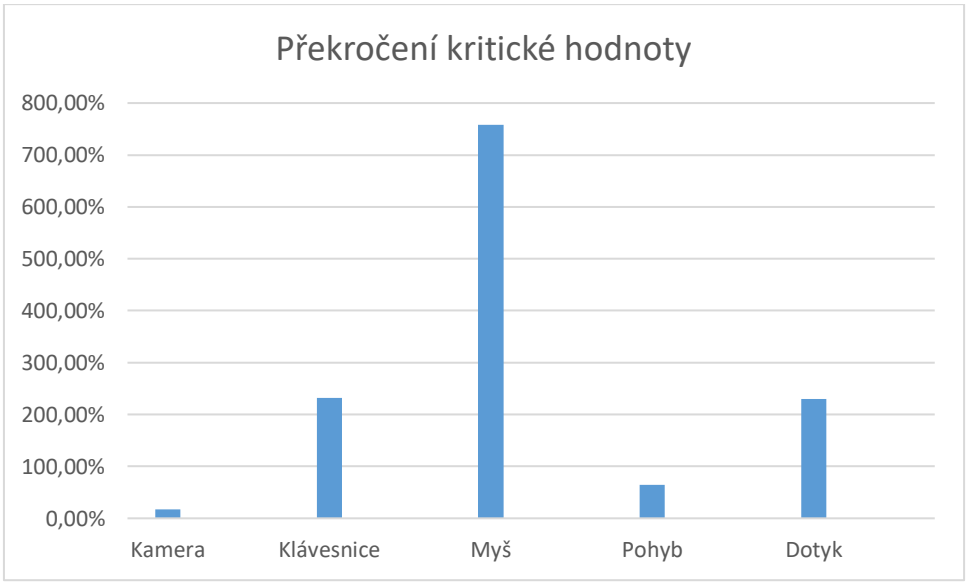
$<u_j;u_{j+1})$		**(-1;0>**	**(0;1>**	**(1;2>**	**(2;3>**	**(3;4>**	**(4;5>**	**(5;6>**	**(6;7>**	**(7;8>**	**(8;9>**
x		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
n_j	1	2834	5840	3633	3444	3329	3642	3780	3505	3457	3189
	2	3542	7618	4589	4726	4742	4743	4558	4441	4465	4173
	3	3589	7212	4503	4693	4575	4689	5033	4569	4291	4097
N_j	1	2834	8674	12307	15751	19080	22722	26502	30007	33464	36653
	2	3542	11160	15749	20475	25217	29960	34518	38959	43424	47597
	3	3589	10801	15304	19997	24572	29261	34294	38863	43154	47251
Φ(x)	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	2	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
	3	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$F_n(x)$	1	0,0773	0,2367	0,3358	0,4297	0,5206	0,6199	0,7231	0,8187	0,9130	1
	2	0,0744	0,2345	0,3309	0,4302	0,5298	0,6295	0,7252	0,8185	0,9123	1
	3	0,0760	0,2286	0,3239	0,4232	0,5200	0,6193	0,7258	0,8225	0,9133	1
$\lvert\Phi(x)-F_n(x)\rvert$	1	0,0227	0,0367	0,0358	0,0297	0,0206	0,0199	0,0231	0,0187	0,0130	0
	2	0,0256	0,0345	0,0309	0,0302	0,0298	0,0295	0,0252	0,0185	0,0123	0
	3	0,0240	0,0286	0,0239	0,0232	0,0200	0,0193	0,0258	0,0225	0,0133	0

$D_{n,\alpha}$	1	0,0075	**D_n**	0,0367	**zamít. H_0?**	ANO
	2	0,0268		0,0345		ANO
	3	0,0077		0,0286		ANO

Tabulka 12 - Kolmogorov-Smirnovův test – Dotyk



Graf 8 - Porovnání testové statistiky s kritickou hodnotou - Kolmogorov-Smirnovův test



Graf 9 - Překročení kritické hodnoty - Kolmogorov-Smirnovův test

8.6. Souhrnné hodnocení

Podle výsledku předchozích testů je patrné, že pro každou metodu hodnocení dat vychází jako vhodný generátor jiný prototyp.

Ačkoliv toto zjištění může být matoucí, není tomu tak. Jak již bylo zmiňováno v kapitole 3.5 Statistické hodnocení náhodnosti čísel, každá metoda může poukázat na slabiny jednotlivých prototypů, její výsledky však nemohou být interpretovány jako potvrzení kvality generátoru.

Abychom mohli kvalifikovaně rozhodnout, zda generátor je, či není kvalitní, je třeba použít sadu různých testů a jejich výsledky pečlivě vyhodnotit.

