Študentská vedecká a odborná činnosť Aplikovaná informatika ŠVOČ 2010
Fakultné kolo, 21. apríl 2010
Fakulta elektrotechniky a informatiky STU Bratislava

KRYPTOANALÝZA KVÁZIGRUPOVEJ HAŠOVACEJ FUNKCIE

Ivana Slaminková, Ing. Milan Vojvoda, PhD.[1]
Fakulta elektrotechniky a informatiky STU Bratislava
Katedra aplikovanej informatiky a výpočtovej techniky
ivana.slaminkova@gmail.com

Abstrakt

Táto práca je zameraná na kryptoanalýzu hašovacej funkcie založenej na kvázigrupách, ktorá bola prezentovaná v [2], [3]. Článok nadväzuje na prácu [6], ktorá sa zaoberá analýzou použitia kvázigrupy modulárneho odčítania. Skúmanou oblasťou je použitie kvázigrupy izotopnej s kvázigrupou modulárneho odčítania. V práci sa zaoberáme konštrukciou hašovacej funkcie a prezentujeme samotný útok.

1. Úvod

Hašovacie funkcie sú v informatike veľmi dobre známe a v modernej kryptografii zohrávajú dôležitú rolu. Ide o funkcie, ktoré transformujú vstupný reťazec ľubovoľnej dĺžky na výstupný reťazec fixnej dĺžky, ktorému hovoríme digitálny odtlačok.

Definícia 1 *[5] Nech $\{0,1\}^*$ je konečná množina slov nad abecedou $\{0,1\}$. Jednocestá hašovacia funkcia (angl. "One-Way Hash Function OWHF") je funkcia $h : \{0,1\}^* \to \{0,1\}^m$, ktorá spĺňa nasledujúce podmienky:*
1. Argument x môže byť ľubovoľnej dĺžky a výsledný odtlačok $h(x)$ má fixnú dĺžku m bitov.
Dnes je $m \geq 160, \ldots, 256$.
2. Hašovacia funkcia je jednocestá, t.j. pre daný digitálny odtlačok y je "ťažké" (v reálnom čase výpočtovo nerealizovateľné) nájsť takú správu x, že $h(x) = y$ (nájdenie vzoru; angl. "preimage resistance") a k danej správe x je "ťažké" nájsť takú správu x', že $x' \neq x$ a $h(x') = h(x)$ (nájdenie druhého vzoru; angl. "second preimage resistance").

Podľa počtu vstupov, delíme hašovacie funkcie na nekľúčové a kľúčové. Hašovacie funkcie, ktorých vstupom je správa, ktorej odtlačok chceme získať, nazývame nekľúčové alebo MDCs (Manipulation Detection Codes). Ak je vstupom správa a tajný kľúč, ide o kľúčové hašovacie funkcie, ktorým taktiež hovoríme MACs (Message Authentication Codes).
Bezpečná hašovacia funkcia vytvára jedinečný vzťah medzi vstupom a digitálnym odtlačkom. Je však zrejmé, že z viacerých rôznych vstupov môžeme dostať ten istý digitálny odtlačok, nakoľko ide o zobrazenie z väčšej množiny do menšej. Práve preto musí byť zaručené, aby boli tieto vstupy ťažko identifikovateľné, teda aby bolo zložité nájsť kolíziu.

[1] Vedúci práce

Definícia 2 *[5] Hašovacia funkcia odolná voči kolíziám (angl. "Collision Resistant Hash Function CRHF") je taká hašovacia funkcia $h : \{0,1\}^* \to \{0,1\}^m$, pre ktorú je "ťažké" nájsť také dve rôzne správy x, x', že $h(x) = h(x')$ (angl. "collision resistance").*

V kryptografii sa v posledných rokoch objavilo niekoľko kryptosystémov založených na kvázigrupách, aj keď takéto riešenie nie je príliš bežné. Ako je však ukázané v [4], práve použitie kvázigrúp v dizajne S-boxov, môže predstavovať nové riešenia v návrhu blokových šifier.
Kvázigrupy môžeme použiť aj pri návrhu hašovacích funkcií, ako vidíme v [2], [3].

