Hashovací funkce založená na kvazigrupách

Jiří Dvorský, Eliška Ochodková, Václav Snášel

Katedra Informatiky
Fakulta elektrotechniky a informatiky
VŠB-TU Ostrava
708 33 Ostrava, Česká republika
E-mail: jiri.dvorsky@vsb.cz, eliska.ochodkova@vsb.cz, vaclav.snasel@vsb.cz

Abstrakt

V tomto článku se zabýváme možnostmi využití neasociativní algebraické struktury kvazigrupa v kryptografii, a to zejména pro konstrukci hashovacích funkcí. Je zde prezentován algoritmus pro generování velkých kvazigrup. Tento algoritmus umožňuje reprezentovat velké kvazigrupy bez nároků na paměť.

Klíčová slova: kvazigrupy, Latinské čtverce, lupa, homotopie, hashovací funkce, jednocestné funkce, analytická kvazigrupa.

1 Úvod

Hashovací funkce jsou matematické funkce, které se v informačních technologiích používají velice často, jsou to funkce, které mapují vstup proměnné délky na výstup pevné délky. Jako příklad lze uvést databázové zpracování, kdy technika hashování umožňuje na základě hodnoty vyhledávacího klíče určit umístění odpovídajícího záznamu. Kryptografické hashovací funkce jsou důležitými nástroji pro kryptografické aplikace jakými jsou například digitální podpis, klíčované hashovací funkce (MAC – Message Authentication Code). Používají se k zajištění důležitých bezpečnostních aspektů současných informačních technologií jakými jsou zejména integrita (neporušenost), autentizace a nepopiratelnost viz [9,14].

Hashovací funkce jsou většinou iterační procesy, které hashují vstup proměnné délky (t-bitové bloky) na výstup pevné délky (r-bitové bloky). Vstup M je nejprve zarovnán na násobek délky bloku a poté je rozdělen na t-bitové bloky M_i. Hashovací funkce H může být popsána takto: $MD_0=IV$, $MD_i = f(MD_{i-1}, M_i)$, $1 \le i \le n$, $H(M)=MD_n$, kde IV je inicializační hodnota, f je kompresní funkce, MD_i jsou r-bitové buffery.

V tomto příspěvku budeme prezentovat novou metodu pro výpočet hashovací funkce. Tato metoda je založena na teorii kvazigrup. Novým výsledkem je pak, realizace velké kvazigrupy pomocí výpočetní metody nevyžadující žádnou pomocnou paměť. Tím se náš postup liší od známých metod využití kvazigrup v kryptografii viz [4,7,8,13].

2 Hashovací funkce

2.1 Nezbytné definice

Definice 1: Funkce $H()$ se nazývá *jednocestnou funkcí* (JF), jestliže splňuje následující vlastnost:

1. Pro daný vstup M a danou funkci H je snadné vypočítat $H(M)$, ale pro danou $H(M)$ je nesnadné vypočítat M.

Definice 2: Jednocestná funkce $H()$ se nazývá *jednocestnou funkcí se zadními vrátky* (JFZV), jestliže navíc splňuje následující vlastnost:

1. Pro danou $H_K(M)$ lze snadno spočítat M tehdy, když je známa hodnota tajemství K.

Definice 3: Funkce *H()* se nazývá *jednocestnou hashovací funkcí* (JHF), jestliže splňuje následující vlastnosti:

1. Argument (zpráva) *M* může mít libovolnou délku
2. Výsledek *H(M)* má pevnou délku (*charakteristika*)
3. Funkci *H(M)* lze relativně snadno realizovat jak hardwarově, tak softwarově
4. Funkce *H(M)* je jednocestná, tj. pro daný vstup *M* a danou funkci *H* je snadné vypočítat *H(M)*, ale pro danou *H(M)* je nesnadné vypočítat *M*.