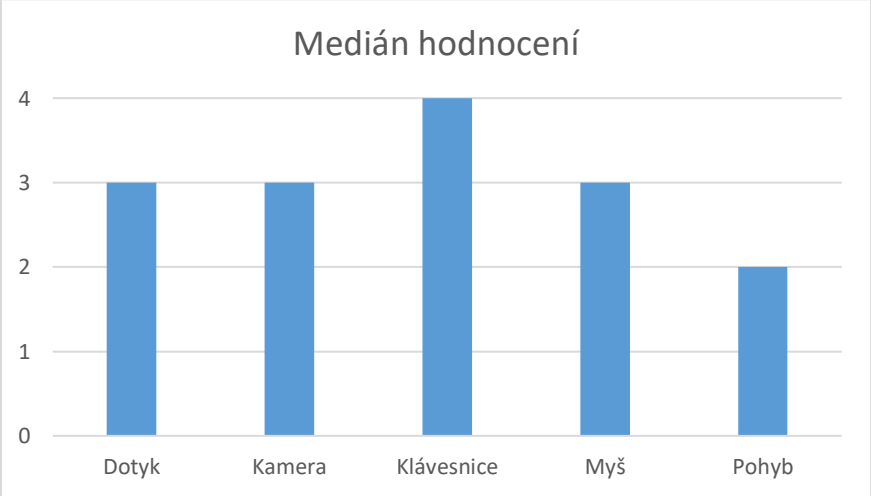
S ohledem na tuto skutečnost bylo zvoleno pro závěrečné shrnutí hodnocení na školní stupnici (máme 5 generátorů, tudíž 5 známek nám stačí). Pro rychlost generování byla jako nejlepší hodnota určena ta nejvyšší, stejně tak pro entropii informace. Naopak pro překročení kritické hodnoty u Kolmogorov-Smirnovova testu byla jako nejlepší ohodnocena nejnižší hodnota. Tato hodnocení uvádí tabulka Tabulka 13.

Generátor	Rychlost generování	Překročení kritické hodnoty	Entropie informace
Dotyk	2	3	4
Kamera	5	1	3
Klávesnice	4	4	1
Myš	3	5	2
Pohyb	1	2	5

Tabulka 13 - Hodnocení výsledků generátorů na školní stupnici

Následně byl ze školního hodnocení prototypů vypočítán medián, který zobrazuje Graf 10. Z něj nám jako nejlepší prototyp generátoru náhodných čísel vychází prototyp vytvořený na základě hodnot generovaných pohybem mobilního zařízení.

Tato metoda se jeví jako výhodná i s ohledem na minimální zatížení uživatele – stačilo by aplikaci, využívající tohoto generátoru zapnout na pozadí běhu systému a například za chůze či běhu nechat vygenerovat velké množství náhodných čísel, které by následně byly užity pro kryptografické potřeby.



Graf 10 - Medián hodnocení výsledků generátorů na školní stupnici

9. Závěr

Hlavním cílem této práce bylo vytvoření několika generátorů náhodných čísel a jejich srovnání na základě různých hledisek.

Na začátku práce je čtenář seznámen s problematikou náhodných čísel, jejich využitím a jsou představeny způsoby hodnocení náhodných posloupností. Dále jsou stručně představena stávající řešení generátorů náhodných čísel, které jsou k dispozici a technologie, které byly použity pro vytvoření prototypů generátorů náhodných čísel.

Bylo vytvořeno 5 generátorů fungujících na smartphonu a počítači. Pro smartphone byl zvolen OS Android, pro desktopové verze byl zvolen jazyk Java. Obojí bylo zvoleno s ohledem na snadné masové rozšíření aplikací využívajících tyto generátory.

Aby byl generováním náhodných čísel co nejméně zatížen uživatel, byly zvoleny takové metody sběru náhodných veličin, které uživatel běžně používá. Je to klávesnice a myš počítače, dotyk na displeji smartphone, jeho pohyb a instalovaná kamera. Pokud by aplikace využívající tyto generátory běžela na pozadí operačního systému, nevyžaduje ani jedna z metod nic, co by musel uživatel dělat navíc.

Jako ukázka praktického využití generátoru, postaveného na pohybu kurzoru myši na obrazovce PC, byla vytvořena jednoduchá aplikace šifrující text pomocí Vernamovy šifry.

Dále byla provedena 3 měření pro každý prototyp generátoru a tato měření byla hodnocena z pohledu rychlosti generování znaků, poté z hlediska míry přenášené informace a pro hodnocení náhodnosti byl využit Kolmogorov-Smirnovův test.

Ačkoliv je generování pomocí pohybu myši na obrazovce v praxi poměrně často využíváno, dopadlo v hodnocení náhodnosti nejhůře. A zatímco by podle očekávání měl být nejlepším generátorem ten založený na kameře smartphone, podle rychlosti generování je téměř nepoužitelný, jelikož generuje data velmi pomalu. Pro zrychlení generování by byl třeba velmi výkonný smartphone, což je zase v rozporu s masovou rozšiřitelností aplikací.