Definícia 3 *[1] Štruktúra $(Q,*)$, $Q = q_1, q_2, \ldots, q_n$, $||Q|| = n$ sa nazýva konečná kvázigrupa rádu n ak platí, že pre ktorékoľvek dva dané prvky $a, b \in Q$, rovnosť $a * x = b$ a $y * a = b$ má práve jedno riešenie. Potom Caleyho tabuľka konečnej kvázigrupy rádu n je latinský štvorec, t.j. pole $n \times n$ s vlastnosťou, že každý riadok a každý stĺpec obsahuje permutáciu prvkov z množiny Q.*

Definícia 4 *[1] Nech $(G,\cdot)$ a $(H,*)$ sú dve konečné kvázigrupy. Usporiadaná trojica (θ, φ, ψ) bijektívnych zobrazení θ, φ, ψ množiny G na množinu H sa nazýva izotópia $(G,\cdot)$ na $(H,*)$, ak $\theta(x) * \varphi(y) = \psi(x \cdot y)$, pre všetky $x, y \in G$. Kvázigrupy $(G,\cdot)$ a $(H,*)$ potom nazývame izotopné.*

2. Konštrukcia

Konštrukcia [2], [3] Nech $(Q, \odot)$ je konečná kvázigrupa. Nech správa, ktorej odtlačok chceme získať, je postupnosť prvkov $(m_1, m_2, \ldots, m_k)$ z kvázigrupy Q. Nech Q^* je množina všetkých konečných slov nad Q. Hašovacia funkcia $H_a : Q \times Q^* \to Q$, $a \in Q$ je daná vzťahom:
$H_a(m_1, m_2, \ldots, m_k) = ((\ldots(a \odot m_1) \odot m_2) \odot \ldots) \odot m_{k-1}) \odot m_k$, kde $m_i \in Q$, $1 \leq i \leq k$.

Príklad 1 *Nech $Q = \{0,1,2,3\}$ a nech operácia $\odot$ na Q je definovaná tabuľkou 1. Nech $a = 1$ a nech správa, ktorej odtlačok chceme vypočítať, je $(1,0,3,1)$. Potom je odtlačok vypočítaný nasledovne:*

$$H_1(1,0,3,1) = (((1 \odot 1) \odot 0) \odot 3) \odot 1 = 2.$$

Tabuľka 1: Caleyho tabuľka operácie $\odot$ definovanej na Q.

$\odot$	0	1	2	3
0	0	2	1	3
1	2	3	0	1
2	1	0	3	2
3	3	1	2	0

Takéto hašovanie nám so sebou prináša obrovské požiadavky na pamäť. K získaniu odtlačku správy, by sme museli uložiť tabuľku o veľkosti n^2 prvkov, pričom za dnes bezpečnú dĺžku odtlačku sa považuje 160 - 256 bitov. Túto pamäťovú náročnosť rieši špeciálna kvázigrupa, kvázigrupa modulárneho odčítania. Operácia $\otimes$ definovaná na Q je daná predpisom

$$a \otimes b = a + (n - b) \bmod n, \qquad n = ||Q||.$$

Tabuľka 2: Tabuľka kvázigrupy modulárneho odčítania pre $n = 4$.

$\otimes$	0	1	2	3
0	0	3	2	1
1	1	0	3	2
2	2	1	0	3
3	3	2	1	0

Tým, že použijeme jednoducho vypočítateľnú operáciu $\otimes$, môžeme používať kvázigrupy s veľkým počtom prvkov. V [6] sa podarilo ukázať, že hašovacia funkcia založená na kvázigrupe modulárneho odčítania nie je spoľahlivým riešením. Kvôli zvýšeniu bezpečnosti sa preto do návrhu hašovacej funkcie zavádzajú izotopné kvázigrupy.

Definícia 5 *Nech $(Q,\otimes)$, $||Q|| = n$, je kvázigrupa modulárneho odčítania. Nech θ, φ a ψ^{-1} sú známe permutácie. Nech $(Q, \cdot)$ je izotopná s kvázigrupou $(Q, \otimes)$. Kvázigrupová operácia v $(Q, \cdot)$ môže byť zapísaná ako*

$$a \cdot b = \psi^{-1}(\theta(a) + (n - \varphi(b)) \bmod n), \qquad n = ||Q||.$$

Generovanie permutácií sa realizuje tak, že sa postupnosť n prvkov rozdelí na niekoľko častí a tie sa rotujú v rôznych smeroch. V [2] a [3] sú prezentované metódy $P1()$, $P2()$, $P3()$, ktoré generujú požadované permutácie θ, φ a ψ^{-1}.
Metóda $P1()$ realizuje výpočet permutácie θ, čo znamená $\theta(x) = P1(x)$. Podobným spôsobom sú realizované aj ďalšie dve permutácie $P2()$ a $P3()$. Metódy sú uvedené v jazyku C++.

