Криптографија со примена на квазигрупи

Стефан Андонов

Факултет за информатички науки и компјутерско инженерство - Скопје

12 декември 2017

Содржина

Квазигрупи

Definition (Квазигрупа)

N-арен групоид (Q,A) со n-арна операција A, така што во равенството $A(x_1, x_2, ..., x_n) = x_{n+1}$ доколку знаеме n елементи од $x_1, x_2, ..., x_{n+1}$, тоа го специфицира и преостанатиот еден елемент, се наракува квазигрупа.

Од оваа дефиниција произлегува дека од една n-квазигрупа (Q,A), произлегуваат $((n+1)! - 1)$ n-квазигрупи, кои што ќе ги нарекуваме парастрофи на дадената квазигрупа.
Конкретно, доколку станува збор за бинарна квазигрупа, од неа произлегуваат 5 бинарни квазигрупи кои се означуваат како ${}^{(13)}Q, {}^{(23)}Q, {}^{(12)}Q, {}^{(123)}Q, {}^{(132)}Q$.По правило операцијата A се означува со $*$, ${}^{(13)}A$ се означува со $/$, и ${}^{(23)}A$ со $\backslash$.

Бинарна квазигрупа

Definition (Бинарна квазигрупа)

Бинарен групоид (Q, A), се нарекува бинарна квазигрупа доколку во множеството Q, постојат операции ${}^{(23)}A$ $(\backslash)$, ${}^{(13)}A$ $(/)$, така што во алгебрата $(Q, {}^{(13)}A, {}^{(23)}A)$ се исполнети следните правила:

$$(x/y) * y = x \tag{1}$$

$$(x * y)/y = x \tag{2}$$

$$x * (x\backslash y) = y \tag{3}$$

$$x\backslash(x * y) = y \tag{4}$$

Латински квадрат

Definition

Латински квадрат е комбинаторна структура која е претставена во форма на матрица со димензии nxn и содржи точно n^2 елементи, во која секој елемент се јавува точно по еднаш во секоја редица и во секоја колона на матрицата.

	1	2	3	4
1	1	2	3	4
2	2	3	4	1
3	3	4	1	2
4	4	1	2	3

Табела: Латински квадрат со димензии 4x4

Доколку сакаме да работиме со три димензии добиваме латинска коцка итн.

Бинарна квазигрупа како латински квадрат

Доколку го земеме множеството $Q = \{1, 2, 3, 4\}$, и дефинираме операција $*$, тогаш бинарната квазигрупа $(Q, *)$ може да се престави слично како латинскиот квадрат.

$*$	1	2	3	4
1	1	2	3	4
2	2	3	4	1
3	3	4	1	2
4	4	1	2	3

Табела: Латински квадрат со димензии 4x4

Во алгоритмите за енкрипција, оваа табела (латински квадрат), има огромна улога во енкрипцијата на податоците. Како што веќе претходно дефиниравме, од една бинарна група произлегуваат пет парастрофи, па можеме да избереме еден од тие парастрофи, ($/$ или $\backslash$), и истиот да го користиме подоцна за декрипција.

n-Квазигрупа

Definition (n-квазигрупа)

n-арен групоид (Q,A) се нарекува n-арна квазигрупа доколку постојат операции во рамки на множеството Q: ${}^{(1,n+1)}A, {}^{(2,n+1)}A, ..., {}^{(n,n+1)}A$ така што во алгебрата $(Q, {}^{(1,n+1)}A, ..., {}^{(n,n+1)}A)$ следните идентитети се исполнети $\forall i \in [1,n]$:

$$A(x_1^{i-1}, {}^{(i,n+1)}A(x_1^n), x_{i+1}^n) = x_i, \tag{5}$$

$${}^{(i,n+1)}A(x_1^{i-1}, A(x_1^n), x_{i+1}^n) = x_i. \tag{6}$$

Шема базирана на бинарна квазигрупа

Нека Q е конечно множество со N елементи. $Q = \{a_1, a_2, a_3, ..., a_n\}$. Дефинираме бинарна квазигрупа $(Q, *)$. Дополнително ќе дефинираме и еден елемент наречен лидер (носач) $l \in Q$. Нека ни е дадена порака $m = m_1 m_2 m_3 ... m_n$. Ќе дефинираме функција за енкрипција на пораката со носач l во шифрирана пораката $c = c_1 c_2 c_3 ... c_n$.