Autor práce si nekladl za cíl vytvořit plnohodnotný generátor náhodných čísel, ale pokusil se zjednodušit analýzu dalším autorům, kteří by hledali nejvhodnější metodu sběru náhodných hodnot pro využití v generátoru náhodných čísel.

Vzhledem k tomu, že budoucnost výpočetní techniky bude záviset na kvalitním šifrování, bude se požadavek na rychlý, jednoduchý a kvalitní generátor skutečně náhodných čísel objevovat čím dál častěji. Doufejme, že tato práce bude jedna ze zdrojů, které k vytvoření tohoto generátoru přispějí.

10. Seznam obrázků

11. Seznam tabulek

12. Seznam grafů

13. Seznam příloh na CD

- Složka Naměřená data obsahující generovaná data z každého běhu jednotlivých prototypů
- Archiv Zdrojové kódy.zip obsahující zdrojové kódy jednotlivých generátorů a implementace Vernamovy šifry
- Dokument DP.pdf – text této práce v elektronické podobě

14. Citovaná literatura

1. **Pecinovský, Rudolf.** *Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky.* 2., aktualiz. a rozš. vyd. Praha : Grada, 2009.

2. **Makovička, Jiří.** *Excel pro přírodovědce.* Vydání první. V Praze : Univerzita Karlova v Praze, nakladatelství Karolinum, 2016.

3. **Random.org: True Random Number Service. [Online] https://www.random.org/.**

4. Lórencz, Róbert a Hlaváč, Josef. Pokročilá kryptologie. místo neznámé : České vysoké učení technické v Praze.

5. HotBits: Genuine random numbers, generated by radioactive decay. [Online] 1996. http://www.fourmilab.ch/hotbits/.

6. LavaRnd. [Online] www.lavarand.org.

7. Elektronický podpis. ***Komunitní server sandbox.cz.*** **[Online] [Citace: 16. 10 2017.] http://sandbox.cz/~varvara/El_podpis/index.html.**

8. Praktické základy Kryptologie a Steganografie. ***Temné území.*** **[Online] [Citace: 16. 10 2017.] http://temneuzemi.webzdarma.cz/.**

9. Janák, Michal. Moderní aplikace šifer. ***Diplomová práce.*** **Praha : Bankovní institut vysoká škola Praha, 2009.**

10. Roman Danel. VŠB-TU Ostrava. [Online] [Citace: 15. 10 2017.] homel.vsb.cz/~dan11/bis/BIS_Danel_kryptografie.pptx.

11. Vysoké učení technické v Brně. [Online] [Citace: 15. 10 2017.] http://www.uai.fme.vutbr.cz/~matousek/TIK/fotoalbum/FotoalbumTIK1/bruceschneier.html.

12. Ikaros. ***Šifrování a šifrovací systémy.*** **[Online] [Citace: 15. 10 2017.] https://ikaros.cz/sifrovani-sifrovaci-systemy.**

13. Masarykova Univerzita. ***Historie Kryptografie.*** **[Online] [Citace: 15. 10 2017.] https://is.muni.cz/el/1451/podzim2010/d008/um/kryptografie.pdf.**

14. ***Algoritmy.net.*** **[Online] 2016. [Citace: 05. 11 2017.] https://www.algoritmy.net/article/95/Vernamova-sifra.**

15. ***Rozpad.cz.*** **[Online] 22. 11 2011. [Citace: 05. 11 2017.] http://rozpad.cz/forum/viewtopic.php?f=16&t=780.**

16. Arjen, K. Lenstra a Verheul, Eric R. ***Selecting cryptographic key size.*** **místo neznámé : Technische Universitet Eindhoven, 2001.**

17. Wolný, Stanislav. Elektronický podpis v podmínkách státní správy. [Online] 2013. [Citace: 05. 11 2017.]

18. Prouza, Marek. Elektronický podpis a certifikační autority. [Online] 2013. [Citace: 05. 11 2017.]

19. Pravděpodobnost. ***Matematika pro střední a základní školy.*** **[Online] 2006-2014. http://www.matematika.cz/pravdepodobnost.**

20. Teorie informace. ***Elektronické studijní materiály.*** **[Online] https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=7017.**

21. Veřejné služby Informačního systému. ***Testování generátorů náhodných čísel.*** **[Online] [Citace: 15. 10 2017.] https://is.muni.cz/el/1431/jaro2016/M6444/um/61762308/prednaska2_tisk.pdf.**

22. Otipka, Petr a Šmajstrla, Vladislav. 5. Základní typy rozdělení pravděpodobnosti spojité náhodné veličiny. ***PRAVDĚPODOBNOST A STATISTIKA.*** **[Online] VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA OSTRAVA. [Citace: 16. 10 2017.] http://homen.vsb.cz/~oti73/cdpast1/kap05/prav5.htm.**

