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 2019

Bibliografické údaje

Pecka, S. 2018: Návrh a implementace generátoru náhodných čísel. [Design and implementation of random number generator. Mgr. Thesis, in Czech.] 77 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Tato diplomová práce se zabývá vytvořením několika generátorů náhodných čísel. Data z prototypů jsou poté porovnána podle různých ukazatelů a statistických metod. Čtenář je seznámen se základními pojmy, existujícími generátory náhodných čísel a použitými technologiemi.

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.

Klíčová slova

Porovnání náhodných čísel, TRNG, generátor skutečně náhodných čísel

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, 26.11.2018

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 poklidnou atmosféru a klid na práci vždy, kdy bylo potřeba.

V neposlední řadě děkuji svému vedoucímu Ing. Petru Břehovskému za odborné rady při psaní této práce a doc. RNDr. Ivě Dostálkové, Ph.D. za konzultace matematické části 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 a 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.

Pro šifrovanou komunikaci je základem použití takové informace, která je pro útočníka, či jinou třetí stranu neznámá a nepředvídatelná. Je tedy třeba používat zdroj náhodných dat.

Pro využití většinu jako tento zdroj bohatě postačují generátory pseudonáhodných čísel, nicméně při vyměňování citlivých dat je žádoucí používat skutečně náhodná čísla. Proto je nutné pseudonáhodný generátor obohatit o vstup zcela náhodné veličiny. Vzhledem k tomu, že člověk je obklopen mnoha ději, které jsou náhodné, může se zdát, že takových generátorů můžeme mít mnoho.

Je však třeba zajistit, aby tento náhodný děj byl vhodně digitálně popsán, aby mohl být následně generátorem čísel použit. Následně jsou na generátory kladeny požadavky na rozložení hodnot, jejich střídání, nevznikání posloupností atd.

Tato práce si stanovuje za cíl několik náhodných dějů zpracovat a na jejich základě vytvořit prototypy generátorů náhodných čísel. Práce čtenáře provede implementací prototypů, následným statistickým zhodnocením vygenerovaných hodnot a dále generátory vzájemně porovnává 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

2. Cíle práce

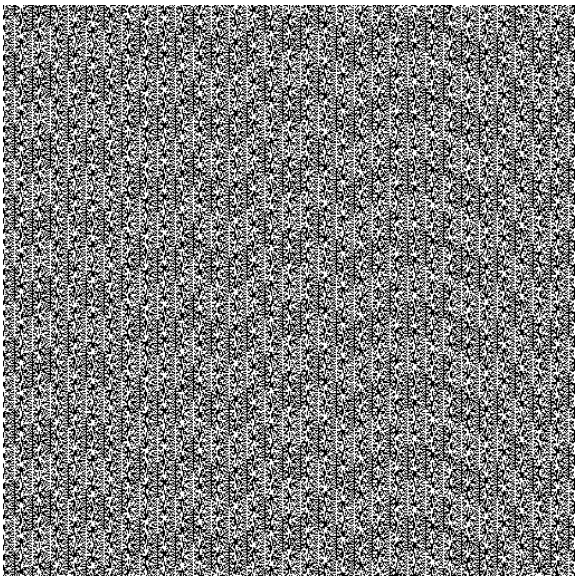
Hlavním cílem této práce je implementace prototypu generátoru skutečně náhodných čísel a následné zhodnocení získaných dat. Aby bylo tohoto cíle dosaženo, byly stanoveny dílčí cíle:

- 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[5], 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)

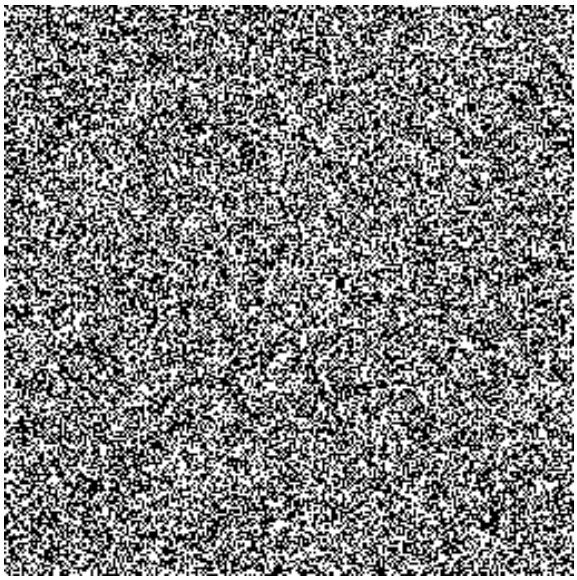
3.2. Náhodné číslo

Pro použití v kryptografii, která vyžaduje vysoký stupeň zabezpečení, se pseudonáhodná čísla nehodí. Používají se proto skutečně náhodná čísla vygenerovaná 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

[5] 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).

Podle Kupči (7) lze matematicky o čísle jako náhodném hovořit, pokud je součástí posloupnosti několika čísel, tj. v určitém kontextu. Pouze tehdy můžeme posloupnost podrobit zkouškám na náhodnost.



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

3.3. Útoky na pseudonáhodné generátory

Generátory pseudonáhodných čísel fungují na bázi konečného algoritmu (jako např. lineární kongruentní generátor, vzorec (3.1)), který vypočítává pseudonáhodné číslo pomocí předchozího vygenerovaného čísla. Při prvním běhu algoritmu ovšem není žádné předchozí číslo známo, je tedy potřeba využít nějaké iniciační sekvence. Tou může být systémové datum, vstupně/výstupní operace pevného disku apod.

$$x_{i+1} = (a \cdot x_i + b)\ mod\ m \tag{3.1}$$

Jelikož jsou generátory pseudonáhodných čísel použity ve většině běžných kryptografických situací, je pro útočníky lákavé zkompromitovat jejich funkčnost. Za takový útok považujeme každý, který nám umožní na výstupu poznat, že výstup je z pseudonáhodného generátoru, místo skutečně náhodných čísel. Kelsey (8) zmiňuje především následující způsoby:

- Přímé kryptoanalytické útoky – útoky, které bezprostředně zjišťují, zda se data na výstupu generátoru liší od náhodných dat. Takto je možno zkoumat většinu generátorů, některé ale mají výstupní sekvence schované a není možné je dopátrat – např. generátory, které jsou subprocesem nějakého většího procesu.
- Útoky na vstup generátoru – jde také o zkoumání výstupních dat, nicméně v tomto případě má útočník kontrolu i nad vstupními daty generátoru. Podle způsobu kontroly vstupních dat rozlišujeme 3 druhy útoku:
    - Zvoleného vstupu – útočník si sám vybírá, jaká data na vstup generátoru posílá. Jde především o generátory, které jako iniciační sekvenci využívají lehko ovlivnitelné údaje – hesla uživatelů, systémový čas apod.
    - Přehraného vstupu – útočník nemůže zvolit vstupní data, ale zná ta, která byla použita v minulosti a jaký výstup k nim byl přiřazen.
    - Známého vstupu – útočník v reálném čase sleduje, jaká data do generátoru vchází a jaká z něj vycházejí. Nemá možnost vstup ovlivnit. Jedná se o generátory využívající jako inicializační data sekvenci, kterou ve stejné chvíli může sledovat i útočník.
- Útoky na kompromitaci vnitřního stavu generátoru – útočník prolomil bezpečnost systému a odhalil vnitřní stav generátoru. Pomocí této informace se pokouší obnovit údaje z výstupu generátoru před kompromitací. Další z možností je trvalá změna vnitřního stavu generátoru, kdy útočník dokáže obnovit jak vstupní data před kompromitací, tak i všechna následující.

3.4. 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.

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

3.5. Šifrování

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

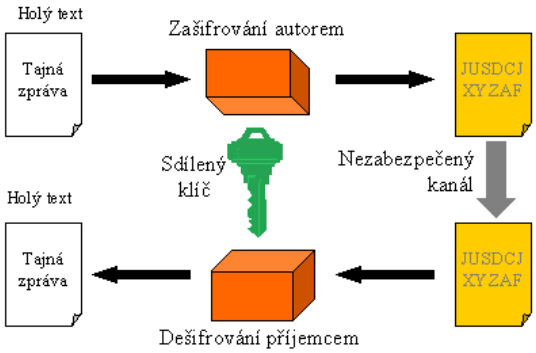
Základními pojmy kryptografie jsou:

- 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 ověření zdroje dat.

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

3.5.1. Symetrické šifrování

Symetrické šifrování je založeno na principu fungování společného klíče pro zašifrování a i dešifrování dat. Nevýhodou je, že odesílatel i příjemce se musí dohodnout na jednom klíči. Problém tedy může nastat při předání tohoto sdíleného klíče. Tento mechanismus znázorňuje Obrázek 3.



Obrázek 3 - Symetrické šifrování (9)

Tímto klíčem mohou být kódové tabulky, šifrovací kotouče (viz Obrázek 4), nastavení šifrovacího stroje, domluvená abeceda (např. Morseova abeceda[6]) apod.

Obrázek 4 - Albertiho šifrovací disk – z doby americké občanské války (10)

Následují příklady symetrických šifer.

3.5.1.1. Blowfish

Šifra navržená Brucem Schneierem a jeho týmem jako alternativa k DES[7]. 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[8] (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.

[6] Morseova abeceda je sestava kódů interpretujících písmena, čísla a speciální znaky (45)
[7] DES (Data/Digital Encryption Standard – Standard datového/digitálního šifrování) – šifra vyvinutá v letech 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).
[8] XOR – bitová operace - exkluzivní součet vrací 1 v případě, že se sčítance liší, v ostatních případech vrací 0 (46).

3.5.1.2. Twofish

Také šifra Twofish vychází z „dílny“ Bruce Schneiera. navazuje na Blowfish a 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).

3.5.1.3. 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.

3.5.1.4. 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).

3.5.1.5. RC4

Licenčně placená proudová 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).

3.5.1.6. 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[9] zvyšujeme tím pravděpodobnost prolomení (15). Stejně tak znovupoužití šifry snižuje bezpečnost – za předpokladu, že útočník toto předpokládá, lze z více zpráv odhadnout šifru.

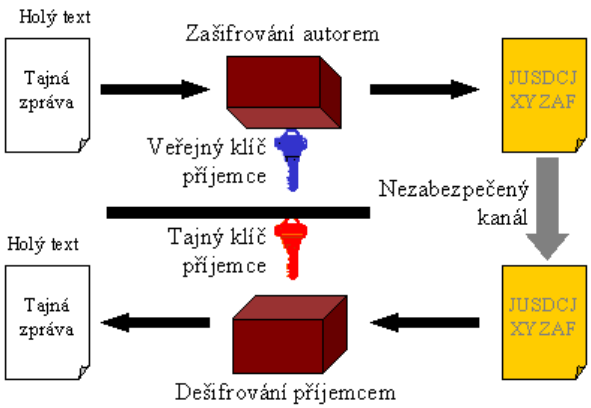
V praxi se pro výměnu klíčů používá velký soubor vygenerovaných čísel předaný za bezpečných okolností mezi dvěma komunikujícími stranami a při zasílání šifrované zprávy se z tohoto souboru „ukrajuje“ podle délky zaslané zprávy.

3.5.2. Asymetrické šifrování

Při asymetrickém šifrování je využito dvou šifrovacích klíčů, a to veřejného a privátního. Pro zašifrování dat na straně odesílatele je použit veřejný klíč, příjemce pro dešifrování použije klíč privátní.

Výhodou tohoto typu je, že dešifrovací klíč není třeba sdílet. Je tedy zajištěno, že data přečte jen oprávněný příjemce. Přitom pro komunikaci s kýmkoliv stačí pouze jedna kombinace veřejného a privátního klíče, zatímco u symetrického šifrování je třeba mít zvláštní šifrovací klíč pro komunikaci s různými odesílateli.

Princip asymetrického šifrování je znázorňuje Obrázek 5



Obrázek 5 - Princip asymetrické šifry (16)

Následují příklady asymetrických šifer.

3.5.2.1. 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),

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

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. „Je snadné vynásobit dvě dlouhá (100 místná) prvočísla, ale bez jejich znalosti je prakticky nemožné zpětně provést rozklad výsledku na původní prvočísla.“ (13) Bezpečný bude do té doby, než bude nalezen algoritmus faktorizace velkých čísel v krátkém čase. Používá šifrování s veřejným klíčem – jednosměrný hashovaný přenos, vhodný pro elektronické podpisy (12).

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 2048 bitů, výjimkou však není ani 3072 bitů. (16).

3.5.2.2. 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.

3.5.2.3. 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, donedávna se využívala jako bezpečný protokol pro zabezpečení https. Používá se tedy pro komunikaci, která nesmí být odposlechnuta. Uživatel potřebuje pro ověření důvěryhodnosti platný certifikát. (13).

