Tatra Mt. Math. Publ. **45** (2010), 137–149
DOI: 10.2478/v10127-010-0010-0

TATRA MOUNTAINS
Mathematical Publications

CRYPTANALYSIS OF A HASH FUNCTION BASED ON ISOTOPY OF QUASIGROUPS

Ivana Slaminková — Milan Vojvoda

ABSTRACT. This paper deals with cryptanalysis of one hash function based on isotopy of quasigroups [J. Dvorský, E. Ochodková, V. Snášel: *Hash function based on quasigroups,* in: Proc. of Mikulášska kryptobesídka, Praha, 2001, pp. 27–36. (In Czech)], [J. Dvorský, E. Ochodková, V. Snášel: *Hash functions based on large quasigroups,* in: Proc. of Velikonoční kryptologie, Brno, 2002, pp. 1–8. (In Czech)]. Our work enhances the paper [M. Vojvoda: *Cryptanalysis of one hash function based on quasigroup,* Tatra Mt. Math. Publ. **29** (2004), 173–181], where the simplified studied hash function, based only on the quasigroup of modular subtraction, was successfully cryptanalysed. In this paper we show how to construct collisions, 2nd preimages, and also preimages for the full hash function based on isotopy of quasigroups.

1. Introduction

Hash functions are well known in computer science and play an important role in modern cryptography. They transform an input string of arbitrary length onto the fixed length output string, called also the hash.

Definition 1 ([8])**.** Let $\{0,1\}^*$ be the set of all finite words over the alphabet $\{0,1\}$. A one-way hash function (OWHF) is a function $h\colon \{0,1\}^* \to \{0,1\}^m$ satisfying the following conditions:

(1) The argument x can be of arbitrary length and the result $h(x)$ has a fixed length of m bits. Nowadays $m \geq 160, \ldots, 512$.

(2) The hash function must be one way in the sense that given y in the image of h, it is "hard" to find a message x such that $h(x) = y$ (preimage resistant) and given x in the domain of h it is "hard" to find a message $x' \neq x$ such that $h(x') = h(x)$ (second preimage resistant).

2010 Mathematics Subject Classification: 94A60.
Keywords: hash function, quasigroup, isotopy, cryptanalysis.
This material is based upon the work supported under the grant NIL-I-004 from Iceland, Lichtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism and also under the grant VEGA 1/0244/09.

According to the number of inputs one can subdivide the hash functions into the two classes: keyed, and unkeyed, respectively. Hash functions that use a single input, a message to be hashed, are called unkeyed, or Manipulation Detection Codes (MDCs), sometimes also cryptographic hash functions or just only hash functions. On the other hand, hash functions with a message to be hashed and a key being their inputs are called Message Authentication Codes (MACs).

Definition 2 ([8]). A collision resistant hash function (CRHF) is a OWHF $h\colon \{0,1\}^* \to \{0,1\}^m$, for which it is "hard" to find two distinct messages x, x', such that $h(x) = h(x')$.

The usage of quasigroups (resp. latin squares) in cryptography is not very common. In spite of that various cryptosystems based on quasigroups appeared in past few years, e.g., see [6], where it is shown that the usage of quasigroups in the design of S-boxes might open new ways in the design of block ciphers. Quasigruops were also used in the design of hash functions [2], [3] and the cryptanalysis of these designs is the main focus of this paper. Some recent designs using quasigroups are, e.g., the stream cipher Edon80 [4] and the hash function Edon-R [5].

Definition 3 ([1]). The structure $(Q, *)$, $Q = \{q_1, q_2, \ldots, q_n\}$, $\|Q\| = n$ is called a finite quasigroup of order n if any two elements $a, b \in Q$ are given, the equations $a * x = b$ and $y * a = b$ have exactly one solution. Thus the Caley table of a finite quasigroup of order n is a latin square, i.e., an $n \times n$ array with the property that each row and each column contains a permutation of symbols from Q.