Definice 4: Jednocestná hashovací funkce *H()* se nazývá *jednocestnou hashovací funkcí odolnou kolizím* (JHFOK), jestliže splňuje následující vlastnosti:

1. Funkce *H(M)* je odolná proti kolizím, tj. pro daný vstup *M* a danou hodnotu *H(M)* je těžké (nemožné) najít takové *M'* ($M \neq M'$), aby platilo *H(M) = H(M')*
2. Funkce *H(M)* je silně odolná proti kolizím, je-li těžké (nemožné) najít jakýkoliv pár *M, M'* (pro $M \neq M'$) takový, aby platilo *H(M) = H(M')*.

Definice 5: Nechť je dán klíč *K* pevné délky a vstup (zpráva) *M* proměnlivé délky. Jednocestná hashovací funkce *H()* se nazývá *klíčovanou (jednocestnou) hashovací funkcí (odolnou kolizím)* (KJHFOK), jestliže splňuje následující vlastnosti:

1. Pro dané *H, K* a *M* je snadné vypočítat *H(K, M)*
2. Bez znalosti *K* je těžké najít *M* pro dané *H(K, M)*
3. Bez znalosti *K* je těžké najít *M, M'* ($M \neq M'$) takové, aby platilo *H(K, M)=H(K, M')*
4. K (mnoha) daným párům [M_i, *H(K,* M_i*)*] je těžké nalézt tajný klíč *K*
5. Bez znalosti *K* je těžké najít *H(K, M)* pro jakékoliv *M*, a to i tehdy, jestliže je dáno mnoho párů [M_i, *H(K,* M_i*)*], kde $M, \neq M_i$, pro každé M_i.

2.2 Typy hashovacích funkcí

V této kapitole je uveden stručný výčet různých typů hashovacích funkcí (podrobnosti o těchto typech funkcí přesahují rámec článku, ale lze je najít např. v [5]).

1. Hashovací funkce založené na blokových šifrách (např. algoritmus DES v CBC režimu)
2. Hashovací funkce založené na modulární aritmetice: bezpečnost kryptosystémů založených na modulární aritmetice plyne z výpočetní složitosti těžkých problémů – problému faktorizace a problému diskrétního logaritmu)
3. Hashovací funkce založené na automatech
4. Hashovací funkce založené na problému batohu
5. Speciální hashovací funkce - mezi nejznámější a nejpoužívanější patří MD5, SHA-1, RIPEMD-160. Tyto hashovací funkce zpravidla vstupní zprávu *M* dělí na bloky velikosti 512 bitů a poskytují výstup (charakteristiku) *H(M)* velikosti buď 128 nebo 160 bitů (v případě RIPEMD-160 se je možnost získat charakteristiku dvojnásobné velikosti 320 bitů). Jádrem kompresní funkce *f* jsou obvykle bitové operace jako XOR, b. negace, b. součin, b. součet, b. negace, b. posuv a b. rotace.

2.3 Útoky proti hashovacím funkcím

V této kapitole jsou uvedeny některé typy útoků na hashovací funkce (podrobněji opět viz. [5]). Útoky se dělí na dvě skupiny – na útoky závisící na algoritmu a na útoky na algoritmu nezávisící.

1. Útok hrubou silou – hledání zpráv se stejnou hodnotu *n*-bitové charakteristiky *h*, tj. např. Pro *n=128* to znamená 2^{128} pokusů nalézt zprávu *M* se stejnou charakteristikou *h*
2. Narozeninový útok (Birthday attack) – útok vycházející z tzv. narozeninového paradoxu (jaká je pravděpodobnost, že se v místnosti mezi *n* lidmi najde alespoň jedna dvojice osob se stejným datem narození (den, měsíc), pro *n=23* je tato pravděpodobnost *p=0.507*, pro *n=30* je *p=0.706*). Pro např. Pro *n=128* útok spočívá v generování 2^{64} zpráv *M1* a 2^{64} podvržených zpráv *M2* a vyhledání takové *M2*, že *H(M1) = H(M2)*.