Metóda na výpočet permutácie θ

```
ZZ Quasigroup::P1(ZZ x)
{
    ZZ Dim2 = m_Dim / 2;
    if(x < Dim2*2)
    {
        if (x & 1)
        {
            x = 2*((x/2 + 1) % Dim2) + 1;
        }
        else
        {
            x = 2*((x/2 + Dim2 - 1) % Dim2);
        }
    }
    return x;
}
```

Metóda na výpočet permutácie φ

```
ZZ Quasigroup::P2(ZZ x)
{
    ZZ Dim3 = m_Dim / 3;
    bool Shift = m_Dim % 2 >= 1;
    if (x < Dim3*3)
    {
        switch (x % 3)
        {
        case 0:
            x = 3 * ((x/3 + Dim3/3) % Dim3);
            break;
        case 1:
            x = (3 * (Dim3 - x/3)) + 1;
            break;
        case 2:
            x = (3 * ((x/3 + Dim3 - 1)% Dim3))+ 2;
            break;
        }
    }
    else
    {
        if (x % 3 == 2)
        {
            if (x == (Dim3 * 3 + 1))
            {
                x = (Dim3 * 3) + 2;
            }
            else
            {
                x = (Dim3 * 3) + 1;
            }
        }
    }
    if (Shift)
    {
        x = (x + Dim3) % Dim3;
    }
    return x;
}
```

Metóda na výpočet permutácie ψ^{-1}

```
ZZ QuasiGroup::P3(ZZ x)
{
    ZZ Dim2 = m_Dim / 2;
    x = (x + Dim2 - 1) % m_Dim;
    return x;
}
```

2.1 Príklad hašovania

Príklad 2 *Nech $(Q,\otimes)$, $||Q|| = 4$, je kvázigrupa modulárneho odčítania s Caleyho tabuľkou danou v tabuľke 1. Nech θ = [1, 3, 2, 0], φ = [3, 0, 1, 2] a ψ^{-1} = [3, 2, 0, 1]. Caleyho tabuľka kvázigrupy $(Q, \cdot)$ izotopnej s kvázigrupou $(Q, \otimes)$ je zobrazená v tabuľke 3.*

Tabuľka 3: Caleyho tabuľka kvázigrupy $(Q, \cdot)$.

$\cdot$	0	1	2	3
0	0	2	3	1
1	3	1	0	2
2	1	0	2	3
3	2	3	1	0

Nech $a = 1$ a nech správa, ktorej odtlačok chceme získať je $(1, 0, 3, 1)$. Potom je odtlačok vypočítaný nasledovne:

$$H_1(1,0,3,1) = (((1 \cdot 1) \cdot 0) \cdot 3) \cdot 1 = \mathbf{2}.$$

$$1 \cdot 1 = \psi^{-1}(\theta(1) + (4 - \varphi(1)) \bmod 4) =$$
$$= \psi^{-1}(3 + (4 - 0) \bmod 4) = \psi^{-1}(3) = \mathbf{1}$$

$$\mathbf{1} \cdot 0 = \psi^{-1}(\theta(1) + (4 - \varphi(0)) \bmod 4) =$$
$$= \psi^{-1}(3 + (4 - 3) \bmod 4) = \psi^{-1}(0) = \mathbf{3}$$

$$\mathbf{3} \cdot 3 = \psi^{-1}(\theta(3) + (4 - \varphi(3)) \bmod 4) =$$
$$= \psi^{-1}(0 + (4 - 2) \bmod 4) = \psi^{-1}(2) = \mathbf{0}$$

$$\mathbf{0} \cdot 1 = \psi^{-1}(\theta(0) + (4 - \varphi(1)) \bmod 4) =$$
$$= \psi^{-1}(1 + (4 - 0) \bmod 4) = \psi^{-1}(1) = \mathbf{2}$$

3. Útok na hašovaciu funkciu

Vlastnosť, ktorú musí nevyhnutne spĺňať každá hašovacia funkcia, je odolnosť voči kolízii. Náš útok preto zameriame na overenie tejto vlastnosti. Pokúsime sa nájsť dve správy s identickým odtlačkom.