Definition 4 ([1]). Let $(G_1, \cdot)$ and $(G_2, *)$ be two finite quasigroups. An ordered triple (θ, φ, ψ) of one-to-one mappings of the set G_1 onto the set G_2 is called an isotopism of $(G_1, \cdot)$ upon $(G_2, *)$ if $\theta(x) * \varphi(y) = \psi(x \cdot y)$ for all x, y in G_1. Then the quasigroups $(G_1, \cdot)$ and $(G_2, *)$ are said to be isotopic.

2. Description of the studied hash function

Definition 5 ([2], [3]). Let $(Q, \odot)$ be a finite quasigroup. Let $(m_1, m_2, \ldots, m_k)$, $m_i \in Q$, $1 \leq i \leq k$, be the message to be hashed. Further let Q^* be a set of all finite strings over Q. The hash function $H_a\colon Q \times Q^* \to Q$, $a \in Q$ is defined by the following relation:

$$H_a(m_1, m_2, \ldots, m_k) = \big((\ldots(a \odot m_1) \odot m_2) \odot \ldots\big) \odot m_{k-1}) \odot m_k,$$

where $m_i \in Q$, $1 \leq i \leq k$ and a is a given element from Q, that plays the role of an initialization vector.

EXAMPLE. Let $Q = \{0, 1, 2, 3\}$ and the operation $\odot$, defined on Q, be given by Table 1. Let $a = 1$ and the message to be hashed be $(1, 0, 3, 1)$. Then the hash is calculated as follows:

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

TABLE 1. Caleyho table of the operation $\odot$ defined on 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

The usage of a general quasigroup would result in the necessity to store the whole Caley table of the quasigroup operation. If we assume a hash length to be 18 bits, then the quasigroup would contain 2^{18} elements and the storage requirements for the Caley table would exceed 1 TB ($18.2^{18}.2^{18}$). Recall that the nowadays used hash length is about 160–512 bits. To overcome this problem, a special quasigroup, namely the quasigroup of modular subtraction, was proposed in [2], [3] to be used. The operation $\otimes$, defined on Q, is then given as

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

The usage of the "easy to evaluate" expression for the definition of the operation $\otimes$ allows us to use quasigroups with a very large number of elements without the necessity to store the Caley tables. However, the usage of the quasigroup of modular subtraction as the only operation for the hash function is insecure, as was shown in [9]. That is why the authors in [2], [3] suggest to use quasigroups isotopic with the quasigroup of modular subtraction.

DEFINITION 6. Let $(Q,\otimes)$, $\|Q\| = \mathrm{n}$, be the quasigroup of modular subtraction. Let θ, φ a ψ^{-1} be given permutations that define isotopy of $(Q, \cdot)$ with $(Q, \otimes)$. The quasigroup operation in $(Q, \cdot)$ can be written as

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

In the following we present the algorithms $P1()$, $P2()$, and $P3()$ for the mappings θ, φ, and ψ^{-1}, respectively. Note that there was only the algorithm $P1()$ specified in [2], [3], the remaining algorithms $P2()$, and $P3()$ were obtained from [7]. The algorithms are given in C++ language with NTL (Number Theory Library) data types.

Algorithm for the mapping 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;
}
```

Algorithm for the mapping 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;
}
```

Algorithm for the mapping ψ^{-1}

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

EXAMPLE. Let $(Q, \otimes)$, $\|Q\| = 4$ be the quasigroup of modular subtraction (its Caley table is given in Table 2). Further let $\theta = [1, 3, 2, 0]$, $\varphi = [3, 0, 1, 2]$, and $\psi^{-1} = [3, 2, 0, 1]$. The Caley table of the quasigroup $(Q, \cdot)$, that is isotopic with $(Q, \otimes)$, is given in Table 3.

Let $a = 1$ and $(1, 0, 3, 1)$ be the message to be hashed. The hash value is then calculated as follows:

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

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

TABLE 2. Caley table of the quasigroup of modular subtraction, $n = 4$.

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

TABLE 3. The Caley table of the quasigroup $(Q, \cdot)$.

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

3. Attack on the hash function

Second preimage resistance is one of the crucial requirements a cryptographic hash function must meet. In the following, we show how to find a message that has the same hash value as another given message.