3. Náhodný útok (Random Attack) – útočník vybere náhodnou zprávu a doufá, že její charakteristika je stejná jako ta pravá.
4. Pseudo Attack – útočník se snaží útočit na klíčovanou hashovací funkcí pomocí podvrženého klíče *K'*
5. Spaciální útoky (diferenční kryptoanalýza, lineární kryptoanalýza, Meet-in-the-Middle Attack, Fixed Point Attack,…)

3 Konstrukce hashovací funkce pomocí kvazigrupy

V této kapitole uvedeme základní pojmy z teorie kvazigrup a způsob konstrukce kvazigrup. Tyto pojmy je možno najít v [3,6,12].

Definice 6. Grupoid *(Q, *)* se nazývá kvazigrupou (algebrou s jednou binární operací), jestliže splňuje podmínku:

(1) $(\forall u, v \in Q)\ (\exists! x, y \in Q)\ (u * x = v \ \wedge\ y * u = v)$.

Z toho vyplývá:

1. $x * y = x * z \ \vee\ y * x = z * x \ \Rightarrow\ y = z$
2. Rovnosti $a * x = b$, $y * a = b$ mají stejná řešení x, y pro každé $a, b \in Q$.

Definice 7. Nechť $A = \{a_1, a_2, ..., a_n\}$, je abeceda. Latinský obdélník velikosti k x n je matice s prvky $a_{ij} \in A, i = 1,2,...,k, j = 1,2,...,n$, taková, že každý její prvek se nachází v každém řádku a každém sloupci právě jednou. Jestliže $k=n$ hovoříme o Latinském čtverci (dále LS). Říkáme, že Latinský čtverec je ve standardní formě (redukovaný), jestliže první řádek a nejlevější sloupec jsou určitým způsobem uspořádány, nejčastěji v alfabetickém pořadí. Dva LS *A, B* řádu *n* jsou ortogonální, jestliže pro každou uspořádanou dvojici *(k,l)* čísel z množiny *{1, 2, ..., n}*existuje právě jediná dvojice indexů *(i,j)* splňující $a_{ij}=k$, $b_{ij}=l$.

Definice 8. Kvazigrupa *(Q, *)* se nazývá lupou, jestliže splňuje podmínku:

(1) $(\exists e \in Q)\ (\forall x \in Q)\ (e * x = x * e = x)$.

Prvek *e* se nazývá jednotkový prvek.

Věta 1. Je-li kvazigrupa *(Q, *)* asociativní pak obsahuje jednotkový prvek a je tudíž grupou. Důkaz viz [3] V.D. Belousov.

Připomínáme, že kvazigrupy jsou ekvivalentní známějším latinským čtvercům. Tabulka násobení nad kvazigrupou řádu *n* je Latinský čtverec řádu *n* a naopak, každý LS řádu *n* je tabulkou násobení pro řádu *n*.

Tabulka 1. Počet všech různých redukovaných Latinských čtverců řádu *n* je vyčíslen pro $n \leq 10$.

n	**L_n**
1	1
2	1
3	1
4	4
5	56
6	9,408
7	16,942,080
8	535,281,401,856
9	377,597,570,964,258,816
10	7,580,721,483,160,132,811,489,280

Problém určení přesného počtu všech LS řádu *n>10* stále vyřešen není. Existuje však nejméně $n!\ (n-1)!...2!$ LS řádu *n*. Pokud $A=\{0…255\}$, potom existuje $256!\ 255!\ ...2! > 10^{58000}$ kvazigrup. Využití Latinských čtverců v kryptografii bylo popsáno například v [1,2,11].

Velmi důležitá vlastnost násobení v kvazigrupách je následující:

Je možno ukázat, že každý prvek kvazigrupy *Q* řádu *n* se vyskytuje právě *n*-krát mezi všemi dvouprvkovými součiny prvků z *Q*, n^2-krát mezi všemi tříprvkovými součiny prvků z *Q* atd. až n^{t-1}

mezi všemi *t*-prvkovými součiny prvků z *Q*. Existuje n^t možných uspořádaných součinů *t* prvků z *Q*, to znamená, že všechny prvky se vyskytují stejně často (se sejnou pravděpodobností) mezi těmito n^t součiny viz [10].

Předcházející tvrzení lze neformálně formulovat asi takto. V případě, že kvazigrupa nebude splňovat „mnoho identit“ budeme násobením získávat mnoho různých výsledků. To znamená pokud není splněna komutativita, asociativita existence jednotkového prvku apod.