Nech $a \in Q$, a je známym parametrom hašovacej funkcie a majme správu $(m_1, m_2, ...m_k)$, $m \in Q$, $1 \leq i \leq k$. Označme digitálny odtlačok ako $H_a(m_1, m_2, ...m_k) = (...((a \cdot m_1) \cdot m_2) \cdot ...) \cdot m_k = d$ [6].

Falošnú správu môžeme vytvoriť pridaním predpony alebo prípony k pôvodnej správe alebo môžeme vytvoriť úplne novú správu, nezávislú na pôvodnej.

3.1 Pridanie predpony / prípony

Pridaním predpony môže falošná správa vyzerať ako $(p_1, p_2, ..., p_l, m_1, m_2, ..., m_k), p_i \in Q, 1 \leq i \leq l$ [6]. Na základe toho musí platiť, že $(...((a \cdot p_1) \cdot p_2) \cdot ...) \cdot p_l = a$. Falošnú správu môžeme taktiež vytvoriť pridaním prípony a môže vyzerať ako $(m_1, m_2, ..., m_k, s_1, s_2, ..., s_t), s_i \in Q, 1 \leq i \leq t$ [6]. Na základe toho musí platiť, že $(...((d \cdot s_1) \cdot s_2) \cdot ...) \cdot s_t = d$. Môžeme si všimnúť, že všetky prvky predpony, resp. prípony môžeme zvoliť ľubovoľne, s výnimkou jedného (na ľubovoľnom mieste), ktorý je potrebné určiť. Nech $a' = (...((a \cdot p_1) \cdot p_2) \cdot ...)$ a nech $d' = (...((d \cdot s_1) \cdot s_2) \cdot ...)$. Potom potrebujeme nájsť také p_l a s_t, aby platilo $a' \cdot p_l = a$, $d' \cdot s_t = d$.

3.2 Vytvorenie úplne novej správy

Vytvorme novú správu $(x_1, x_2, ..., x_v), x_i \in Q, 1 \leq i \leq v$. Prvky $(x_1, x_2, ..., x_{v-1})$ môžu byť zvolené ľubovoľne. Nech $d' = (...((a \cdot x_1) \cdot x_2) \cdot ...) \cdot x_{v-1}$. Našou úlohou je teda nájsť také x_v , aby $d' \cdot x_v = d$,

$$d = (\psi^{-1}(\theta(d') + (n - \varphi(x_v)) \bmod n),$$

$$\psi(d) = \theta(d') + (n - \varphi(x_v)) \bmod n,$$

$$\varphi(x_v) = \theta(d') + (n - \psi(d)) \bmod n,$$

$$\boldsymbol{x_v = \varphi^{-1}(\theta(d') + (n - \psi(d)) \bmod n)}.$$

Rovnako vieme chýbajúci prvok odvodiť aj pri pridaní predpony, resp. prípony k pôvodnej správe z podkapitoly 3.1.

$$\boldsymbol{p_l = \varphi^{-1}(\theta(a') + (n - \psi(a)) \bmod n)}.$$
$$\boldsymbol{s_t = \varphi^{-1}(\theta(d') + (n - \psi(d)) \bmod n)}.$$

Ako vidíme, zložitosť nájdenia správneho prvku spočíva v nájdení inverzných permutácií φ^{-1} a ψ, či už vytvoríme úplne novú správu alebo doplníme pôvodnú správu o predponu alebo príponu. Permutáciu θ ivertovať nepotrebujeme. Pre obidva tieto typy vytvorenia novej správy je potom zložitosť útoku rovnaká. Tiež vidíme, že ani známy parameter a takto zvoleného návrhu hašovacej funkcie, nijako neovplyvní zložitosť hľadania kolízie, či už ho poznáme alebo nie.

3.3 Hľadanie inverzných permutácií

Je známe, že invertovanie permutácií je vo všeobecnosti náročný problém. Keď sa však podrobnejšie pozrieme na metódy generovania permutácií z kapitoly 2, prichádzame na zaujímavý fakt. Metóda na generovanie permutácie φ je navrhnutá tak, že permutáciu nikdy nevygeneruje. Pre ukážku uvádzame niekoľko príkladov v tabuľke 4.