3.5.3. 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 žadatelem při žádosti o certifikát 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 ověřující 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).

3.6. 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 těžko odhalitelné hardwarové závady. 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 zárove nelze prokázat, že kvalitní je. Skutečnost, že generátor „projde“ všemi testy, ještě nezaručuje, že neobsahuje slabinu, jež testy nebyly schopny odhalit.

Shodně se vyjadřuje i Kupča (7) – protože klasické počítače pracují na bázi deterministických algoritmů, lze vždy skončit u závěru, že je generátor budˇ lepší nebo horší. Vždy je možnost, že daná posloupnost splní několik testů na náhodnost, ale další zkouška v posloupnosti odhalí skrytý vzorec.

Kupča (7) dále tvrdí, že odhalení náhodnosti je problém nealgoritmizovatelný, tudíž nevypočitatelný.

3.6.1. 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

$$P(A) = \frac{|A|}{|\omega|} \tag{3.2}$$

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

3.6.2. 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 (4) 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.

Požadované vlastnosti funkce pro výpočet množství informace jsou:

- Jev[10] X má n realizací[11], 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 realizace, tím je větší množství informace v něm obsažené.

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 střední hodnotou vlastních informací všech realizací jevů:

[10] Jev je náhodný pokus s n možnými realizacemi, např. tah loterie
[11] Realizace jevu je jeden projev, získání výsledku

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log p(x_i) \tag{3.3}$$

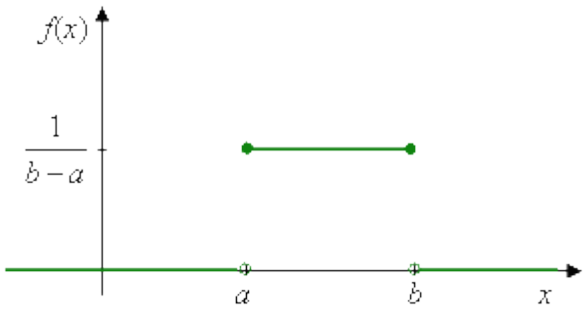
3.6.3. Kolmogorov-Smirnovův test

„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).

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

$$\Phi(\mathrm{x}) = \begin{cases} \dfrac{1}{b-a} & \text{pro } x \in \langle a, b \rangle \\ 0 & \text{pro } x \notin \langle a, b \rangle \end{cases} \tag{3.4}$$

Rovnoměrné rozdělení je znázorněno na Obrázek 6.



Obrázek 6 - Graf hustoty pravděpodobnosti pro rovnoměrné rozložení (22)

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

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

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

$$F_n(x) = \frac{1}{n} card\{i; X_i \leq x\}^{12} \tag{3.6}$$

Na hladině významnosti α testujeme nulovou hypotézu

$$H_0: \Phi(x) = F_n(x) \text{ pro } \forall x \in R \tag{3.7}$$

proti alternativě

$$H_1: \Phi(x) \neq F_n(x) \tag{3.8}$$

pro aspoň jednu hodnotu x. Testová statistika má tvar

$$D_n = \max_{x \in R} |\phi(x) - F_n(x)| \tag{3.9}$$

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

$$D_{n,a} \approx \sqrt{\frac{1}{2n} ln \frac{2}{\alpha}}. \text{ (21).“} \tag{3.10}$$

3.6.4. 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 bodů 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

[12] Funkce card je v jazyce R definována jako souhrn počtu sousedních čísel v oblasti *i* ze seznamu sousedů

$$U = \frac{Y - \frac{2(n-2)}{3}}{\sqrt{\frac{16n-29}{90}}} \approx N(0,1) \tag{3.11}$$

Kritický obor $W = (-\infty, -u_{1-\frac{\alpha}{2}}\rangle \cup \langle -u_{1-\frac{\alpha}{2}}, \infty)$.

Pokud $U \in W$, nulovou hypotézu zamítáme na hladině významnosti α. Hodnota $u_{1-\frac{\alpha}{2}}$ se určí aproximací normalizovaného normálního rozdělení. Pro nejčastěji používané hladiny významnosti se používá tabelovaná hodnota.

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

3.6.5. 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 prvních 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

$$U = \frac{Y - \frac{n-1}{2}}{\sqrt{\frac{n+1}{12}}} \approx N(0,1) \tag{3.12}$$

Kritický obor: $W = (-\infty, -u_{1-\alpha/2}\rangle \cup \langle -u_{1-\alpha/2}, \infty)$. Pokud $U \in W$, nulovou hypotézu zamítáme na hladině významnosti α. Stejně jako u extremálních bodů se hodnota $u_{1-\alpha/2}$ vyčte z matematických tabulek.

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

3.6.6. 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

$$R_k = \frac{\frac{1}{n-k}\sum_{i=1}^{n-k}(x_i - \bar{x})(x_{k+i} - \bar{x})}{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_{k+i} - \bar{x})^2} \tag{3.13}$$

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).

3.6.7. Chi-kvadrát test

Testuje nulovou 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).

Chi-kvadrát test a Kolmogorov-Smirnovův test testují totéž, avšak u Chi-kvadrát testu musí být vyšší četnosti v jednotlivých intervalech.

3.6.8. 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[13] test.

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

$$p_r = \frac{d(d-1)\dots(d-r+1)}{d^k}\binom{k}{r} \tag{3.14}$$

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

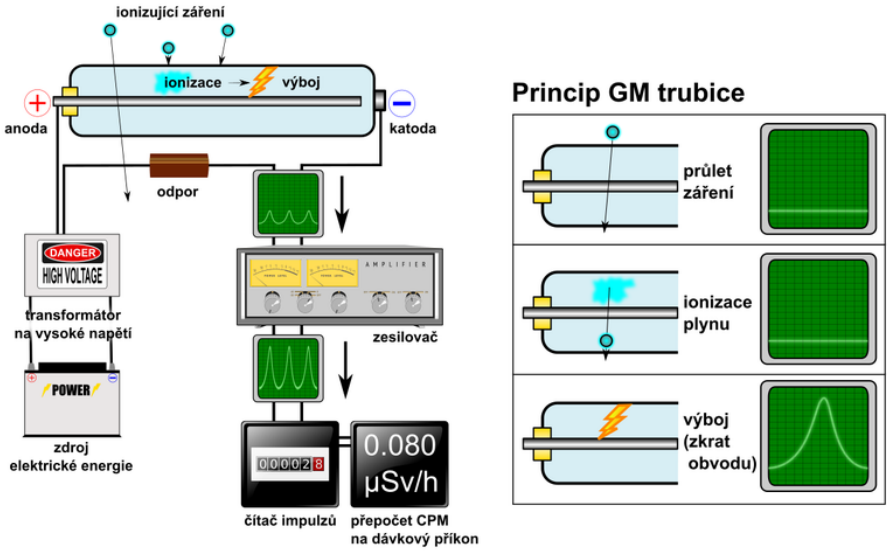
[13] Poker je karetní hra vyvinutá v USA. Hraje se na základě sázek na výherní kombinace 5 karet, které je možné složit ze 2 karet v hráčově ruce a 5 vyložených karet dealerem.

4. Stávající řešení generátorů skutečně náhodných čísel

4.1. HotBits – radioaktivní rozpad

Hotbits je generátor čísel, který nabízí skutečně náhodné sekvence. Řídí se inherentní[14] chybou kvantových mechanických přírodních zákonů při radioaktivním rozpadu. Generátor vytvořil John Walker.

Hotbits přenáší po sobě jdoucí dvojice radioaktivních rozpadů pomocí Geiger-Müllerovy trubice (viz Obrázek 7) 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 7 - 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[15] 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

[14] 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
[15] 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

jádro 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 se zahodí 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, jejichž 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 – 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 ekvalizace
- barevný – s nerovnoměrným rozložením hustoty kmitočtových složek ve spektru,
- praskavý šum – vytvářen znečištěným přechodem mezi bází a emitorem iontů těžkých kovů, hustota jejich výkonu má mnoho frekvencí a klesá s rostoucím počtem kmitočtů,
- blikavý – 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 (viz kapitola 3.1). 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 dostupných do roku 1991 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.

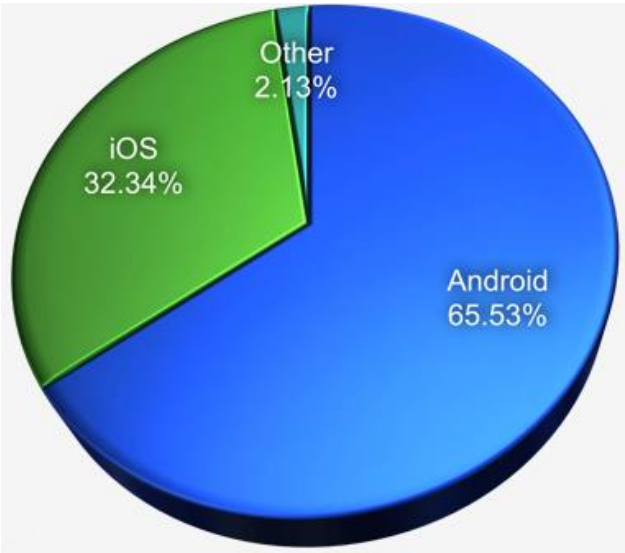
Mezi další výhody Javy podle Schildta (30) patří:

- 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 se výrazně 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. Je šířen jako opensource[16] a v současné době je výrazně nejrozšířenějším na trhu (viz Obrázek 8). Programovacím jazykem, který je použit pro psaní aplikací pro Android je jazyk Java.



Obrázek 8 - Podíl mobilních operačních systémů na trhu v srpnu 2017 (33)

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

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

pro moderní chytré přístroje. Z počátku byli jen jedni z mnoha s tímto cílem, avšak roku 2005 Android koupila společnost Google.

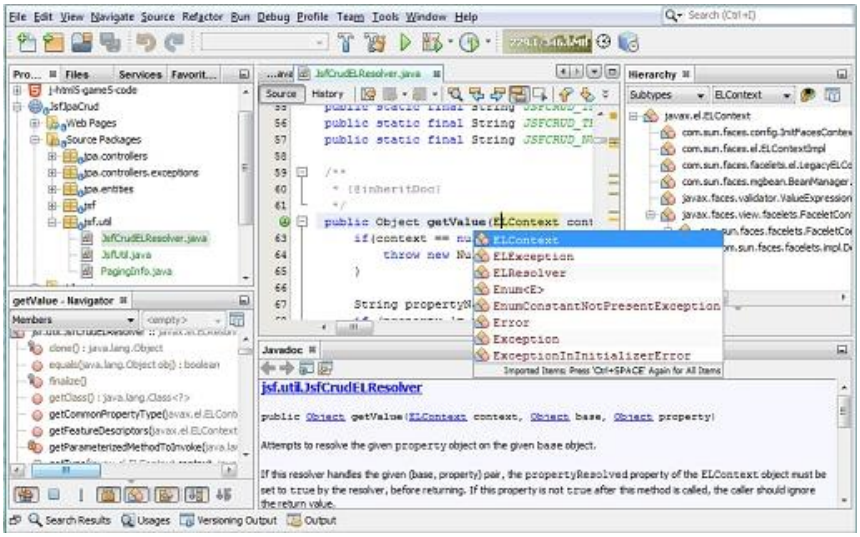
Google v roce 2007 založil spolu se společnostmi Nvidia, Samsung, LG, HTC, Motorola, Intel, Qualcomm, Ebay, T-Mobile a Telefonica konsorcium Open Handset Alliance, jehož cílem bylo podporovat a vyvíjet otevřený operační systém Android. První telefon s OS Android se dostal do prodeje v roce 2008 (32).

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

5.3. Netbeans IDE

Netbeans IDE[17] 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 roku 1996 skupinou studentů jako Xelfi. V roce 1999 byl koupen společností Sun Microsystems, která je zároveň (nyní prostřednictvím mateřské společnosti Oracle) hlavním sponzorem.