Definice 9. Hashovací funkce nad kvazigrupou *(Q, *)* se nazývá zobrazení $H_a()$ definované následujícím předpisem:

$H_a(q_1 q_2 ... q_n) = ((...(a * q_1) * q_2) * ...) * q_n$

Prvek *a* je pevně zvolený prvek z kvazigrupy *(Q, *)*.

Neformálně řečeno hodnotu hashovací funkce vypočteme tak, že vynásobíme postupně všechny znaky ze kterých se skládá slovo.

Příklad 1. Tabulka kvazigrupy modulárního odčítaní vypadá následovně:

0	3	2	1
1	0	3	2
2	1	0	3
3	2	1	0

To že tato tabulka definuje kvazigrupu je dáno tím, že splňuje podmínky pro Latinský čtverec, to znamená, že každý prvek se v řádku a sloupci vyskytuje právě jednou.

Násobení v této kvazigrupě je definováno předpisem *a * b =(a + 4 – b) mod 4*.

Je zřejmé, že tato kvazigrupa není komutativní *1*2 = 3, 2*1=1*. Neobsahuje jednotkový prvek a tudíž není ani asociativní. Kdyby byla asociativní musí obsahovat i jednotkový prvek, protože by byla grupou.

Hodnota hashovací funkce $H_2(0013) = (((2 * 0) * 0) * 1) * 3 = 2$

Definice 10. Kvazigrupy *(Q, *)* a *(R, *)* se nazveme homotopické jestliže existují permutace π, ω, ρ takové, že platí :

(2) $(\forall u, v \in R)\ (u * v = \pi(\omega(u) * \rho(v)))$.

Homotopii kvazigrup si můžeme snadno představit jako permutování sloupců a řádků tabulky násobení kvazigrupy.

Příklad 2. Tabulka kvazigrupy homotopická s kvazigrupou modulárního odčítaní:

0	3	2	1
2	1	0	3
1	0	3	2
3	2	1	0

Tato tabulka vznikla prohozením druhého a třetího řádku v tabulce násobení kvazigrupy modulárního odčítaní. Permutace π, ρ jsou identity a $\omega = [0213]$. Například $1 * 0 = \omega(1) * 0 = 2 * 0 = 2$.

Tento příklad je možno považovat za návod jak konstruovat nové kvazigrupy.

Déle budeme používat kvazigrupy homotopické kvazigrupě modulárního odčítání. Tyto kvazigrupy můžeme generovat tak, že vygenerujeme tři náhodné permutace a pomocí těchto permutací upravíme tabulku násobení. Takto vzniklé kvazigrupě budeme říkat **tabulková kvazigrupa**. Nevýhodou tohoto postupu je velká paměťová náročnost. Je potřeba uložit n^2 **prvků**.

Homotopie nám dává možnost výsledek násobení v kvazigrupě počítat analyticky a to tak, že zvolíme předpis pro výpočet permutací π, ω, ρ. Násobení v této kvazigrupě je definováno takto: $a * b = \pi((\omega(a) + n - \rho(b)) \bmod n)$.

Výpočet permutací jsme provedli tak, že jsme posloupnost n prvků rozdělili na několik částí a ty jsme rotovali v různých směrech. Pro názornost uvádíme ukázku kódu pro generovaní těchto permutací. Procedura P1 je vlastně výpočet permutace ω ta znamená $\omega(x) = P1(x)$. Podobným způsobem jsou realizovány i další dvě permutace.

```
const unsigned int cQuasiGroupA2::P1(unsigned int x) const
{
      unsigned int Dimension2 = m_Dimension / 2;
      if (x < Dimension2 * 2)
      {
            if (x & 1)
            {
                  x = 2 * ((x / 2 + 1) % Dimension2) + 1;
            }
            else
            {
                  x = 2 * ((x / 2 + Dimension2 - 1) % Dimension2);
            }
      }
      return x;
}
```

Výše uvedený postup nám dává možnost pracovat s kvazigrupami velkých rozměrů. Dosud známé práce využívaly pouze malých kvazigrup realizovaných pomocí tabulky umístěné v operační paměti. Hypotézy uvedené v předcházející části jsme o věřili experimentálně. Výsledky experimentů budeme prezentovat v následující části.