Tabuľka 4: Tabuľka generovania prvkov φ.

$\|\|Q\|\|$	Generované prvky φ
4	[0, 4, 2, 3]
7	[2, 2, 0, 5, 6, 4, 1]
8	[0, 7, 5, 3, 4, 2, 6, 7]
14	[3, 13, 11, 6, 10, 2, 9, 7, 5, 0, 4, 8, 12, 13]

Z hľadiska definície 4, to predstavuje podstatný problém, keďže podľa nej majú byť θ, φ a ψ^{-1} jednoznačne permutácie. Nakoľko ale jedna z nich nie je, neplatia ďalšie vlastnosti, ktoré sú základom návrhu tejto hašovacej funkcie.

Príklad 3 *Nech $||Q|| = 4$ a nech θ, φ a ψ^{-1} sú vygenerované metódami uvedenými v kapitole 2. Potom je θ = [2, 3, 0, 1], φ = [0, 4, 2, 3] a ψ^{-1} = [1, 2, 3, 0]. Tabuľka operácie $\cdot$ na množine Q je zobrazená v tabuľke 5.*

Tabuľka 5: Tabuľka operácie $\cdot$ na množine Q.

$\cdot$	0	1	2	3
0	3	3	1	0
1	0	0	2	1
2	1	1	3	2
3	2	2	0	3

Podľa definície 3 musí pre konečnú kvázigrupu platiť, že pre ktorékoľvek dva prvky $a, b \in Q$, musí mať rovnosť $a \cdot x = b$ a $y \cdot a = b$ práve jedno riešenie. Zvoľme teda $a = 0$ a $b = 3$. Po dosadení dostávame

$$0 \cdot x = 3,$$

$$y \cdot 0 = 3.$$

Z tabuľky 5 je potom

$$x = 0 \text{ alebo } x = 1 \text{ a } y = 0.$$

Vidíme, že x má dve možné riešenia tejto rovnosti, čím nie je splnená podmienka v definícii 3. Taktiež neplatí, že príslušná tabuľka je latinský štvorec.

Dôsledok 1 *$(Q, \cdot)$ nie je konečná kvázigrupa a Caleyho tabuľka ku $(Q, \cdot)$ nie je latinský štvorec.*

Čo sa týka teoretického hľadiska, vidíme, že celá konštrukcia návrhu, je na základe takéhoto generovania permutácie φ úplne narušená. Hašovať sa ale aj napriek tomuto faktu dá, čo si ukážeme na nasledujúcom príklade.

Príklad 4 *Nech $||Q|| = 4$ a nech $\theta = [2, 3, 0, 1]$, $\varphi = [0, 4, 2, 3]$ a $\psi^{-1} = [1, 2, 3, 0]$ sú vygenerované metódami uvedenými v kapitole 2. Tabuľka operácie $\cdot$ na množine Q je zobrazená v tabuľke 5. Nech správa, ktorej odtlačok chceme vypočítať je (2, 1, 1, 3) a nech $a = 0$. Odtlačok správy vypočítame pomocou tabuľky 5 nasledovne:*

$$H_0(2, 1, 1, 3) = (((0 \cdot 2) \cdot 1) \cdot 1) \cdot 3 = \boldsymbol{3}.$$

Keďže vieme hašovať, môžeme sa aj v tomto prípade pokúsiť vytvoriť falošnú správu s rovnakým odtlačkom. Už v predchádzajúcich príkladoch sme prišli na to, že zložitosť útoku sa odvíja od zložitosti invertovania permutácie ψ a od zložitosti invertovania φ^{-1}, ktorá však, už ako vieme, nie je permutáciou.
Vytvorme teda novú správu, ktorej odtlačok sa bude zhodovať s vyššie vypočítaným $d = 3$. Nech nová správa je $(0, 2, 3, x_v)$. Označme $d' = ((0 \cdot 0) \cdot 2) \cdot 3 = 0$. Platí, že $d' \cdot x_v = d$, teda $0 \cdot x_v = 3$.
Vieme, že x_v môžeme vypočítať nasledovne:

$$x_v = \varphi^{-1}(\theta(d') + (n - \psi(d)) \bmod n).$$

Po dosadení dostávame

$$x_v = \varphi^{-1}(\theta(0) + (4 - \psi(3)) \bmod 4).$$

K určeniu x_v potrebujeme poznať φ^{-1} a ψ. Pozrime sa preto na možnosti invertovania funkcií, ktoré sme uviedli v časti 2.