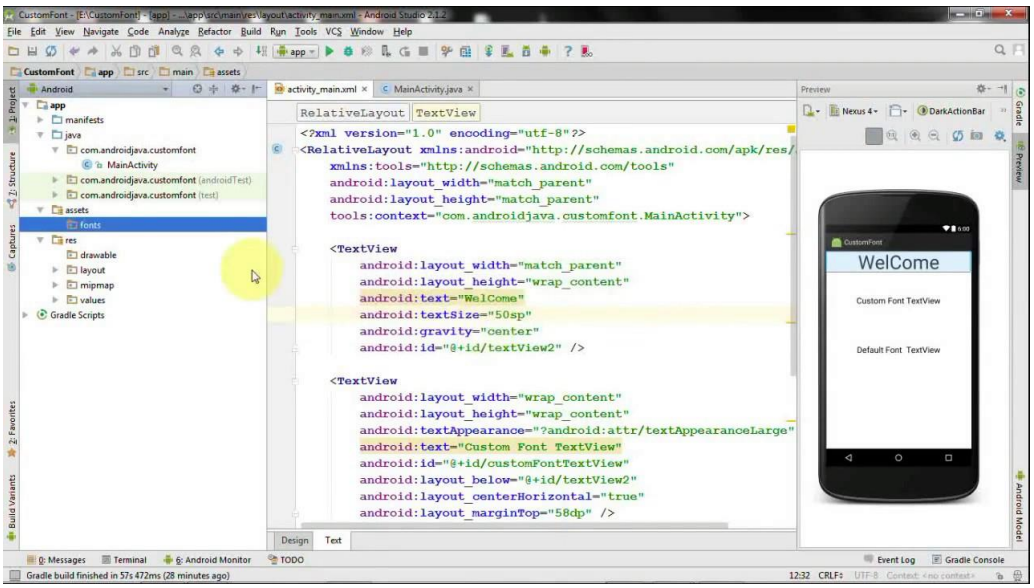
Obrázek 9- Netbeans IDE (33)

[17] 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 10 - 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 nutné 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 použita ve chvíli, kdy jsou potřeba.

6.1. Obecná funkce prototypů

Jelikož náhodné hodnoty zjišťované prototypy generátorů mohou být odlišné, a tudíž by se mezi sebou těžko porovnávaly, je třeba zajistit, aby generovaná čísla byla ze společného intervalu hodnot. Jako vhodný interval byl zvolen $< 0; 1)$. Aby přepočet do tohoto intervalu nedegradoval generované hodnoty, byl použit vzorec jehož čitatel je inspirován lineárně kongruentním generátorem pseudonáhodných čísel:

$$Random = \frac{(DM + RND)\,mod\,(HM - DM + 1)}{HM + C} \tag{6.1}$$

V tomto vzorci je $Random$ výsledné náhodné číslo z intervalu $< 0; 1)$, DM značí dolní mez generovaných hodnot (pro všechny prototypy v této práci platí $DM = 0$), RND značí náhodnou hodnotu získanou generátorem, HM značí horní mez generovaných hodnot a C je pomocná konstanta pro přepočet generované hodnoty na výsledné náhodné číslo.

Konstanta C by měla být zvolena co nejmenší, aby HM při přepočtu do intervalu $< 0; 1)$ měla hodnotu co nejbližší číslu 1. Pro potřeby této práce dostačovala tzv. dvoudevítková[18] přesnost, hodnota konstanty C se tedy určí

$$C = \frac{HM}{100} \tag{6.2}$$

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 byly nastaveny tak, aby měřily přesně 60 sekund. V průběhu měření jsou data shromažďována v paměti zařízení. Po uplynutí daného časového intervalu 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.

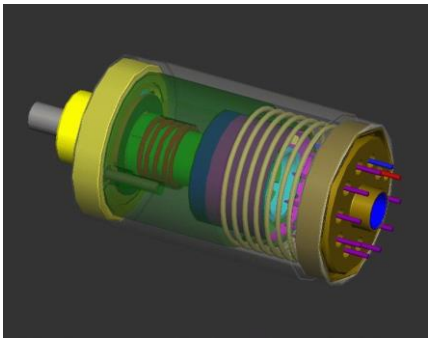
6.2. Snímač pohybu smartphone

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

[18] Devítková přesnost se uvádí např. jako čistota látek a znamená počet devítek za desetinou čárkou. Dvoudevítková přesnost tedy znamená 0,99, třídevítková 0,99 atp.
[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 nebo G-senzor

mikroskopické krystaly, na kterých se prostřednictvím piezoelektrického jevu z vibrací, způsobených změnou rychlosti, generuje napětí (35).



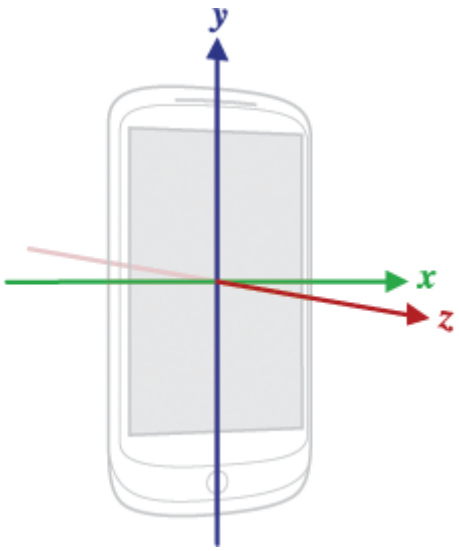
Obrázek 11 – Akcelerometr (35)

Pomocí akcelerometru tedy poznáme, že je s telefonem pohybováno. Vzhledem k tomu, že pohyb telefonu vykazuje určitou náhodnost – pohybuje se jinak, když ho máme 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 toto generování může probíhat na pozadí, bez aktivního zapojení uživatele – stačí když uživatel nechá aplikaci generovat čísla, když má telefon v kapse nebo pouzdře. Během toho 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

[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.



Obrázek 12 - 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 $0\frac{m}{s^2}$. Pokud tedy chceme zjistit reálnou akceleraci zařízení, je potřeba gravitační zrychlení zohlednit (37).

Výsledná hodnota, která používá jako náhodná hodnota RND je součet hodnot z těchto tří indexů pole *values*. Vzhledem ke skutečnosti, že akcelerace telefonu může být teoreticky nekonečná a pevné zvolení by způsobilo nerovnoměrné rozložení generovaných čísel na intervalu $< 0; 1)$, hodnota HM byla zvolena jako maximální hodnota z generovaných hodnot.

```
public void onSensorChanged(SensorEvent event) {
        if((System.currentTimeMillis() >= startTime + MEASURING_TIME) &&
write == true ){
            final double DM = 0;
            double HM = Collections.max(RNDs);
            final double C = HM/100;

            String toWrite = "DM=" + DM + "\n";
            toWrite += "HM=" + HM + "\n";
            toWrite += "C=" + C;

            for (double RND : RNDs) {
                double random = ((DM + RND) % (HM - DM + 1)) / (HM + C);
                finalValue += "\n" + String.valueOf(random);
            }

            toWrite += finalValue;

            String path = writeToFile(toWrite, "generated_values_move_" +
startTime + ".values");
            Toast.makeText(MoveActivity.this, "Měření bylo dokončeno. Soubor
s daty najdete zde: " + path, Toast.LENGTH_LONG).show();

            write = false;
        } else {
            float[] values = event.values;
            double value = Math.abs(values[0] + values[1] + values[2]);

            RNDs.add(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 pro hodnotu RND využit součet 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.

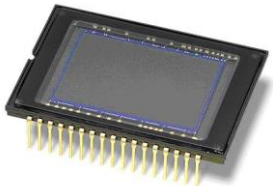
Hodnota *HM* byla zvolena jako součet maximální hodnoty AXIS_X a AXIS_Y.

```
lay.setOnTouchListener(new View.OnTouchListener() {
            public boolean onTouch(View view, MotionEvent event){
                final double C = HM/100;

                if(System.currentTimeMillis() >= startTime +
MEASURING_TIME  && write == true){
                    String toWrite = "DM=" + DM + "\n";
                    toWrite += "HM=" + HM + "\n";
                    toWrite += "C=" + C;

                    toWrite += finalValue;

                    String path = writeToFile(toWrite,
"generated_values_touch_" + startTime + ".values");
                    Toast.makeText(DotekActivity.this, "Měření bylo
dokončeno. Soubor s daty najdete zde: " + path,
Toast.LENGTH_LONG).show();
                    write = false;
                }else{
                    double RND = event.getRawX()+ event.getRawY();
                    double random = ((DM+RND)%(HM+1))/(HM-DM+C);
                    finalValue += "\n" + String.valueOf(random);
                }

                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 13 - 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 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. Dále z vektorového obrazu vytvoříme bitmapu pomocí třídy Bitmap a její funkne createBitmap. Následně je zvolen náhodný bod (pomocí funkce rand()) na zachyceném obrázku, z něhož je zjištěna hodnota barevných složek R, G a B a sečtena, čímž získáme hodnotu RND. Hodnota HM je součtem maximálních hodnot barevných složek kódu RGB.

Obrázek 14 - 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.

```
ImageReader.OnImageAvailableListener readerListener = new
ImageReader.OnImageAvailableListener() {
                @Override
                public void onImageAvailable(ImageReader reader) {
                    if (write == true) {
                        Image image = null;

                        image = reader.acquireLatestImage();

                        int width = image.getWidth();
                        int height = image.getHeight();

                        ByteBuffer buffer =
image.getPlanes()[0].getBuffer();
                        image.close();

                        Bitmap bitmap = Bitmap.createBitmap(width,
height, Bitmap.Config.ARGB_8888);
                        bitmap.copyPixelsFromBuffer(buffer);

                        Random rand = new Random();

                        int x = rand.nextInt(width);
                        int y = rand.nextInt(height);

                        int R = (bitmap.getPixel(x, y) >> 16) & 0xff;
                        int G = (bitmap.getPixel(x, y) >> 8) & 0xff;
                        int B = (bitmap.getPixel(x, y) >> 0) & 0xff;

                        double RND = R + G + B;
                        double random = ((DM + RND) % (HM - DM + 1)) /
(HM + C);


                        finalValue += "\n" + String.valueOf(random);

                        if (System.currentTimeMillis() >= startTime +
MEASURING_TIME && write == true) {
                            String toWrite = "DM=" + DM + "\n";
                            toWrite += "HM=" + HM + "\n";
                            toWrite += "C=" + C;

                            toWrite += finalValue;

                            String path = writeToFile(toWrite,
"generated_values_camera_" + startTime + ".values");
                            Log.e("STAKIDO", "mereni dokonceno " + path);
                            Toast.makeText(CameraActivity.this, "Měření
bylo dokončeno. Soubor s daty najdete zde: " + path,
Toast.LENGTH_LONG).show();

                            write = false;
                        }
                    }
                }
```

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. Tyto hodnoty jsou následně sečteny a použity jako hodnota RND. Hodnota HM je součtem maximální hodnoty x s maximální hodnotou y.

```
public void mouseMoved(MouseEvent e) {
                        double DM = 0;
                        double HM = frame.getContentPane().getHeight() +
frame.getContentPane().getWidth();
                        double C = HM/100;

                        if (System.currentTimeMillis() >= startTime +
MEASURING_TIME) {
                            String path;
                            try {
                                String toWrite = "DM=" + DM + "\n";
                                toWrite += "HM=" + HM + "\n";
                                toWrite += "C=" + C;

                                toWrite += finalValue;

                                path = writeToFile(toWrite,
"generated_values_mouse_" + startTime + ".values");
                            } 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 {
                            double RND = e.getPoint().x + e.getPoint().y;
                            double random = ((DM+RND)%(HM-DM+1))/(HM+C);
                            finalValue += "\n" + String.valueOf(random);
                        }
```

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, nebo deseti prsty, není prodleva mezi jednotlivými stisky stejná a vykazuje určitou míru náhodnosti.

```
public void keyPressed(KeyEvent e) {
                        if (System.currentTimeMillis() >= startTime +
MEASURING_TIME) {
                            double DM = 0;
                            double HM = Collections.max(RNDs);
                            System.out.println("HM je na indexu " +
RNDs.indexOf(HM));
                            double C = HM / 100;

                            String toWrite = "DM=" + DM + "\n";
                            toWrite += "HM=" + HM + "\n";
                            toWrite += "C=" + C;

                            for (double RND : RNDs) {
                                double random = ((DM + RND) % (HM - DM +
1)) / (HM + C);
                                finalValue += "\n" +
String.valueOf(random);
                            }

                            toWrite += finalValue;

                            String path;
                            try {
                                path = writeToFile(toWrite,
"generated_values_keyboard_" + startTime + ".values");
                            } 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 {
                            double time = System.currentTimeMillis();
                            double timeBetweenPress = time -
lastPressedTime;

                            if (lastPressedTime != 0)
                                RNDs.add(timeBetweenPress);

                            lastPressedTime = time;
                        }
                    }
```

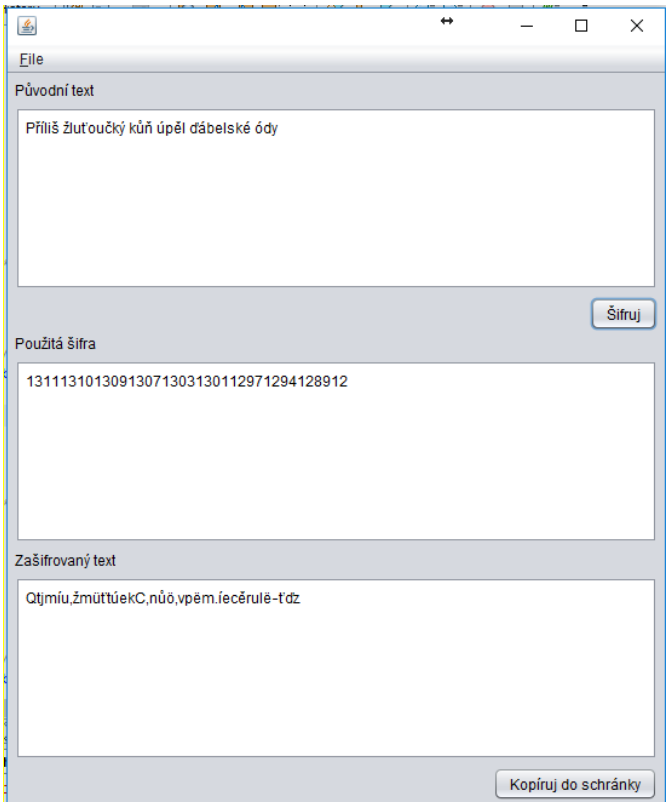
Ukázka kódu 5 – Zpracování hodnot ze stisku kláves

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 použit jako hodnota RND. Stejně jako u akcelometru akcelerace, může být i prodleva mezi stisky kláves nekonečné dlouhá. Hodnota HM byla tedy opět zvolena jako maximální prodleva.