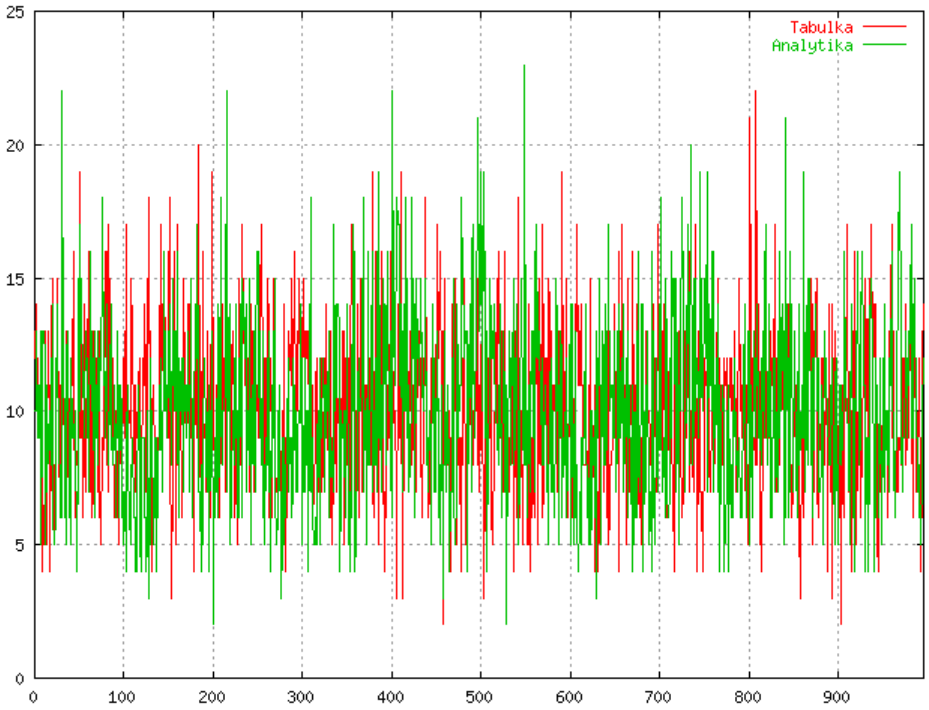
4 Experimentální výsledky

Pro ověření našich teoretických předpokladů byla vytvořena jednoduchá aplikace poskytující jisté experimentální výsledky. Vstupem do této aplikace je vždy množina slov extrahovaných z určitého textu. Prvním takovým textem je soubor bible.txt (dále jen *bible*) z tzv. Canterbury Corpusu (sekce Large files). Tento korpus je určen pro testování kompresních algoritmů. Soubor *bible* obsahuje anglicky psanou Bibli a má délku cca 4 MB. Druhým takovým souborem je elektronická podoba článků z deníku Los Angeles Times (dále jen *latimes*), ročníky 1989 a 1990, s délkou cca 450 MB. Tyto články jsou částí korpusu určeného pro testování fulltextových vyhledávacích systémů. Ze souboru *bible* bylo extrahováno 10 tisíc různých slov, ze souboru *latimes* to bylo 200 tisíc unikátních slov.