Metóda na invertovanie permutácie ψ^{-1}
Už na prvý pohľad vidíme, že permutácia ψ^{-1} sa negeneruje zložito. Veľmi jednoducho vieme nájsť inverzný algoritmus k pôvodnému. Metóda je uvedená v jazyku C++.

```
ZZ Quasigroup::invP3(ZZ x)
{
    ZZ Dim2 = m_Dim / 2;
    x = (x + 1 - Dim2) % m_Dim;
    return x;
}
```

Metóda na invertovanie permutácie φ
Algoritmus na generovanie prvkov φ už vyzerá o čosi zložitejšie a problém, ktorý tu vzniká, je hlavne ten, že nejde o permutáciu. Problém sa redukuje na invertovanie postupností s párnym počtom prvkov, teda $||Q|| = 2k$, $k \in N$. Dôvodom je to, že počet prvkov postupnosti (permutácie, vo všeobecnom prípade) vypočítavame z dĺžky odtlačku, a to $||Q|| = 2^d$, kde d je dĺžka odtlačku. Nikdy sa teda nestane, aby permutácia mala nepárny počet prvkov.
Z tabuľky 4 vidíme, že jednoznačne určiť pôvodný prvok k inverznému, môže predstavovať problém, ak by sa prvky opakovali kdekoľvek v danej postupnosti a opakovalo by sa ich rôzne veľa. Algoritmus bol však navrhnutý tak, že nám vygeneruje 3 typy postupností, čo vidíme v nasledujúcej tabuľke 6.

Tabuľka 6: Tabuľka typov postupností φ.

$\|\|Q\|\|$	Špecifický znak
4, 10, 16, 22, ...	1 sa zobrazuje na n
6, 12, 18, 24, ...	1 sa zobrazuje na $n + 1$
8, 14, 20, 26, ...	1 a $n - 1$ sa zobrazuje na $n - 1$

Príklady jednotlivých typov môžeme nájsť v tabuľke 4. Ostatné prvky postupnosti φ, okrem vyššie spomenutých prvkov 1 a $n - 1$, sa generujú s jednoznačným určením inverzného prvku. Pri prvom a druhom type postupnosti ho však vieme určiť jednoznačne tiež, je ním 1. Pri treťom type vznikajú dve možnosti chýbajúcich prvkov, no otestovanie dvoch možností, nepredstavuje žiadny problém. Metóda je uvedená v jazyku C++.

```
ZZ* Quasigroup::invP2(ZZ x)
{
    ZZ * arr = new ZZ[2];
    arr[0] = x;
    arr[1] = to_ZZ(-1);
    ZZ Dim3 = m_Dim / 3;
    if(m_Dim % 2 == 0)
    {
      if(x < Dim3 * 3)
      {
        switch(x % 3)
        {
```

```
          case 0:
            arr[0]=3*((x/3 - Dim3/3)% Dim3);
            break;
          case 1:
            arr[0]=3*(Dim3 - (x-1)/3) + 1;
            break;
          case 2:
            arr[0]=3*(((x-2)/3 + 1 - Dim3)% Dim3)+ 2;
            break;
        }
      }
      else
      {
        if(x == m_Dim || x == m_Dim + 1)
        {
          arr[0] = 1;
          return arr;
        }
      }
      if(x == (Dim3 * 3) + 1)
      {
        arr[0] = 1;
        arr[1] = (Dim3 * 3) + 1;
      }
    }
    return arr;
}
```

Metóda na invertovanie permutácie θ
Pre úplnosť uvádzame aj inverznú funkciu k permutácii θ, o ktorej sme si však povedali, že ju k vytvoreniu falošnej správy invertovať nepotrebujeme. Metóda je uvedená v jazyku C++.

```
ZZ Quasigroup::invP1(ZZ x)
{
    if (m_Dim % 2 == 0)
    {
        if(x % 2 == 0)
        {
            x = 2 * ((x / 2 + 1 - Dim2) % Dim2);
        }
        else
        {
            x = 2 * (((x - 1) / 2 - 1) % Dim2) + 1;
        }
    }
    else
    {
        if(x == m_Dim)
        {
            x = m_Dim;
        }
        else
        {
            if(x % 2 == 0)
            {
                x = 2 * ((x / 2 + 1 - Dim2) % Dim2);
            }
            else
            {
                x = 2 * (((x - 1) / 2 - 1) % Dim2) + 1;
            }
        }
    }
    return x;
}
```