7. Implementace Vernamovy šifry

Pro názornou ukázku využití generátoru náhodných čísel byla zvolena Vernamova šifra (viz kapitola 3.5.1.6) pro implementaci společně s generátorem náhodných čísel pomocí pohybu myší po obrazovce.



Obrázek 15 - Hlavní okno aplikace

Do hlavního okna (viz Obrázek 15) 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 potřeba distribuovat jak zašifrovaný text, tak i šifru samotnou. 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 15 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(ArrayList<Integer> cipher, String originalText)
throws Exception {
        if (cipher.size() == 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 + cipher.get(i);
                while (encIndex > (alphabet.length - 1)) {
                    encIndex -= (alphabet.length);
                }
                String newChar = Character.toString(alphabet[encIndex]);
                encodedText += newChar;
            }

            cipherArea.setText(Arrays.toString(cipher.toArray()));
            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 Dogee X5 (4 jádrový procesor 1,3 GHz, paměť RAM 1 GB, fotoaparát 5 MPx, 64 bitový operační systém Android 5.1).

Následně na datech bylo provedeno hodnocení kvality generátorů pomocí několika statistických metod.

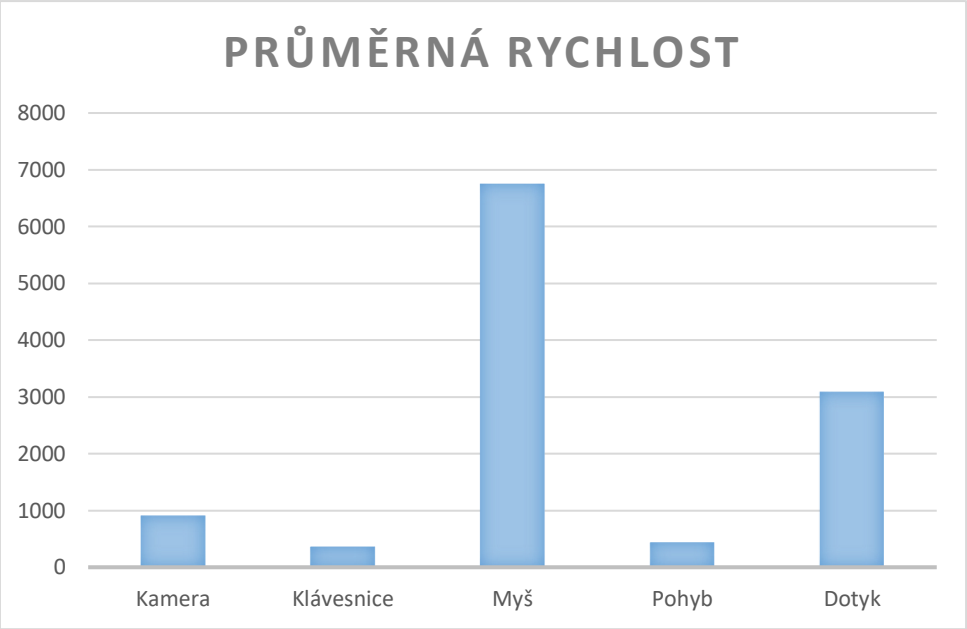
8.2. Rychlost generování dat

Množství vygenerovaných náhodných čísel za dobu 60 sekund obsahuje Tabulka 2. Žádný z generátorů neměl při jednotlivých měřeních větší odchylku – generátory za 1 minutu vygenerují vždy přibližně stejné množství dat.

Z prvního pohledu na Graf 1 je zřejmé, že zdaleka nejvíce, a to průměrně 6753 čísel, generuje prototyp generátoru založený na pohybu myši po obrazovce počítače. Zajímavou rychlost má i generátor založený na dotyku displeje – průměrně 3086 čísel. Naopak nejhorší z hlediska rychlosti generování je generátor využívající klávesnici počítače. Průměrná rychlost 364 čísel za minutu je velmi pravděpodobně způsobena tím, že není v silách člověka uhodit do klávesnice stejně často, jako např. počítač zaznamená pohyb myši. Dále velmi malé množství vygenerovaných čísel má pohyb telefonu.

Generátor	Číslo měření	Čísel za 1 min
Kamera	1	885
	2	917
	3	945
	průměr	**915**
Klávesnice	1	358
	2	366
	3	369
	průměr	**364**
Myš	1	6 970
	2	6 865
	3	6 424
	průměr	**6 753**
Pohyb	1	448
	2	430
	3	434
	průměr	**437**
Dotyk	1	3 259
	2	2 933
	3	3 066
	průměr	**3 086**

Tabulka 2 – Rychlost generování dat

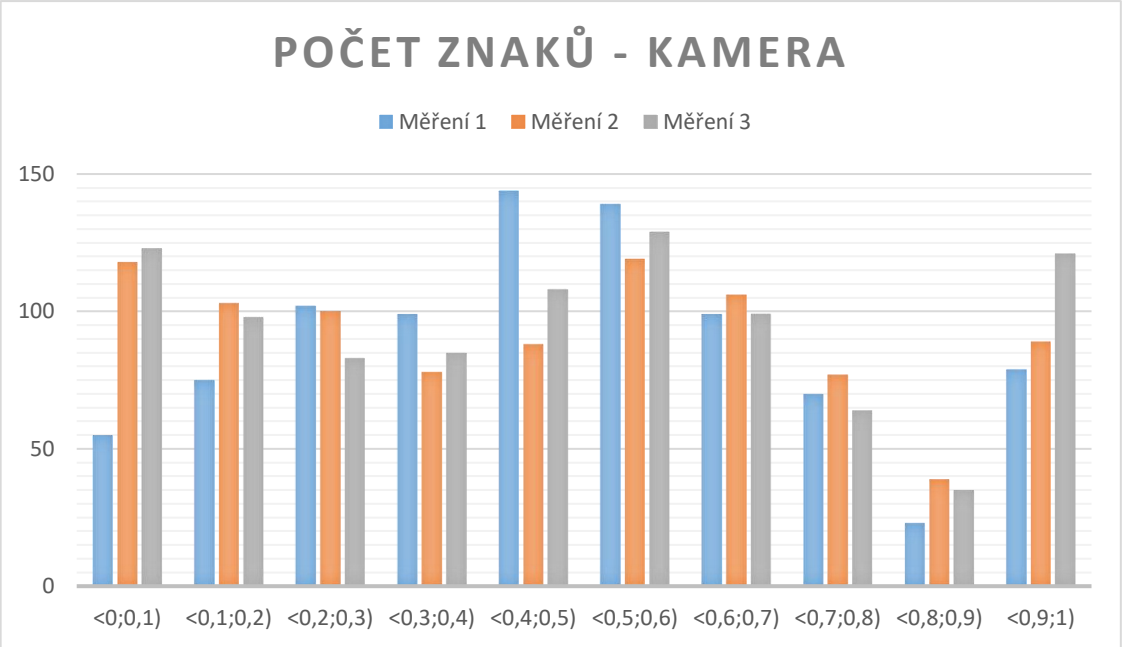


Graf 1 - Rychlost generování dat

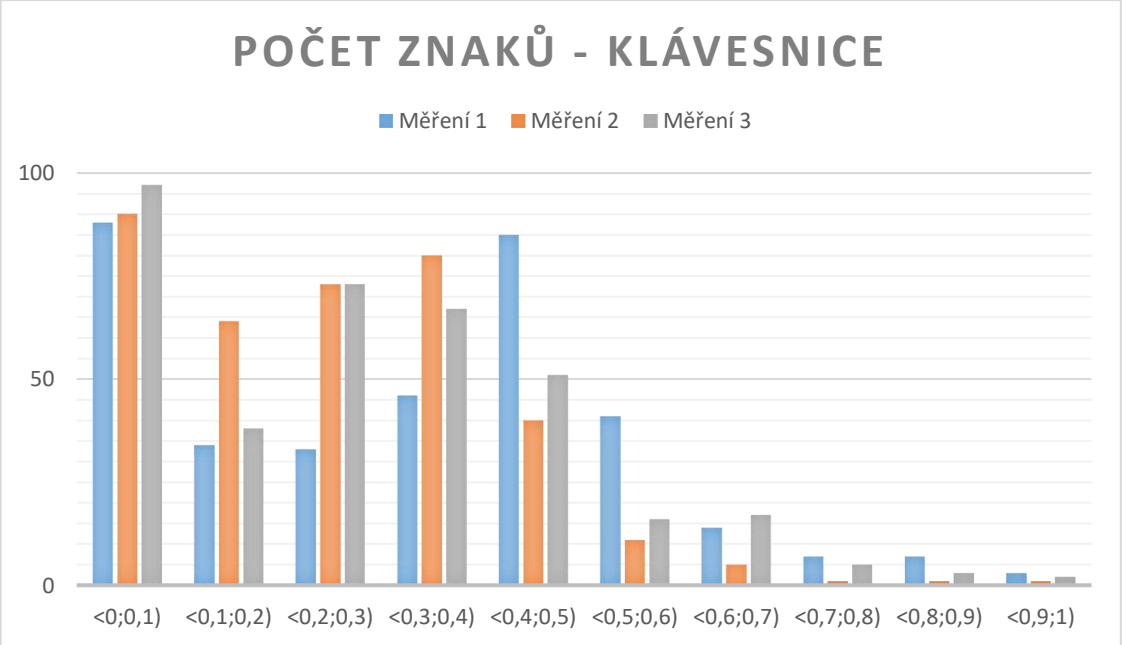
8.3. Počet jednotlivých znaků

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 objevují určitá pravidla v množství vygenerovaných čísel.

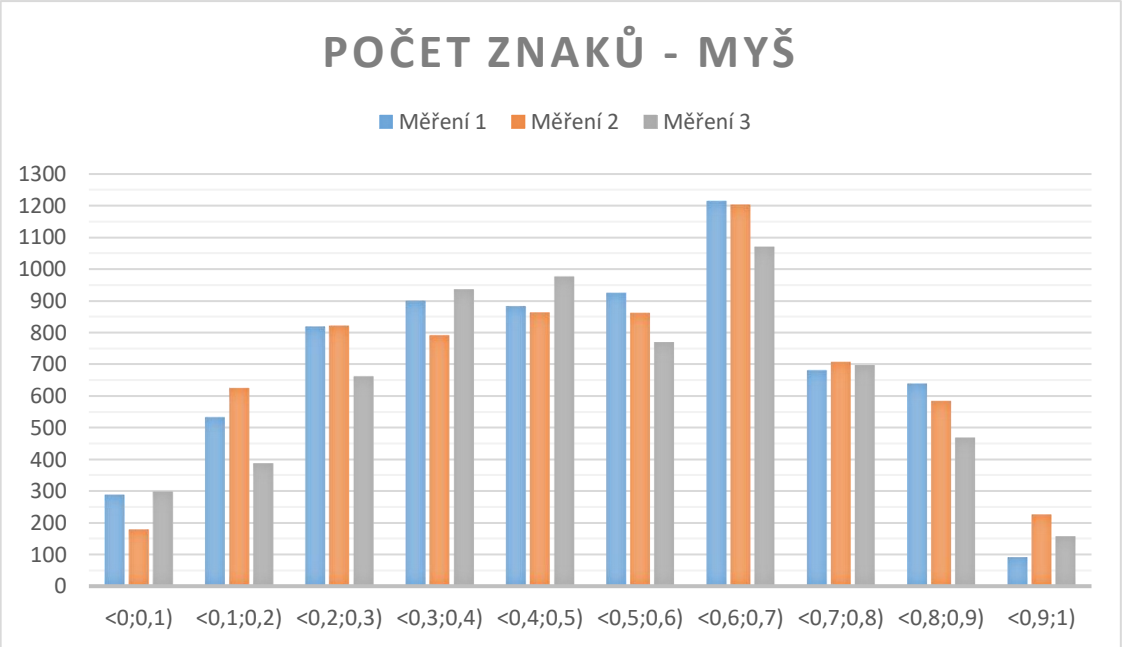
Například Graf 4 jasně ukazuje, že generátor využívající pohyb myši na monitoru generuje více znaků blíže ke středu intervalu $< 0; 1)$. Tato skutečnost poukazuje, že při generování hodnot nejčastěji kurzor myši prochází středem obrazovky. Stejný důvod bude i u generátoru využívajícího dotyk na displeji telefonu, který využívá stejnou metodiku pro zisk náhodných hodnot jako pohyb myši (Graf 6). Generátory využívající pohyb telefonu (Graf 5, měření číslo 1) a prodlevu mezi stisknutím kláves (Graf 3) poukazují na špatně zvolenou metodu při zvolení hodnoty HM. Tím, že došlo k extrémně dlouhé prodlevě mezi stiskem kláves, potažmo k extrémně prudkému pohybu, k nimž se už další prodlevy, resp. pohyby, nepřiblížily, se poté horní část intervalu $< 0; 1)$ nenaplnila dostatečným množstvím hodnot.