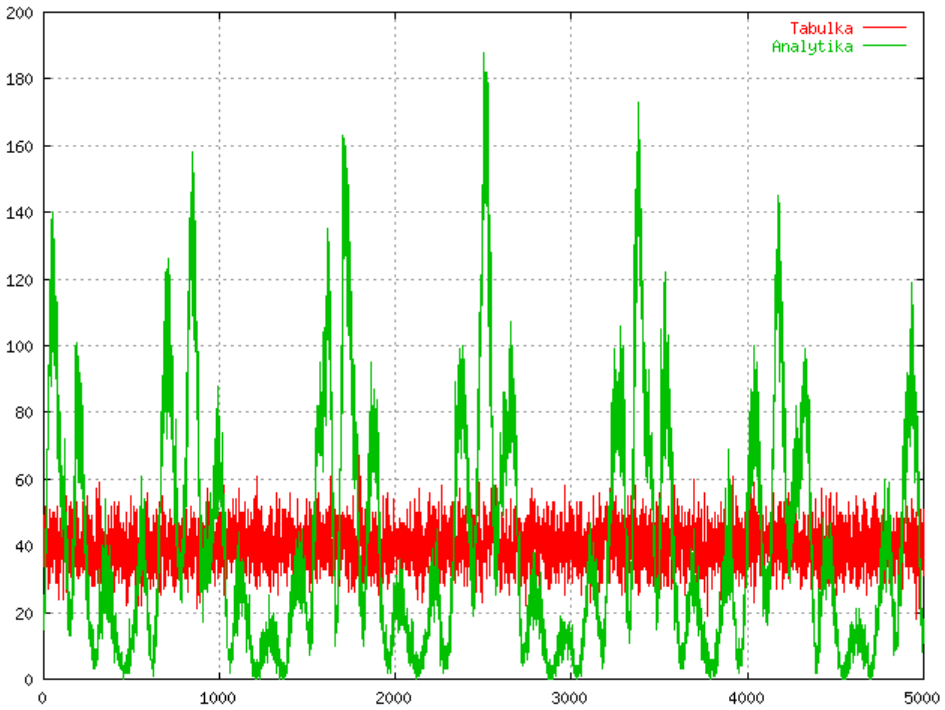
Abychom získali statistické údaje o rozložení klíčů v prostoru hashovacích hodnot, vytvoříme si pomyslnou hashovací tabulku, do které zaznamenáváme informace o hashovaných klíčích (slovech). Naše testy sledují několik parametrů:

1. střední kvadratickou odchylku mezi počty klíčů se stejnou hashovací hodnotou pro námi zvolenou hashovací funkci a ideálním případem hashování tj. rovnoměrným rozdělením hashovaných klíčů v závislosti na měnící se velikosti tabulky, počty klíčů v jednotlivých slotech tabulky,
2. histogram rozdělení délek slotů.

Rozdělení klíčů do jednotlivých slotů v pomyslné hashovací tabulce je znázorněno v následujících grafech.