23. Dřímal, J. a Trunec, D. ***Úvod do metody Monte Carlo.*** **místo neznámé : UJEP Brno, 1988.**

24. Rabová, Z. a kolektiv. ***Modelování a Simulace.*** **Brno : FEKT VUT, 1992.**

25. Státní ústav radiační ochrany. Jak funguje Geiger-Müllerův (GM) detektor? ***Státní ústav radiační ochrany.*** **[Online] [Citace: 11. 03 2018.] https://www.suro.cz/cz/faq/copy_of_jak-funguje-geiger-mulleruv-gm-detektor.**

26. Walker, John. HotBits: Genuine random numbers, generated by radioactive decay. ***HotBits.*** **[Online] 5 1996. [Citace: 11. 3 2018.] https://www.fourmilab.ch/hotbits/.**

27. Buchovecká, Simona. ***Testovanie pseudonáhodných postupností.*** **Praha : autor neznámý, 2010.**

28. ***Random.org.*** **[Online] 1998-2018. [Citace: 11. 3 2018.] https://www.random.org/.**

29. Petržela, Jiří. teorie elektronických obvodů. ***Ústav radioelektroniky.*** **[Online] VUT Brno. [Citace: 11. 3 2018.] http://www.urel.feec.vutbr.cz/MTEO/mteo/sumy.pdf.**

30. Schildt, Herbert. ***Mistrovství - Java.*** **Brno : Computer Press, 2014. ISBN 978-80-251-4145-8.**

31. There's no hope of anyone catching up to Android and iOS. ***Business Insider UK.*** **[Online] 22. 8 2016. [Citace: 24. 2 2018.] http://uk.businessinsider.com/smartphone-market-share-android-ios-windows-blackberry-2016-8.**

32. Marvan, Filip. Mobilní operační systém Android. ***diit.cz.*** **[Online] CDR server s.r.o., 27. 7 2011. [Citace: 24. 2 2018.] https://diit.cz/clanek/mobilni-operacni-system-android.**

33. Oracle. Oracle. ***Netbeans IDE.*** **[Online] [Citace: 24. 2 2018.] http://www.oracle.com/technetwork/developer-tools/netbeans/overview/index.html.**

34. Google Inc. Android Studio. ***Android Studio.*** **[Online] Google Inc. [Citace: 24. 2 2018.] https://developer.android.com/studio/index.html.**

35. Senzory v mobilních telefonech od A do Z. ***beryko.cz: specialisté na smartphony.*** **[Online] 2013-2017. https://www.beryko.cz/blog/recenze/senzory-v-mobilnich-telefonech-od-a-do-z.html.**

36. SensorEvent. ***Android Developers.*** **[Online] https://developer.android.com/reference/android/hardware/SensorEvent.html.**

37. Google, Inc. SensorEvent. ***Android Developers.*** **[Online] [Citace: 09. 11 2017.] https://developer.android.com/reference/android/hardware/SensorEvent.html#values.**

38. Kovařík, David. Mobil v roli fotoaparátu aneb snímače CMOS vs. CCD (vědecké okénko). ***Mobilizujeme.*** **[Online] 29. 1 2012. [Citace: 25. 2 2018.]**

https://mobilizujeme.cz/clanky/mobil-v-roli-fotoaparatu-aneb-snimace-cmos-vs-ccd-vedecke-okenko.

39. Zkratky.cz. *SMS.* [Online] http://www.zkratky.cz/SMS/6794.

40. Smartphony. *MobilMania.cz: O mobilech víme vše.* [Online] 2017. http://www.mobilmania.cz/smartphony/sc-319.

41. *How the iPod touch Works.* Howstuffworks.

42. Algoritmy.net. [Online] https://www.algoritmy.net/.

43. *Deterministický algoritmus.* 2016. Wikipedie: Otevřená encyklopedie.

44. *Generování náhodných čísel.* Elektronické studijní materiály.

45. Morseova abeceda. *Morseova abeceda.* [Online] [Citace: 16. 10 2017.] http://morseovaabeceda.cz/.

46. Hordějčuk, Vojtěch. Binární čísla a bitové operace. *Vojta Hordějčuk aka voho - Software Engineer and Bedroom Music Producer.* [Online] [Citace: 16. 10 2016.] http://voho.eu/wiki/bit/.

47. Open source. *Živě.* [Online] [Citace: 24. 2 2018.] https://www.zive.cz/open-source/sc-116/default.aspx. ISSN 1213-8991.

48. IDE. *IT Slovník.* [Online] [Citace: 24. 2 2018.] https://it-slovnik.cz/pojem/ide.