Ak už poznáme algoritmy na generovanie inverzných prvkov, môžeme dokončiť vytvorenie falošnej správy. Potrebujeme vypočítať

$$x_v = \varphi^{-1}(\theta(0) + (4 - \psi(3)) \bmod 4).$$

Vyššie uvedenými metódami vypočítame inverzné prvky a po dosadení dostávame

$$x_v = \varphi^{-1}(2 + (4 - 2) \bmod 4),$$

$$x_v = \varphi^{-1}(0)$$

a teda

$$x_v = 0.$$

Opäť sa nám podarilo nájsť správu, ktorej odtlačok je zhodný s odtlačkom pôvodnej správy.

$$H_0(2, 1, 1, 3) = (((0 \cdot 2) \cdot 1) \cdot 1) \cdot 3 = \mathbf{3},$$

$$H_0(0, 2, 3, 0) = (((0 \cdot 0) \cdot 2) \cdot 3) \cdot 0 = \mathbf{3}.$$

Dôsledok 2 *Hašovacia funkcia H_a nie je odolná voči nájdeniu kolízie, voči nájdeniu druhého vzoru ani voči nájdeniu prvého vzoru.*

4. Záver

V práci sme predstavili návrh hašovacej funkcie založenej na kvázigrupách, podľa [2], [3]. Ide o použitie kvázigrupy izotopnej s kvázigrupou modulárneho odčítania. Už pri študovaní samotnej konštrukcie, sme prišli na podstatný problém v generovaní permutácií, nakoľko jedna z uvedených metód permutáciu negeneruje (časť 3.3). Z teoretického hľadiska to problém je, ale hašovať sa aj napriek tomuto faktu dá, čo sme ukázali na príklade 4.
Útok sme založili na nájdení kolízie a zistili sme, že jeho zložitosť, závisí od zložitosti nájdenia permutácie ψ inverznej k ψ^{-1} a od zložitosti nájdenia postupnosti φ^{-1} inverznej k φ.
Z podkapitoly 3.1 a 3.2 môžeme vidieť, že útok je rovnako zložitý, ak falošnú správu vytvoríme modifikovaním pôvodnej správy, teda pridáme predponu, príponu, resp. obidva, alebo vytvoríme úplne novú správu nezávislú na pôvodnej. Taktiež vidíme, že známy parameter a nijako nevplýva na bezpečnosť hašovacej funkcie a zložitosť útoku je rovnaká, či ho poznáme alebo nie.
Nakoľko sa daná permutácia a postupnosť negenerujú príliš zložito, podarilo sa nájsť inverzné metódy na ich generovanie (časť 3.3). Následne sa podarilo vytvoriť falošnú správu, ktorej odtlačok bol zhodný s odtlačkom pôvodnej správy.

5. Literatúra

[1] DÉNES, J., KEEDWELL, A. D., "Latin Squares and their Applications", Academic Press, New York, 1974

[2] DVORSKÝ, J., OCHODKOVÁ, E., SNÁŠEL, V., "Hashovací funkce založená na kvázigrupách", Mikulášska kryptobesídka, Praha, 2001, pp. 27-36

[3] DVORSKÝ, J., OCHODKOVÁ, E., SNÁŠEL, V., "Hash functions based on large quasigroups", Proc. of Velikonoční kryptologie, Brno, 2002, pp. 1-8

[4] GROŠEK, O., SATKO, L., NEMOGA, K., Ïdeal difference tables from an algebraic point of view", Cryptology and Information Security, Proceedings of VI RECSI, Teneriffe, Spain, 2000, pp. 453-454

[5] PREENEL, B., "The state of hash functions", Cryptology and Information Security, Proceedings of VI RECSI, Teneriffe, Spain, 2000, pp. 3-27

[6] VOJVODA, M., "Cryptoanalysis Of One Hash Function Based On Quasigroup", Tatra Mt. Math.Publ. 29, 2004, pp. 173-181