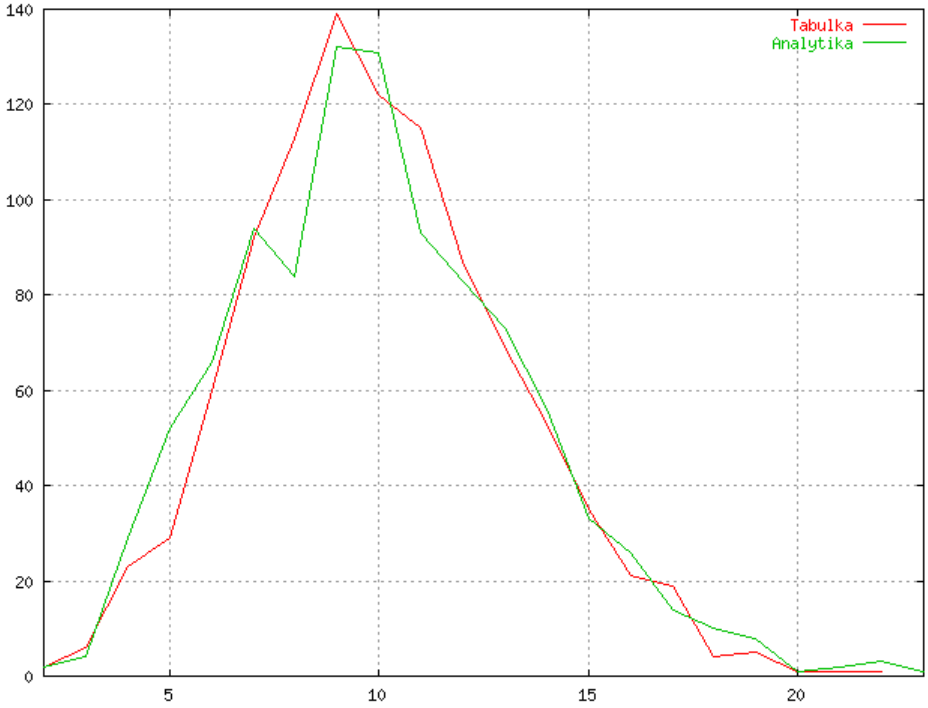


Graf 1 Rozdělení klíčů v jednotlivých slotech pro soubor *bible*; velikost hashovací tabulky 997

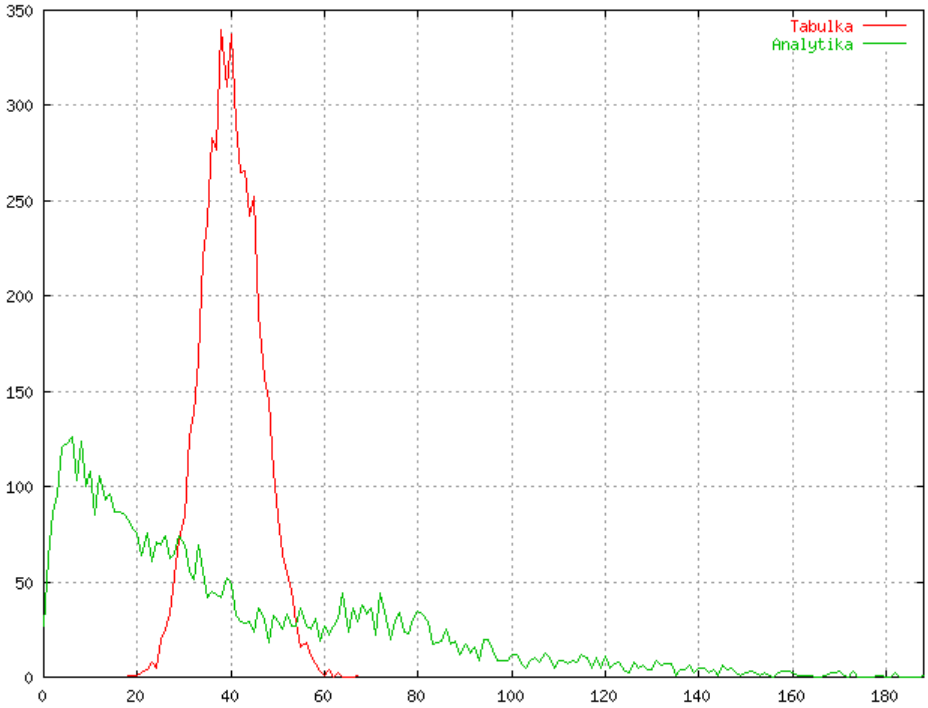


Graf 2: Rozdělení klíčů v jednotlivých slotech pro soubor *latimes*; velikost hashovací tabulky 5003

Z grafu 1 je patrno, že rozložení klíčů v jednotlivých slotech pomyslné tabulky je vcelku rovnoměrné. A to jak pro kvazigrupu generovanou pomocí tabulky, tak pro kvazigrupu vyjádřenou analyticky. V grafu 2 jsou již ale patné odchylky analytické kvazigrupy od kvazigrupy generované tabulkou. Anomálie v rozložení dokonce mají pravidelný tvar. Tento nedostatek je způsoben konstantními parametry permutací v analytické kvazigrupě.



Graf 3: Histogram rozdělení délek slotů; soubor *bible*, velikost tabulky 997



Graf 4: Histogram rozdělení délek slotů; soubor *latimes*, velikost tabulky 5003

Na grafech 3 a 4 můžeme sledovat histogramy délek slotů pro stejná nastavení jako v grafech 1 a 2. Pro soubor bible je dosaženo dobré shody mezi analytickou kvazigrupou a generovanou. V případě druhém tomu již tak není.

Z experimentů vyplývá nutnost měnit parametry permutací v analytické kvazigrupě na základě velikosti používané kvazigrupy.