$$e_l(m_1 m_2 ... m_n) = (c_1 c_2 c_3 ... c_n) = \begin{cases} c_1 = l * m_1, \\ c_{i+1} = c_i * m_{i+1}, 1 < i \leq n-1 \end{cases}$$

Шема базирана на бинарна квазигрупа

Дополнително, како што веќе погоре дефиниравме, за да овозможиме и декрипција со помош на бинарна квазигрупа, потребно е да дефинираме некој од парастрофите. Тука, на пример ќе го користиме парастрофот $/$, одосно ќе имаме нова квазигрупа $(Q, /)$. Формално декрипцијата ќе ја дефинираме како:

$$d_l(c_1c_2...c_n) = (m_1m_2m_3...m_n) = \begin{cases} m_1 = l/c_1, \\ m_{i+1} = c_i/c_{i+1}, 1 < i \leq n-1 \end{cases}$$

Шема базирана на бинарна квазигрупа

Со цел да се постигнеме посилна безбедност креираме низа од лидери (носачи) $l_1, l_2, ..., l_k, l_i \in Q, \forall i = 1, 2, ..., k$, и потоа енкрипцијата и декрипцијата ги извршуваме преку композиција на функциите $e_{l_i}, \forall i = 1, 2, ..., k$ и $d_{l_i} \forall i = 1, 2, ..., k$.

$$E_k = e_{l_1} \circ e_{l_2} \circ e_{l_3} \circ ... \circ e_{l_k}$$

$$D_k = d_{l_k} \circ d_{l_{k-1}} \circ d_{l_{k-2}} \circ ... \circ d_{l_1}$$

Пример

Нека ни е дадено множество $Q = \{1, 2, 3, 4\}$ и нека е дефинирана бинарната квазигрупа $(Q, *)$, исто како во примерот со латинскиот квадрат.

Да се изврши енкрипција, а подоцна и декрипција на пораката $M = 324124322413$ со низата од носачи $L = 12$.

За декрипција да се дефинира и парастрофот $/$ на квазигрупата $(Q, *)$

Пример

*	3	2	4	1	2	4	3	2	2	4	1	3
1	3	4	3	3	4	3	1	2	3	2	2	4
2	4	3	1	3	2	4	4	1	3	4	1	4

Табела: Енкрипција

Оттука следи дека шифрираната порака е
$C(c_1c_2...c_n) = 431324413414.$

Пример

Се со цел да можеме да ја извршиме декрипција, мора да дефинираме квазигрупа $(Q, /)$, која што ќе биде инверзна на квазигрупата $(Q, *)$ и ќе ја користиме за декрипција.