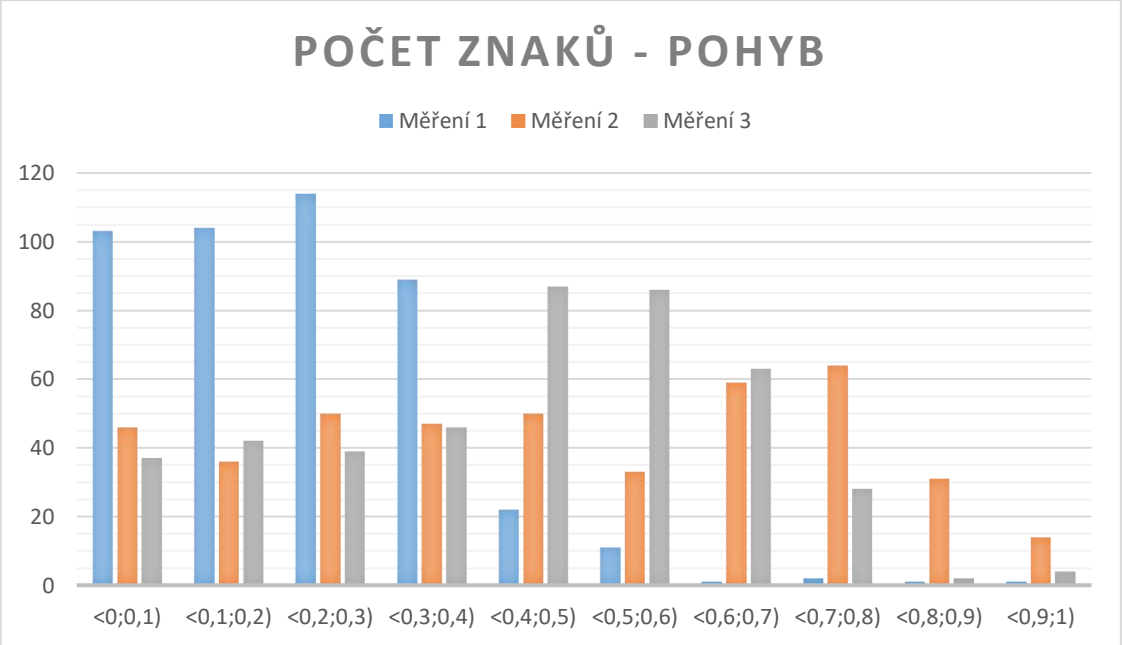
Graf 2 - Počet znaků – Kamera



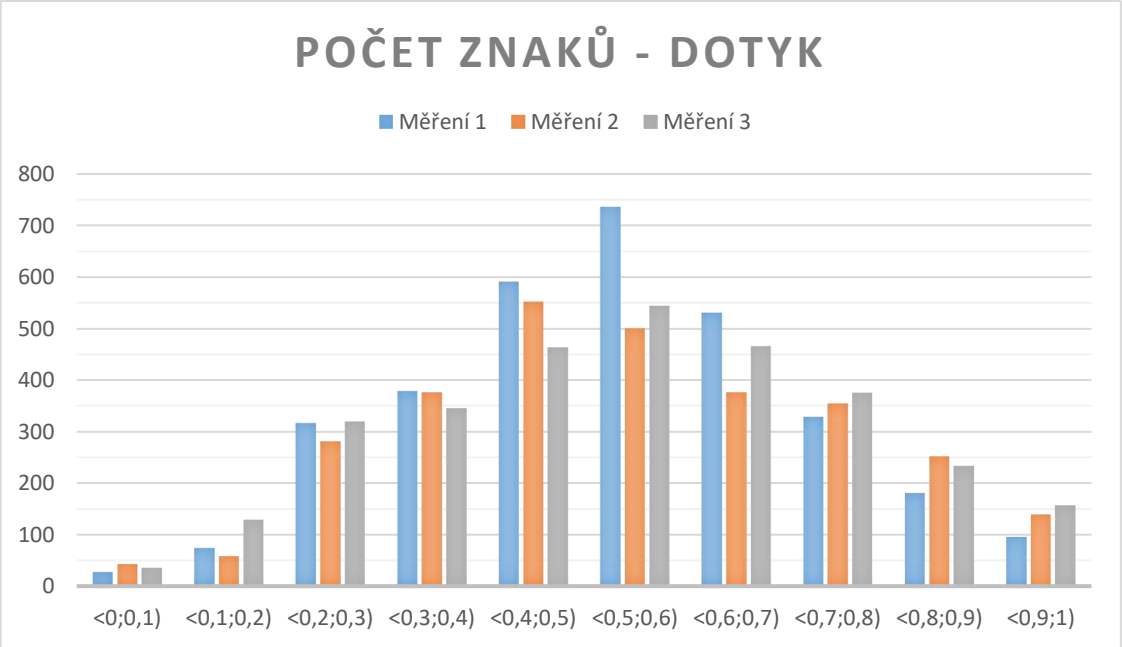
Graf 3 - Počet znaků – Klávesnice



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 následující v řetězci. Pokud je složité další znak odhadnout, dodává nám jeho zjištění větší informaci, než kdybychom věděli, jaké další číslo následuje. 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 (3.3) z kapitoly 3.6.2 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 telefonu. Naopak nejnižší entropii vykazuje generátor využívající kameru mobilního telefonu.

Spolu s pohybem myši po obrazovce má generátor využívající kameru v rámci jednotlivých měření stabilní entropii informace, zatímco u jiných prototypů (především pohyb telefonu a klávesnice počítače) entropie různě kolísá.

Následují tabulky s dílčími výsledky pro jednotlivá měření.

znak		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
počet	1	55	75	102	99	144	139	99	70	23	79
	2	118	103	100	78	88	119	106	77	39	89
	3	123	98	83	85	108	129	99	64	35	121
P(x) [%]	1	6,21%	8,47%	11,53%	11,19%	16,27%	15,71%	11,19%	7,91%	2,60%	8,93%
	2	12,87%	11,23%	10,91%	8,51%	9,60%	12,98%	11,56%	8,40%	4,25%	9,71%
	3	13,02%	10,37%	8,78%	8,99%	11,43%	13,65%	10,48%	6,77%	3,70%	12,80%
log P(x)	1	-1,21	-1,07	-0,94	-0,95	-0,79	-0,80	-0,95	-1,10	-1,59	-1,05
	2	-0,89	-0,95	-0,96	-1,07	-1,02	-0,89	-0,94	-1,08	-1,37	-1,01
	3	-0,89	-0,98	-1,06	-1,05	-0,94	-0,86	-0,98	-1,17	-1,43	-0,89
H(X)	1	10,45									
	2	10,17									
	3	10,25									

Tabulka 3 - Entropie informace – Kamera

znak		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
počet	1	88	34	33	46	85	41	14	7	7	3
	2	90	64	73	80	40	11	5	1	1	1
	3	97	38	73	67	51	16	17	5	3	2
P(x) [%]	1	24,58%	9,50%	9,22%	12,85%	23,74%	11,45%	3,91%	1,96%	1,96%	0,84%
	2	24,59%	17,49%	19,95%	21,86%	10,93%	3,01%	1,37%	0,27%	0,27%	0,27%
	3	26,29%	10,30%	19,78%	18,16%	13,82%	4,34%	4,61%	1,36%	0,81%	0,54%
log P(x)	1	-0,61	-1,02	-1,04	-0,89	-0,62	-0,94	-1,41	-1,71	-1,71	-2,08
	2	-0,61	-0,76	-0,70	-0,66	-0,96	-1,52	-1,86	-2,56	-2,56	-2,56
	3	-0,58	-0,99	-0,70	-0,74	-0,86	-1,36	-1,34	-1,87	-2,09	-2,27
H(X)	1	12,03									
	2	14,77									
	3	12,80									

Tabulka 4 - Entropie informace – Klávesnice

znak		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
počet	1	288	533	818	900	883	925	1213	680	639	91
	2	179	625	821	792	864	862	1203	707	585	227
	3	299	387	662	936	977	769	1071	697	468	158
P(x) [%]	1	4,13%	7,65%	11,74%	12,91%	12,67%	13,27%	17,40%	9,76%	9,17%	1,31%
	2	2,61%	9,10%	11,96%	11,54%	12,59%	12,56%	17,52%	10,30%	8,52%	3,31%
	3	4,65%	6,02%	10,31%	14,57%	15,21%	11,97%	16,67%	10,85%	7,29%	2,46%
log P(x)	1	-1,38	-1,12	-0,93	-0,89	-0,90	-0,88	-0,76	-1,01	-1,04	-1,88
	2	-1,58	-1,04	-0,92	-0,94	-0,90	-0,90	-0,76	-0,99	-1,07	-1,48
	3	-1,33	-1,22	-0,99	-0,84	-0,82	-0,92	-0,78	-0,96	-1,14	-1,61
H(X)	1	10,79									
	2	10,58									
	3	10,60									

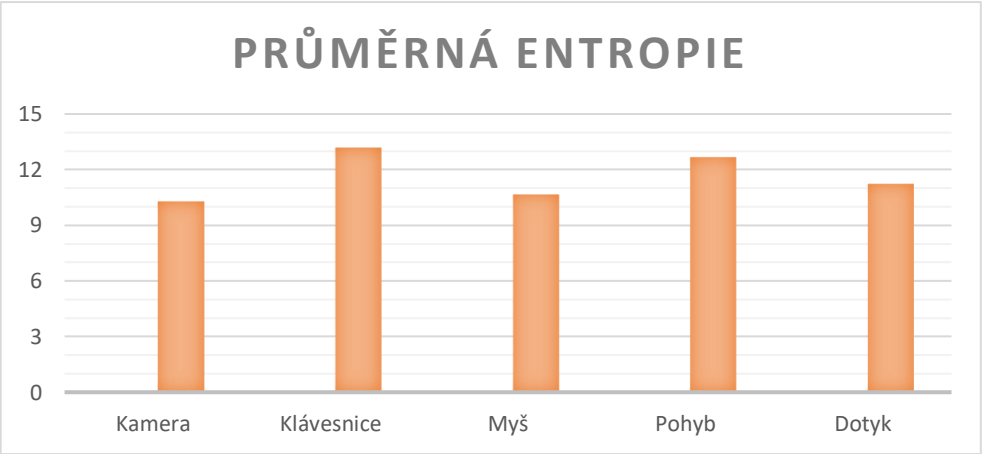
Tabulka 5 - Entropie informace – Myš

znak		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
počet	1	103	104	114	89	22	11	1	2	1	1
	2	46	36	50	47	50	33	59	64	31	14
	3	37	42	39	46	87	86	63	28	2	4
P(x) [%]	1	22,99%	23,21%	25,45%	19,87%	4,91%	2,46%	0,22%	0,45%	0,22%	0,22%
	2	10,70%	8,37%	11,63%	10,93%	11,63%	7,67%	13,72%	14,88%	7,21%	3,26%
	3	8,53%	9,68%	8,99%	10,60%	20,05%	19,82%	14,52%	6,45%	0,46%	0,92%
log P(x)	1	-0,64	-0,63	-0,59	-0,70	-1,31	-1,61	-2,65	-2,35	-2,65	-2,65
	2	-0,97	-1,08	-0,93	-0,96	-0,93	-1,11	-0,86	-0,83	-1,14	-1,49
	3	-1,07	-1,01	-1,05	-0,97	-0,70	-0,70	-0,84	-1,19	-2,34	-2,04
H(X)	1	15,79									
	2	10,31									
	3	11,91									

Tabulka 6 - Entropie informace – Pohyb

znak		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
počet	1	27	74	316	378	591	736	531	329	181	96
	2	43	58	281	376	552	501	376	355	252	139
	3	36	129	319	345	463	544	465	375	233	157
P(x) [%]	1	0,83%	2,27%	9,70%	11,60%	18,13%	22,58%	16,29%	10,10%	5,55%	2,95%
	2	1,47%	1,98%	9,58%	12,82%	18,82%	17,08%	12,82%	12,10%	8,59%	4,74%
	3	1,17%	4,21%	10,40%	11,25%	15,10%	17,74%	15,17%	12,23%	7,60%	5,12%
log P(x)	1	-2,08	-1,64	-1,01	-0,94	-0,74	-0,65	-0,79	-1,00	-1,26	-1,53
	2	-1,83	-1,70	-1,02	-0,89	-0,73	-0,77	-0,89	-0,92	-1,07	-1,32
	3	-1,93	-1,38	-0,98	-0,95	-0,82	-0,75	-0,82	-0,91	-1,12	-1,29
H(X)	1	11,63									
	2	11,14									
	3	10,95									

Tabulka 7 - Entropie informace – Dotyk



Graf 7 - Průměrná entropie informace

8.5. Shoda rozložení

Pro ověření shody s rovnoměrným rozložením čísel v generovaných posloupnostech byl vybrán Kolmogorov-Smirnovův test. Ten porovnává teoretickou a naměřenou distribuční funkci.