5 Závěr

V článku jsme ukázali, že použití neasociativních algebraických struktur pro návrh hashovacích funkcí je poměrně zajímavé. Tato práce je pokračováním publikací [15,16].

Stručný nástin důkazu složitosti výpočtu hodnoty hashovací funkce pomocí analytické kvazigrupy.

Má-li kvazigrupa *n* prvků budeme počítat tří permutace a to tak, že posloupnosti o *n* prvcích rozdělíme na *log(n)* částí. Pro výpočet jednoho násobení budeme potřebovat *3 * 5 + 2* operace (násobení, sčítání a dělení). To znamená, že pro vstup o *m* znacích budeme potřebovat

m(3 * 5 + 2)* operací.

Ukazuje se, že časová náročnost výpočtu hodnoty hashovací funkce je ze třídy *O(m)* kde *m* je délka řetězce pro který budeme počítat hodnotu hashovací funkce.

Je zřejmé, že pro velká čísla bude potřeba pro výpočet hodnoty hashovací funkce použít aritmetiku pro práci s velkými čísly.

V další práci bychom chtěli využít neasociativní strukturu neofield viz [13], která je neasociativním ekvivalentem pole.

6 Literatura

[1] S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk, „A Message Authentication Code based on Latin Squares“, proc. Australasian Conference on Information Security and Privacy, Sydney, 1997, Lecture Notes in Computer Science, vol. 1270, Springer, 1997.

[2] D.R. Stinson, R. Wei, L. Zhu, „New Construction for Perfect Hash Familes and Related Structures using Combinatorial Designs and Codes“, Journal of Combinatorial Designs Vol. 8, No. 3, pp. 189-200, 2000.

[3] V.D. Belousov, „Osnovi teorii kvazigrup i lup”, Nauka, Moscow, 1967.

[4] J. Denes, A.D. Keedwell, „A New Authentication Scheme based on Latin Squares”, Discrete Mathematics, no. 106/107, pp. 157-161, 1992.

[5] Bakhtiari, Safavi-Naini, Pieprzyk, „Cryptographic Hash Functions: A Survey”, Technical Report 95-09, Department of Computer Science, University of Wollongong, Australia.

[6] Dénes, J., and Keedwell, A.D.: Latin Squares and their Applications, Akadémiai Kiadó, Budapest; Academic Press, New York, 1974.

[7] Koscielny, C.: A method of Constructing Quasigroup-Based Stream Cipher, Applied Mathematics and Computer Science, vol. 6 No.1, pp. 109-121, 1996.

[8] Koscielny, C., and Mullen, G.L.: A Quasigroup-Based Public-Key Cryptosystem, Int. J. Appl. Math. And Comp. Sci, vol. 9 No.2, pp. 101-109, 1999.

[9] Schneier, B.: Applied Cryptography, John Wiley & Sons, Inc., New York, 1996.

[10] Keedwell, A.D.: Construction, Properties and Applications of Finite Neofields, Commentationes Mathematicae Universitatis Carolinae, vol. 41 No. 2, pp. 283-297, 2000.

[11] McKay, B, and Rogoyski, E.: Latin Square of Order 10, Electronic Journal of Combinatorics, vol.2, 1995, http://www.combinatorics.org/volume_2/cover.html.

[12] Hall, M.: Combinatorial theory, Blaisdell Publishing Company, Massachusetts, 1967.

[13] A.D. Keedwell, „Crossed-inverse quasigroups with long inverse cycles and applications to cryptography”, Australasian Journal of Combinatorics, vol. 20, pp.241-250, 1999.

[14] W. Stallings, „Cryptography and Network Security“, 2nd ed, Prentice Hall, New Jersey, 1999.

[15] E.Ochodková, V.Snášel.Using quasigroups for secure encoding of file system.Security and Protection of Information, International Scientific NATO PfP/PWP Conference, Brno 2001.

[16] E.Ochodková, V.Snášel. CRYPTOGRAPHIC ALGORITHMS WITH AN UNIFORM STATISTICS. Conference NATO Warszawa 2001.