$/$	1	2	3	4
1	1	2	3	4
2	4	1	2	3
3	4	4	1	2
4	2	3	4	1

Табела: Квазигрупа $(Q, /)$ која што се користи за декрипција

Пример

/	4	3	1	3	2	4	4	1	3	4	1	4
2	3	4	3	3	4	3	1	2	3	2	2	4
1	3	2	4	1	2	4	3	2	2	4	1	3

Табела: Декрипција

Ја добиваме истата порака која што ја шифриравме и со тоа ја покажуваме точноста на криптоалгоритмот.

```
import java.util.Arrays;
import java.util.Collections;

class QuasigroupChiper{
   private static final int [][] quasigroupLookUpTable = {
        {1,2,3,4},
        {2,3,4,1},
        {3,4,1,2},
        {4,1,2,3}
   };

   private static final int [][] parastropheLookUpTable = {
        {1,2,3,4},
        {4,1,2,3},
        {3,4,1,2},
        {2,3,4,1}

   };
```

```
private static void printArray (int [] array){
   Arrays.stream(array).forEach(x -> System.out.print(x + "
       "));
   System.out.println();
}

public String encryption (String message, String leaders) {
   String [] m = message.split("\\s+");
   String [] l = leaders.split("\\s+");
   int [] m1 = new int [m.length];
   for (int i=0;i<m.length;i++)
      m1[i]=Integer.parseInt(m[i]);
   int [] l1 = new int [l.length];
   for (int i=0;i<l.length;i++)
      l1[i]=Integer.parseInt(l[i]);
   int [] c1 = new int [m.length];
```

```
    for (int i=0;i<l.length;i++){
       for (int j=0;j<c1.length;j++){
          if (j==0)
             c1[j]=quasigroupLookUpTable[l1[i]-1][m1[j]-1];
          else
             c1[j]=quasigroupLookUpTable[c1[j-1]-1][m1[j]-1];
       }
       QuasigroupChiper.printArray(c1);
       m1=c1;
       c1 = new int [m1.length];
    }
    c1=m1;
    StringBuilder sb = new StringBuilder();
    Arrays.stream(c1).forEach(x -> sb.append(x+" "));
    return sb.toString();
}

public String decryption (String chipertext, String leaders){
    String [] c = chipertext.split("\\s+");
    String [] l = leaders.split("\\s+");
    int [] c1 = new int [c.length];
```

```
for (int i=0;i<c.length;i++)
   c1[i]=Integer.parseInt(c[i]);
int [] l1 = new int [l.length];
for (int i=0;i<l.length;i++)
   l1[i]=Integer.parseInt(l[i]);
int [] m1 = new int [c.length];
int [] pom = new int [l1.length];
for (int i = 0;i<pom.length;i++){
   pom[i]=l1[pom.length-1-i];
}
l1=pom;

for (int i=0;i<l1.length;i++){
   for (int j=0;j<c1.length;j++){
      if (j==0)
         m1[j]=parastropheLookUpTable[l1[i]-1][c1[j]-1];
      else
         m1[j]=parastropheLookUpTable[c1[j-1]-1][c1[j]-1];
   }
   QuasigroupChiper.printArray(m1);
   c1=m1;
```

```
            m1=new int [c1.length];
        }
        m1=c1;
        StringBuilder sb = new StringBuilder();
        Arrays.stream(m1).forEach(x -> sb.append(x + " "));
        return sb.toString();
    }
}
public class QuasigroupChiperTest {
    public static void main(String[] args) {
        QuasigroupChiper qgc = new QuasigroupChiper();
        //System.out.println(qgc.encryption("3 2 4 1 2 4 3 2 2 4 1
            3", "1 2"));
        //System.out.println(qgc.encryption("3 2 4 1 2 4 3 2 2 4 1
            3", "1 2 3 4 4 3 3 2 1 1 2 3 4 3 2 1 1 2 3 4 1 2 3 4 4
            3 2 1 3 2 4 1 3 4 3 2 1 3 2 3 4 1 2 3 4")); //dava
            ramnomerna raspredelba
        System.out.println(qgc.decryption("4 3 1 3 2 4 4 1 3 4 1 4
            ", "1 2"));
    }
}
```

Безбедност на алгоритмот

- Бројот на бинарни квазигрупи (латински квадрати) од ред 4 изнесува 576, додека пак од ред 11 изнесува 776,966,836,171,770,144,107,444,346,734,230,682,311,065,600,000. Замислете колкав е бројот кога станува збор за множество од 26 или 31 елементи (букви).
- Колку што е поголем бројот на елементи во низата од носачи (лидери), горниот број се множи со толку.
- Дополнително, како што погоре дефиниравме, од една бинарна квазигрупа, може да произлезат пет парастрофи. Добиениот број го множиме уште со 5.
- Добиваме огромен број на можни комбинации, кои со brute force не може во години да бидат извршени.
- Алгоритмот е целосно отпорен на статистички напад (фреквенција на букви), бидејќи е докажано дека колку е поголема низата од носачи, толку повеќе распределбата на карактерите во шифрираната порака е униформно распределена.

Шема базирана на тернарни квазигрупи

Нека A е конечно множество кое е наречено **дефиниционa азбука**.
Просторот на пораката M е множество на сите конечни и непразни секвенци на елементите од A.
Пораката или plaintext ја означуваме како $m = m_1 m_2 ... m_n$, која што припаѓа на M. Со C ја означуваме **шифрираната порака (chipertext)**. За почеток ставаме дека $C = M$.
$\forall a \in M$ нека f_a биде пермутација на сите елементи во A.
Нека $(A, \alpha, \alpha_1, \alpha_2, \alpha_3)$ е тернарна квазигрупа со носач A.
Просторот на клучот го дефинираме како $K = A^4$ x [1,2,3]. **Клуч** ќе претставува $k = a_1 a_2 a_3 a_4 i$. Секој клуч k полнозначно дефинира биекција (енкрипциска функција) $E_k : M \to C$, на следниот начин:

Шема базирана на тернарни квазигрупи

Дефинираме нова тернарна квазигрупа (A, β), така што

$$\beta(x_1, x_2, x_3) = f_4(\alpha(f_1^{-1}(x_1), f_2^{-1}(x_2), f_3^{-1}(x_3))) \tag{7}$$

каде што $f_j = f_{aj} \forall i \in [1, 4]$.
Енкрипциската функција $_k(m_1 m_2 ... m_n) = c_1 c_2 ... c_n$ ја дефинираме на следниот начин:
Ако $i = 1$:

$$c_1 = \beta(m_1, a_1, a_2),$$

$$c_2 = \beta(m_2, a_3, a_4),$$

$$c_j = \beta(m_j, c_{j-2}, c_{j-1}), \forall j > 2$$

Шема базирана на тернарни квазигрупи

Ако $i = 2$:

$$c_1 = \beta(1, m_1, a_2),$$

$$c_2 = \beta(3, m_2, a_4),$$

$$c_j = \beta(c_{j-2}, m_j, c_{j-1}), \forall j > 2$$

Ако $i = 3$:

$$c_1 = \beta(a_1, a_2, m_1),$$

$$c_2 = \beta(a_3, a_4, m_2),$$

$$c_j = \beta(c_{j-2}, c_{j-1}, m_j), \forall j > 2$$

Декрипција на шемата со тернарни квазигрупи

Декрипцијата се извршува со помош на **декрипциската функцијата** D_k која претставува биекција $D_k : C \to M$.

Истата е дефинирана наполно идентично како енкрипциската функција, со тоа што на секое место каде што има $c_j, j = 1, 2, 3, ..,$ се заменува со соодветно $m_j, j = 1, 2, 3, ...,$ и обратното, m_j се заменува со c_j.

Својства и безбедност на шемата со тернарна квазигрупа

Оваа енкрипциска и декрипциска шема се користи во главно за проточни шифрувачи, но може да има и своја примена во блоковски шифрувачи.
Шемата е комплетно робусна на грешки.
Шемата е целосно отпорна на brute force напди. Таков напад е невозможен. Да претпоставиме дека напаѓачот го пресретнал шифрираниот текст. За да изврши напад ќе треба да реши три системи равенки. Тоа можеби и не изгледа толку сложено, но треба да се земе во предвид и фактот дека бројот на тернарни квазигрупи кои што може да се моделираат со носач од $n-$елементи е огромен.
Системот е целосно отпорен на напад со фреквенција на букви. Анализите покажале дека распределбата на букви во шифрираниот текст е **униформна**.

Примена во компјутерски систем и безбедност

Секој карактер од азбуката се преставува како низа од 8 битови (ASCII код). Согласно тоа, има точно $2^8 = 256$ можни комбинации на тие 8 битови. Затоа азбуката $A = \{0, 1, 2, ..., 255\}$. Во овој случај постојат најмалку $256!255!...2!1! > 10^{580000}$ бинарни квазигрупи. Од секоја од овие бинарни квазигрупи $(A, *)$ можеме да изведеме по две тернарни квазигрупи:
$\alpha(x_1, x_2, x_3) = (x_1 * x_2) * x_3$ и
$\beta(x_1, x_2, x_3) = x_1 * (x_2 * x_3)$
Претходно пресметаниот огромен број, сега се дуплира.
Просторот на клучот е со големина $(256^4 - 1) \cdot 3$.

Примена во компјутерски систем и безбедност

Претходниот пример компјутерски е реализиран со користење на следните функции во квазигрупите:

$$\alpha(x_1, x_2, x_3) = x_1 - x_2 + x_3 (mod 256)$$

$$\beta(x_1, x_2, x_3) = x_1 - x_2 + x_3 - a_1 + a_2 - a_3 + a_4 (mod 256)$$

Пермутациите се дефинираат како:

$$f_a(x) = x + a (mod 256)$$

Примена на квазигрупите во криптографија

Претходно прикажани методи и постапки се само еден мал дел од примената на квазигрупите во криптографијата, како и во други области.
Во криптографијата имаат примена во некои од најважните криптографски области како:

- проточни шифрувачи
- блоковски шифрувачи
- хеш функции
- генератори на рандом броеви (PRGA)