Kolmogorov-Smirnovův test byl pro hodnocení upřednostněn před Chi-kvadrát testem z toho důvodu, že pro Chi-kvadrát test jsou zapotřebí vyšší četnosti v jednotlivých intervalech, což by generátory založené na pohybu, klávesnici a kameře nesplňují.

V našem případě tedy test porovnává rovnoměrné rozdělení definované vzorcem (3.6) v kapitole 3.6.3 Kolmogorov-Smirnovův test a náhodný výběr definovaný vzorcem (3.5) 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 (3.10) zmíněné kapitoly.

Nulová hypotéza byla zamítnuta u všech hodnocení, což se ale dalo předpokládat při pouhém pohledu na četnost znaků v jednotlivých intervalech (Graf 2 – 6 v kapitole 8.3 Počet jednotlivých znaků). Jednotlivé generátory tak zhodnotíme podle toho, jakou měrou překročila testová statistika kritickou hodnotu.

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

Zajímavý je pohled na testové statistiky jednotlivých prototypů – zatímco prototypy využívající myš a dotyk (Tabulka 10 a Tabulka 12) ji mají pro jednotlivá měření bez kolísání, prototypy založeny na pohybu telefonu a klávesnice (Tabulka 11 a Tabulka 9) kolísají značně.

$<u_j;u_{j+1})$		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
x		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
n_j	1	55	75	102	99	144	139	99	70	23	79
	2	118	103	100	78	88	119	106	77	39	89
	3	123	98	83	85	108	129	99	64	35	121
N_j	1	55	130	232	331	475	614	713	783	806	885
	2	118	221	321	399	487	606	712	789	828	917
	3	123	221	304	389	497	626	725	789	824	945
Φ(x)		0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$F_n(x)$	1	0,0621	0,1469	0,2621	0,3740	0,5367	0,6938	0,8056	0,8847	0,9107	1,0000
	2	0,1287	0,2410	0,3501	0,4351	0,5311	0,6609	0,7764	0,8604	0,9029	1,0000
	3	0,1302	0,2339	0,3217	0,4116	0,5259	0,6624	0,7672	0,8349	0,8720	1,0000
$\lvert\Phi(x)-F_n(x)\rvert$	1	0,0379	0,0531	0,0379	0,0260	0,0367	0,0938	0,1056	0,0847	0,0107	0,0000
	2	0,0287	0,0410	0,0501	0,0351	0,0311	0,0609	0,0764	0,0604	0,0029	0,0000
	3	0,0302	0,0339	0,0217	0,0116	0,0259	0,0624	0,0672	0,0349	0,0280	0,0000

$D_{n,\alpha}$			D_n		**zamít. H_0?**	
	1	0,0457		0,1056		ANO
	2	0,0448		0,0764		ANO
	3	0,0442		0,0672		ANO

Tabulka 8 - Kolmogorov-Smirnovův test – Kamera

$<u_j;u_{j+1})$		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
x		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
n_j	1	88	34	33	46	85	41	14	7	7	3
	2	90	64	73	80	40	11	5	1	1	1
	3	97	38	73	67	51	16	17	5	3	2
N_j	1	88	122	155	201	286	327	341	348	355	358
	2	90	154	227	307	347	358	363	364	365	366
	3	97	135	208	275	326	342	359	364	367	369
Φ(x)	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$F_n(x)$	1	0,2458	0,3408	0,4330	0,5615	0,7989	0,9134	0,9525	0,9721	0,9916	1,0000
	2	0,2459	0,4208	0,6202	0,8388	0,9481	0,9781	0,9918	0,9945	0,9973	1,0000
	3	0,2629	0,3659	0,5637	0,7453	0,8835	0,9268	0,9729	0,9864	0,9946	1,0000
$\lvert\Phi(x)-F_n(x)\rvert$	1	0,1458	0,1408	0,1330	0,1615	0,2989	0,3134	0,2525	0,1721	0,0916	0,0000
	2	0,1459	0,2208	0,3202	0,4388	0,4481	0,3781	0,2918	0,1945	0,0973	0,0000
	3	0,1629	0,1659	0,2637	0,3453	0,3835	0,3268	0,2729	0,1864	0,0946	0,0000

$D_{n,\alpha}$			D_n		**zamít. H_0?**	
	1	0,0718		0,3134		ANO
	2	0,0710		0,4481		ANO
	3	0,0707		0,3835		ANO

Tabulka 9 - Kolmogorov-Smirnovův test – Klávesnice

$<u_j;u_{j+1})$		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
x		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
n_j	1	288	533	818	900	883	925	1213	680	639	91
	2	179	625	821	792	864	862	1203	707	585	227
	3	299	387	662	936	977	769	1071	697	468	158
N_j	1	288	821	1639	2539	3422	4347	5560	6240	6879	6970
	2	179	804	1625	2417	3281	4143	5346	6053	6638	6865
	3	299	686	1348	2284	3261	4030	5101	5798	6266	6424
Φ(x)	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$F_n(x)$	1	0,0413	0,1178	0,2352	0,3643	0,4910	0,6237	0,7977	0,8953	0,9869	1,0000
	2	0,0261	0,1171	0,2367	0,3521	0,4779	0,6035	0,7787	0,8817	0,9669	1,0000
	3	0,0465	0,1068	0,2098	0,3555	0,5076	0,6273	0,7941	0,9026	0,9754	1,0000
$\lvert\Phi(x)-F_n(x)\rvert$	1	0,0587	0,0822	0,0648	0,0357	0,0090	0,0237	0,0977	0,0953	0,0869	0,0000
	2	0,0739	0,0829	0,0633	0,0479	0,0221	0,0035	0,0787	0,0817	0,0669	0,0000
	3	0,0535	0,0932	0,0902	0,0445	0,0076	0,0273	0,0941	0,1026	0,0754	0,0000

$D_{n,\alpha}$			D_n		**zamít. H_0?**	
	1	0,0163		0,0977		ANO
	2	0,0164		0,0829		ANO
	3	0,0169		0,1026		ANO

Tabulka 10 - Kolmogorov-Smirnovův test – Myš

$<u_j;u_{j+1})$		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
x		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
n_j	1	103	104	114	89	22	11	1	2	1	1
	2	46	36	50	47	50	33	59	64	31	14
	3	37	42	39	46	87	86	63	28	2	4
N_j	1	103	207	321	410	432	443	444	446	447	448
	2	46	82	132	179	229	262	321	385	416	430
	3	37	79	118	164	251	337	400	428	430	434
Φ(x)	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$F_n(x)$	1	0,2299	0,4621	0,7165	0,9152	0,9643	0,9888	0,9911	0,9955	0,9978	1,0000
	2	0,1070	0,1907	0,3070	0,4163	0,5326	0,6093	0,7465	0,8953	0,9674	1,0000
	3	0,0853	0,1820	0,2719	0,3779	0,5783	0,7765	0,9217	0,9862	0,9908	1,0000
$\lvert Φ(x)-F_n(x)\rvert$	1	0,1299	0,2621	0,4165	0,5152	0,4643	0,3888	0,2911	0,1955	0,0978	0,0000
	2	0,0070	0,0093	0,0070	0,0163	0,0326	0,0093	0,0465	0,0953	0,0674	0,0000
	3	0,0147	0,0180	0,0281	0,0221	0,0783	0,1765	0,2217	0,1862	0,0908	0,0000

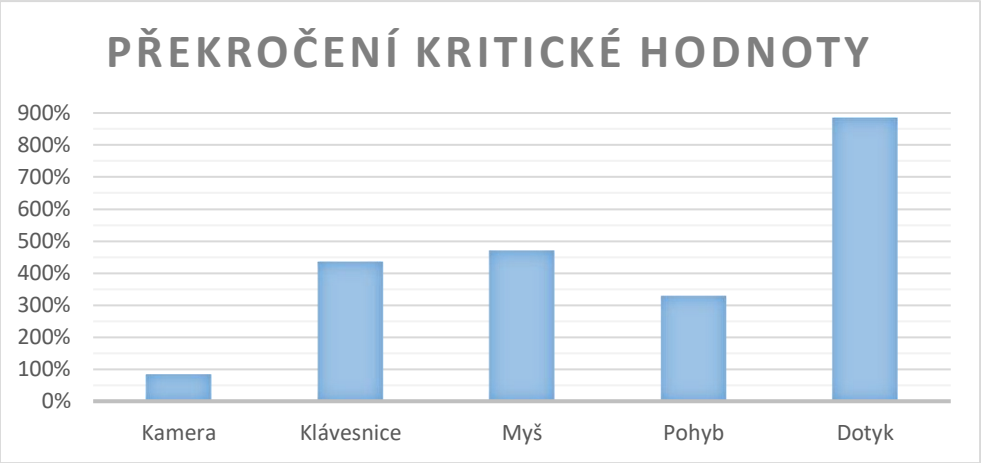
$D_{n,\alpha}$			D_n		**zamít. H_0?**	
	1	0,0642		0,5152		ANO
	2	0,0655		0,0953		ANO
	3	0,0652		0,2217		ANO

Tabulka 11 - Kolmogorov-Smirnovův test – Pohyb

$<u_j;u_{j+1})$		<0;0,1)	<0,1;0,2)	<0,2;0,3)	<0,3;0,4)	<0,4;0,5)	<0,5;0,6)	<0,6;0,7)	<0,7;0,8)	<0,8;0,9)	<0,9;1)
x		**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
n_j	1	27	74	316	378	591	736	531	329	181	96
	2	43	58	281	376	552	501	376	355	252	139
	3	36	129	319	345	463	544	465	375	233	157
N_j	1	27	101	417	795	1386	2122	2653	2982	3163	3259
	2	43	101	382	758	1310	1811	2187	2542	2794	2933
	3	36	165	484	829	1292	1836	2301	2676	2909	3066
Φ(x)	1	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$F_n(x)$	1	0,0083	0,0310	0,1280	0,2439	0,4253	0,6511	0,8141	0,9150	0,9705	1,0000
	2	0,0147	0,0344	0,1302	0,2584	0,4466	0,6175	0,7457	0,8667	0,9526	1,0000
	3	0,0117	0,0538	0,1579	0,2704	0,4214	0,5988	0,7505	0,8728	0,9488	1,0000
$\lvert Φ(x)-F_n(x)\rvert$	1	0,0917	0,1690	0,1720	0,1561	0,0747	0,0511	0,1141	0,1150	0,0705	0,0000
	2	0,0853	0,1656	0,1698	0,1416	0,0534	0,0175	0,0457	0,0667	0,0526	0,0000
	3	0,0883	0,1462	0,1421	0,1296	0,0786	0,0012	0,0505	0,0728	0,0488	0,0000

$D_{n,\alpha}$			D_n		**zamít. H_0?**	
	1	0,0163		0,1720		ANO
	2	0,0164		0,1698		ANO
	3	0,0169		0,1462		ANO

Tabulka 12 - Kolmogorov-Smirnovův test – Dotyk



Graf 8 - Překročení kritické hodnoty - Kolmogorov-Smirnovův test

8.6. Kolísání hodnot

Další důležitou vlastností generátoru náhodných čísel pro otestování, je dostatečně časté střídání hodnot. Tedy že prostřední ze 3 čísel, které po sobě následují, je lokálním extrémem – je buď větší než obě krajní čísla, nebo menší. K tomu byla použita metoda Extremálních bodů (bodů zvratu).

Při výpočtu hodnoty testové statistiky U podle vzorce 3.12. byla použita hladina významnosti $\alpha = 0{,}05$. Počet hodnot n se liší od celkového počtu vygenerovaných čísel, protože pokud se 2 sousedící hodnoty rovnají, pak se jedna z nich odstraní. Hodnota $u_{0{,}975}$ byla zjištěna z matematických tabulek.

Hypotézu H_0, tedy že posloupnost je náhodná, jsme nezamítli u generátoru využívajícího kameru, u dvou měření generátoru využívajícího klávesnici a jednoho měření generátoru s pohybem telefonu.

Při pohledu na Graf 9, kde jsou zobrazeny průměrné odchylky od kritického oboru, zjistíme, že po nejlepším výsledku generátoru s kamerou následuje generátor s klávesnicí, který po zprůměrování také splňuje náhodnost. Výrazně špatně dopadly generátory s myší a dotykem. To lze vysvětlit tím, že při tahu myší nebo prstem dochází k plynulé změně hodnot, a tedy lokálně nevznikají extrémy.

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	576	868	-0,1074	1,96	(-∞;-1,96>∪<1,96; ∞)	NE
2	584	899	-1,1085			NE
3	606	914	-0,1571			NE

Tabulka 13 - Extremální body – Kamera

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	250	354	1,9378	1,96	(-∞;-1,96>∪<1,96; ∞)	NE
2	270	363	3,6606			ANO
3	243	365	0,1245			NE

Tabulka 14 - Extremální body – Klávesnice

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	205	6798	-124,4462	1,96	(-∞;-1,96>∪<1,96; ∞)	ANO
2	215	6710	-123,2714			ANO
3	182	6315	-120,1939			ANO

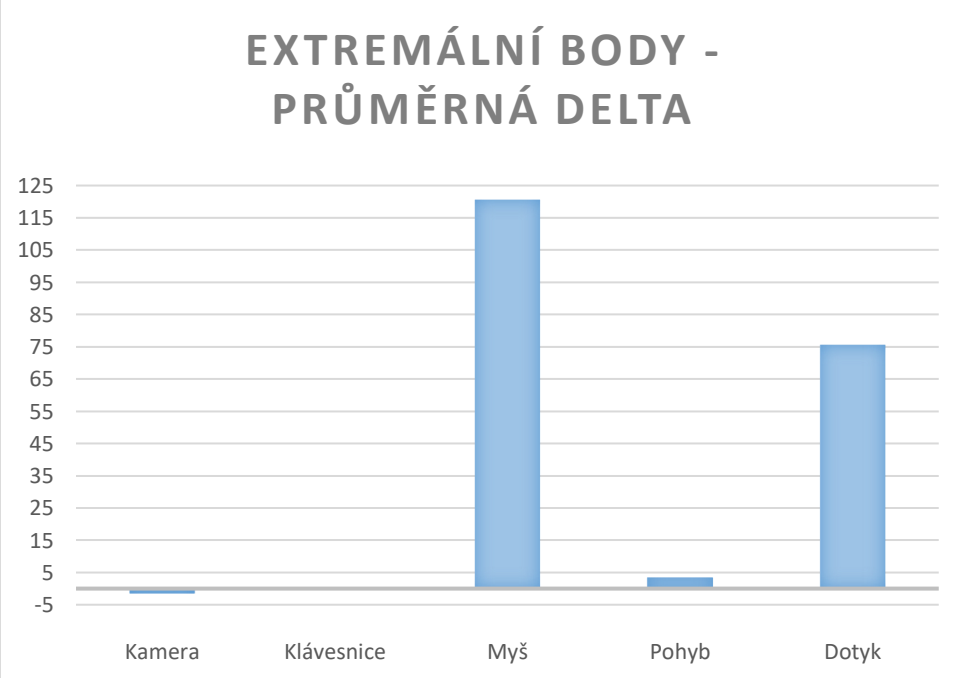
Tabulka 15 - Extremální body – Myš

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	193	299	-0,6879	1,96	(-∞;-1,96>∪<1,96; ∞)	NE
2	133	289	-8,1639			ANO
3	143	298	-7,4876			ANO

Tabulka 16 - Extremální body – Pohyb

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	202	3200	-80,9406	1,96	(-∞;-1,96>∪<1,96; ∞)	ANO
2	214	2874	-75,2617			ANO
3	238	3027	-76,6972			ANO

Tabulka 17 - Extremální body – Dotyk



Graf 9 - Extremální body - průměrná delta

8.7. Existence trendu

Stejně jako nás zajímá časté kolísání hodnot, je nežádoucí, aby v posloupnosti vznikaly trendy – tedy že hodnoty budou delší dobu klesat nebo stoupat.

Pro tento hodnotící aspekt byl využit test znamének diferencí. Při výpočtu testové statistiky byl využit vzorec 3.13. Hladina významnosti byla zvolena $\alpha = 0{,}05$. Množství hodnot v posloupnosti n se opět liší od počtu vygenerovaných hodnot kvůli vyřazení dvou stejných hodnot vedle sebe. Hodnota pro $u_{0{,}975} = 1{,}96$ byla zjištěna z matematických tabulek.

Hypotéza H_0 byla zamítnuta u všech měření generátoru založeném na dotyku displeje. Dále u 2 měření generátoru s kamerou, jednoho měření generátorů s klávesnicí a myší a pohybem.

Při zprůměrování odchylek (Graf 10) od kritického oboru ovšem bylo zjištěno, že hypotéza byla zamítnuta pouze u generátoru využívajícího myš, u ostatních zamítnuta nebyla. Nejlepších výsledků dosáhl generátor využívající klávesnici a generátor s pohybem telefonu (viz Tabulka 19 a Tabulka 21), nejhorších pak s pohybem myši (Tabulka 20).

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	438	868	0,5288	1,96	(-∞;-1,96>∪<1,96; ∞)	NE
2	467	899	2,0785			ANO
3	475	914	2,1186			ANO

Tabulka 18 – Znaménka diferencí – Kamera

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	186	354	1,7466	1,96	(-∞;-1,96>∪<1,96; ∞)	NE
2	182	363	0,1816			NE
3	193	365	1,9918			ANO

Tabulka 19 – Znaménka diferencí – Klávesnice

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	3331	6798	-2,8358	1,96	(-∞;-1,96>∪<1,96; ∞)	ANO
2	3338	6710	-0,6977			NE
3	3143	6315	-0,6102			NE

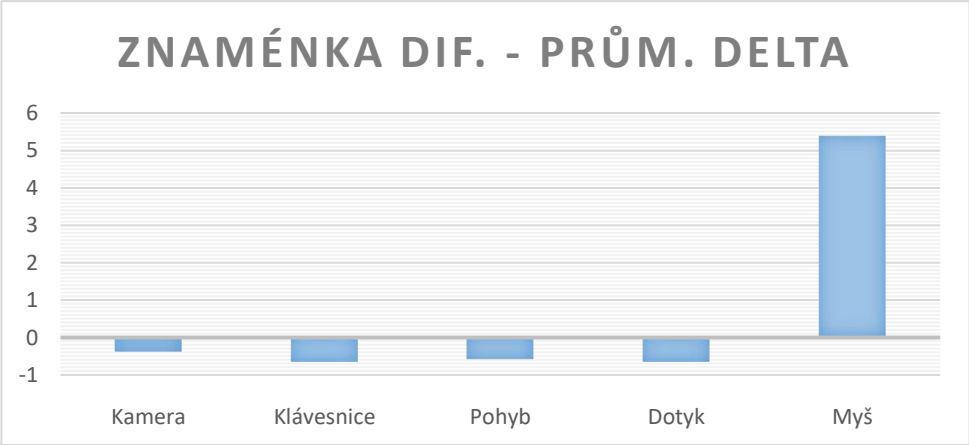
Tabulka 20 – Znaménka diferencí – Myš

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	143	299	-1,2000	1,96	(-∞;-1,96>∪<1,96; ∞)	NE
2	141	289	-0,6103			NE
3	159	298	2,1035			ANO

Tabulka 21 – Znaménka diferencí – Pohyb

Měření	Y	n	U	$u_{0,975}$	W	Zamítnuta H_0
1	1632	3200	1,9899	1,96	(-∞;-1,96>∪<1,96; ∞)	ANO
2	1603	2874	10,7569			ANO
3	1660	3027	9,2540			ANO

Tabulka 22 – Znaménka diferencí – Dotyk



Graf 10 - Znaménka diferencí - průměrná delta

8.8. Opakování hodnot

Jak bylo probíráno v kapitole 3.6.8 Poker test, předchozí testy, především ale Kolmogorov-Smirnovův test, by vyšly s dobrými výsledky i pro posloupnosti, kde se opakují stejná čísla po sobě (nemusí být bezprostředně po sobě jdoucí).

Proto byl použit poker test, který zjišťuje, kolik různých hodnot se nachází v pětici po sobě jdoucích hodnot. Protože tento test porovnává celá čísla, byla generovaná čísla nejprve vynásobena 100x. Počet různých hodnot v pětici je vyjádřen hodnotou r, následně je zjištěn počet jednotlivých skupin (dvojice, trojice, …) a porovnán s referenčním počtem $počet_{exp}$ podle pravděpodobnosti p_{ref} v Tabulka 1 v kapitole 3.6.8 Poker test.

Dále byla vypočítána průměrná odchylka od referenčních hodnot (Graf 11). Nejlépe dopadl generátor založený na pohybu telefonu (Tabulka 26), s nímž měl srovnatelné výsledky generátor využívající klávesnici (Tabulka 24). Nejhůře dopadl generátor založený na pohybu myši, který měl extrémně velké množství pětic (Tabulka 25).

skupina	r	počet	$počet_{exp}$	počet	$počet_{exp}$	počet	$počet_{exp}$	p_{ref}
jednice	5	678	268	657	277	697	286	0,3024
dvojice	4	173	446	211	462	181	476	0,5040
dvě dvojice	3	3	96	10	99	8	102	0,1080
trojice	3	27	64	27	66	38	68	0,0720
trojice a dvojice	2	0	8	0	8	0	9	0,0090
čtveřice	2	0	4	7	4	12	4	0,0045
pětice	1	0	0	1	0	5	0	0,0001

Tabulka 23 - Poker test – Kamera

	r	počet	$počet_{exp}$	počet	$počet_{exp}$	počet	$počet_{exp}$	p_{ref}
jednice	5	241	108	281	111	261	112	0,3024
dvojice	4	102	180	76	184	97	186	0,5040
dvě dvojice	3	6	39	5	40	4	40	0,1080
trojice	3	5	26	0	26	3	27	0,0720
trojice a dvojice	2	0	3	0	3	0	3	0,0090
čtveřice	2	0	2	0	2	0	2	0,0045
pětice	1	0	0	0	0	0	0	0,0001

Tabulka 24 - Poker test - Klávesnice

	r	počet	$počet_{exp}$	počet	$počet_{exp}$	počet	$počet_{exp}$	p_{ref}
jednice	5	1453	2108	1426	2076	1712	1943	0,3024
dvojice	4	2101	3513	2100	3460	1772	3238	0,5040
dvě dvojice	3	946	753	1090	741	965	694	0,1080
trojice	3	333	502	374	494	268	463	0,0720
trojice a dvojice	2	0	63	0	62	0	58	0,0090
čtveřice	2	774	31	696	31	622	29	0,0045
pětice	1	1361	1	1175	1	1081	1	0,0001

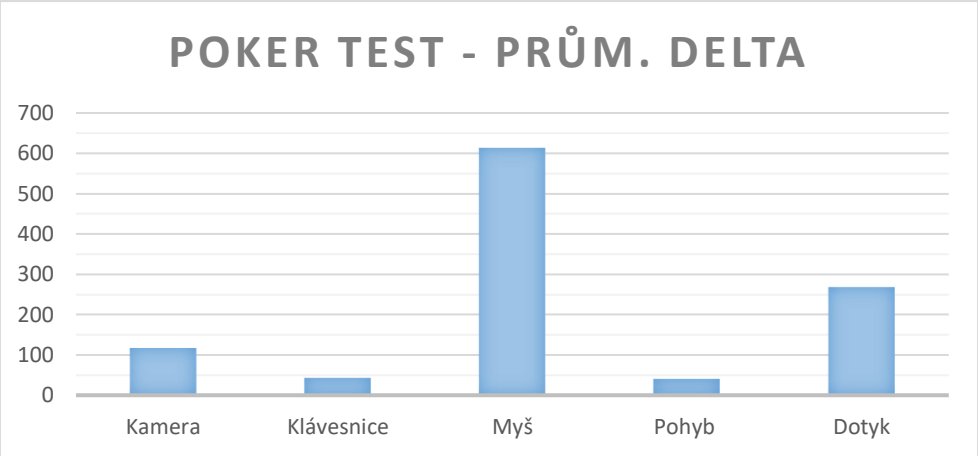
Tabulka 25 - Poker test - Myš

	r	počet	$počet_{exp}$	počet	$počet_{exp}$	počet	$počet_{exp}$	p_{ref}
jednice	5	6	135	12	130	22	131	0,3024
dvojice	4	266	226	258	217	262	219	0,5040
dvě dvojice	3	156	48	145	46	131	47	0,1080
trojice	3	12	32	9	31	11	31	0,0720
trojice a dvojice	2	0	4	0	4	0	4	0,0090
čtveřice	2	4	2	2	2	4	2	0,0045
pětice	1	0	0	0	0	0	0	0,0001

Tabulka 26 - Poker test - Pohyb

	r	počet	$počet_{exp}$	počet	$počet_{exp}$	počet	$počet_{exp}$	p_{ref}
jednice	5	1706	986	1393	887	1561	927	0,3024
dvojice	4	801	1643	682	1478	766	1545	0,5040
dvě dvojice	3	258	352	216	317	202	331	0,1080
trojice	3	246	235	241	211	252	221	0,0720
trojice a dvojice	2	0	29	0	26	0	28	0,0090
čtveřice	2	165	15	210	13	193	14	0,0045
pětice	1	79	0	187	0	88	0	0,0001

Tabulka 27 - Poker test – Dotyk



Graf 11 - Poker test - Průměrná delta

8.9. Souhrnné hodnocení

Podle výsledku předchozích testů je patrné, že pro každou metodu hodnocení dat vychází jako vhodný generátor jiný prototyp. Nejrychlejším generátorem je prototyp s pohybem myši, nejvyšší entropii má prodleva kláves, nejblíže normálnímu rozdělení je a zároveň nejlepší kolísání hodnot splňuje prototyp s kamerou a poker test vychází nejlépe generátoru s pohybem telefonu.

Ačkoliv toto zjištění může být matoucí, není tomu tak. Jak již bylo zmiňováno v kapitole 3.6 Statistické hodnocení náhodnosti čísel, každá metoda může poukázat na slabiny jednotlivých prototypů, její výsledky však nemohou být interpretovány jako potvrzení kvality generátoru. Navíc, ačkoli Kolmogorov-Smirnovův test, extremální body a znaménka diferencí jsou testy určené k zamítnutí nulové hypotézy, že posloupnost je náhodná, každý posloupnost hodnotí z jiného pohledu. Jak je zřejmé z dílčích výsledků testů jednotlivých generátorů, zatímco jeden test náhodnost zamítne, druhý ji nevylučuje.

Abychom mohli kvalifikovaně rozhodnout, zda generátor je, či není kvalitní, je potřeba 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 postačuje). Pro rychlost generování byla jako nejlepší hodnota určena ta nejvyšší, stejně tak pro entropii informace. Naopak pro překročení kritické hodnoty u Kolmogorov-Smirnovova testu, extremálních bodů a znamének diferencí byla jako nejlepší ohodnocena nejnižší hodnota. Tato hodnocení uvádí Tabulka 28.

	Kamera	Klávesnice	Myš	Pohyb	Dotyk
Rychlost generování	3	5	1	4	2
Entropie	5	1	4	2	3
K-S test	1	3	4	2	5
Extremální body	1	2	5	3	4
Znaménka diferencí	4	2	5	3	1
Poker test	3	2	5	1	4

Tabulka 28 - Hodnocení výsledků generátorů na školní stupnici

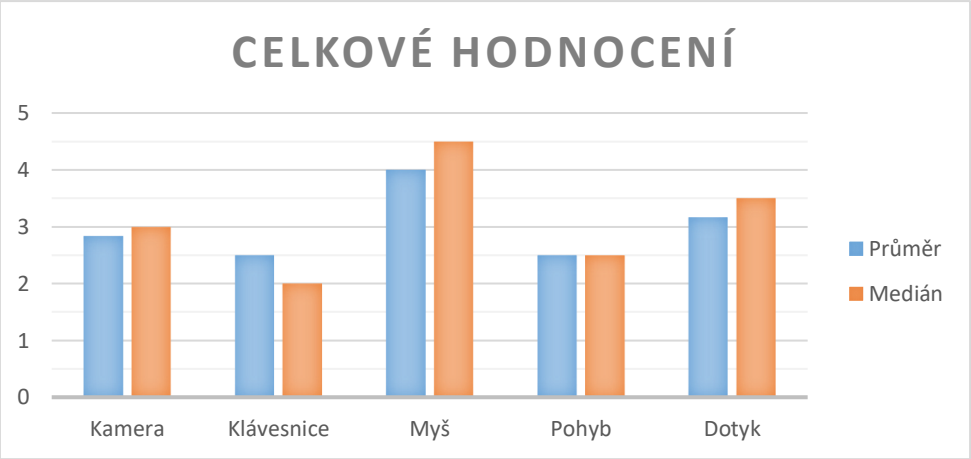
Následně byl z hodnocení prototypů vypočítán aritmetický průměr a medián, které zobrazuje Graf 12.

Z něj nám jako nejlepší prototyp generátoru náhodných čísel vychází ten vytvořený pomocí hodnot generovaných na základě časových prodlev mezi stiskem kláves.

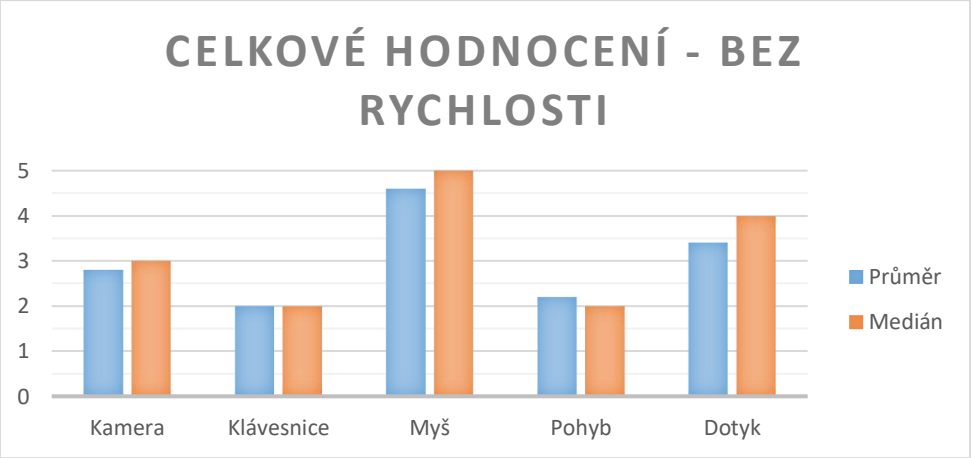
Tato metoda se jeví jako výhodná i s ohledem na minimální zatížení uživatele – stačilo by aplikaci, využívající tohoto generátoru, zapnout na pozadí běhu systému a při běžném používání počítače nechat vygenerovat velké množství náhodných čísel, které by následně byly užity pro kryptografické potřeby.

Se stejným průměrem, jako generátor na bázi klávesnice dopadl generátor na bázi pohybu telefonu. Ten má navíc výhodu oproti předchozímu, že uživatel na telefonu nemusí nic aktivně provádět – stačí ho nechat v kapse a telefon bez zásahu uživatele generuje náhodná čísla.

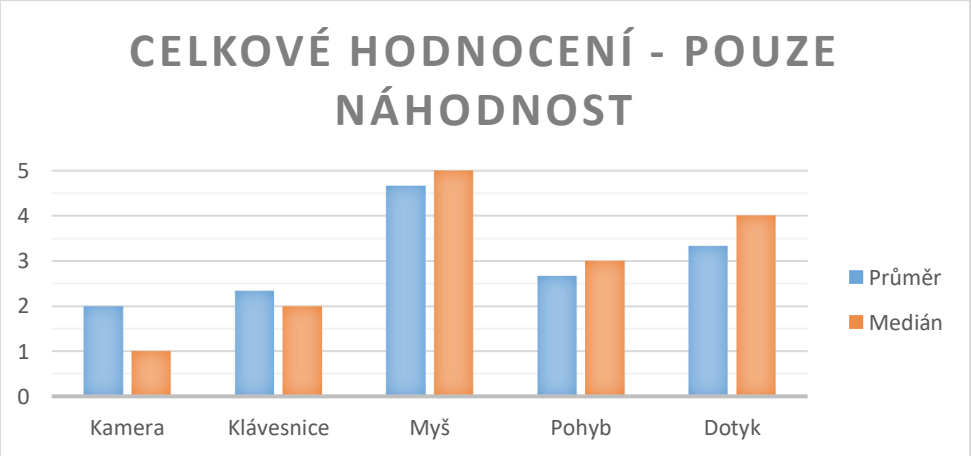
Oba tyto prototypy by si navíc vylepšily výslednou známku (medián by byl roven 2), pokud bychom nezohledňovali rychlost generování (Graf 13). A pokud by hodnotícím kritériem byla pouze náhodnost, tedy Kolmogorov-Smirnovův test, extremální body a znaménka diferencí, pak by nejlepším generátorem byl prototyp na bázi kamery (medián 1, průměr 2).



Graf 12 - Celkové hodnocení výsledků generátorů na školní stupnici



Graf 13 - Hodnocení výsledků generátorů na školní stupnici bez zohlednění rychlosti



Graf 14 – Hodnocení výsledků generátorů na školní stupnici – pouze náhodnost

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 mu představeny způsoby hodnocení náhodných posloupností. Dále jsou stručně popsána stávající řešení generátorů náhodných čísel, která jsou k dispozici a technologie použité pro vytvoření prototypů generátorů náhodných čísel.

V praktické části práce bylo vytvořeno 5 generátorů fungujících na smartphonu a počítači. Pro smartphone byl zvolen OS Android, pro desktopové verze pak 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 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, shody rozložení, kolísání hodnot, existence trendu a opakování hodnot.

Ačkoliv je generování pomocí pohybu myši na obrazovce v praxi poměrně často využíváno, v celkovém hodnocení absolutně propadlo. Naopak pohyb telefonem, který je využíván naprosto minimálně, vykazuje velmi dobré vlastnosti. Navíc, jak bylo zmíněno v souhrném hodnocení v kapitole 8.9, tento generátor nepotřebuje, aby byl telefon aktivně uživatelem používán. Nejlépe však dopadl generátor s klávesnicí, který s výjimkou rychlosti dopadl ve všech hodnotících testech velmi solidně.

Autor práce si nekladl za cíl vytvořit plnohodnotný generátor náhodných čísel, ale snažil se poukázat na slabé stránky jednotlivých implementací, a umožnil zjednodušit

analýzu dalším autorům, kteří by hledali nejvhodnější metodu sběru náhodných hodnot pro využití v generátoru náhodných čísel.

Výstupy této práce budou použity pro další autorovo studium generátorů náhodných čísel a při práci na implementaci již v běžné praxi použitelného generátoru.

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, implementace Vernamovy šifry a kvantifikátory pro testy extremálních bodů a znamének diferencí
- 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. Kupča, Vojtěch. Teorie a perspektiva kvantových počítačů. *Matematická sekce.* [Online] Matematicko-fyzikální fakulta Univerzita Karlova, 2001. [Citace: 6. 11 2018.] http://www.karlin.mff.cuni.cz/~holub/soubory/qc/.

8. Kelsey, John, a další. Cryptanalytic Attacks on Pseudorandom Number Generators. [Online] [Citace: 11. 11 2018.] http://www.schneier.com/paper-prngs.pdf.

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

10. Saiko, Dušan. Všeobecně používaná asymetrická šifra a Vernamovo XOR šifrování. *iDnes.cz Blog.* [Online] 8. 03 2011. [Citace: 24. 11 2018.] https://saiko.blog.idnes.cz/blog.aspx?c=180687.

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

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

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

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

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

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

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

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

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

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

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

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

23. 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.

24. 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.

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

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

27. 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.

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

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

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

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

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

33. GSM Arena. OS (Operating System) - definition. *GSM ARENA.* [Online] 2000-2018. [Citace: 24. 11 2018.] https://www.gsmarena.com/glossary.php3?term=os.

34. 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.

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

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

37. 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.

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

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

40. 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.

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

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

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

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

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. 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.

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

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