Let $a \in Q$ be a given known parameter of the hash function and $(m_1, m_2, \ldots, m_k)$, $m_i \in Q$, $1 \leq i \leq k$ be a given message with the hash value $d = H_a(m_1, m_2, \ldots, m_k) = \big(\ldots((a \cdot m_1) \cdot m_2) \cdot \ldots\big) \cdot m_k$. The false message can be created by adding a prefix or suffix to the given message, or moreover, one can create a totally new message, that is independent on the given one [9].

3.1. Creating a false message by adding a prefix/suffix to the given message

The false message created from the original one by adding a prefix can be written as $(p_1, p_2, \ldots, p_l, m_1, m_2, \ldots, m_k), p_i \in Q, 1 \leq i \leq l$. Hence, it must hold that $\big(\ldots((a \cdot p_1) \cdot p_2) \cdot \ldots\big) \cdot p_l = a$. A false message can be created by adding a suffix to the original message as well. In this case the false message can be written as $(m_1, m_2, \ldots, m_k, s_1, s_2, \ldots, s_t)$, $s_i \in Q$, $1 \leq i \leq t$. Hence, it must hold that $\big(\ldots((d \cdot s_1) \cdot s_2) \cdot \ldots\big) \cdot s_t = d$. Note that only the last element of the last added/changed part of the message has to be chosen in a proper way. All the other elements can be chosen arbitrarily, i.e., they can represent

meaningful data. (In general, the element that should be chosen in proper way could be at any position in the message and not only be the last one. However, the non-associativity of a quasigroup would make difficult to find this element.) Let $a' = \big(\ldots((a \cdot p_1) \cdot p_2) \cdot \ldots\big)$ and $d' = \big(\ldots((d \cdot s_1) \cdot s_2) \cdot \ldots\big)$. Then one has to find such p_l and s_t, so that it would hold $a' \cdot p_l = a$, and $d' \cdot s_t = d$, respectively.

3.2. Creating a new (false) message

Let us create a message $(x_1, x_2, \ldots, x_v)$, $x_i \in Q$, $1 \leq i \leq v$, that has the same hash value d. The elements $(x_1, x_2, \ldots, x_{v-1})$ can be chosen arbitrarily. Let $d' = \big(\ldots((a \cdot x_1) \cdot x_2) \cdot \ldots\big) \cdot x_{v-1}$. The task of the attacker is to find such x_v, that $d' \cdot x_v = d$,

$$\begin{aligned} d &= \psi^{-1}\big(\theta(d') + (n - \varphi(x_v)) \bmod n\big), \\ \psi(d) &= \theta(d') + \big(n - \varphi(x_v)\big) \bmod n, \\ \varphi(x_v) &= \theta(d') + \big(n - \psi(d)\big) \bmod n, \\ x_v &= \varphi^{-1}\big(\theta(d') + (n - \psi(d)) \bmod n\big). \end{aligned}$$

Moreover, we can find the last element of the prefix/suffix added to the original message (see Section 3.1) in a similar way:

$$\begin{aligned} p_l &= \varphi^{-1}\big(\theta(a') + (n - \psi(a)) \bmod n\big), \\ s_t &= \varphi^{-1}\big(\theta(d') + (n - \psi(d)) \bmod n\big). \end{aligned}$$

As we can see, the problem of finding the proper element is in finding the permutations φ^{-1}, and ψ, respectively. The permutation θ need not be inverted. It can be also seen that the complexity of creating a false message by adding a prefix/suffix to a given message or creating a new false message is the same. The parameter a has no influence on the difficulty of creating a false message.

3.3. Inverting the mappings θ, φ, and ψ^{-1}

In general, inverting a permutation could be a difficult problem. After an initial inspection of the φ mapping we observed that it is not a permutation! Some examples confirming this statement are shown in the Table 4. Concerning the Definition 4, it is a significant problem, because the mappings θ, φ, and ψ^{-1} should be permutations.

Example. Let $\|Q\| = 4$ and θ, φ, and ψ^{-1} be generated by the methods P1, P2, and P3, respectively, stated in Section 2. Then $\theta = [2, 3, 0, 1]$, $\varphi = [0, 4, 2, 3]$, and $\psi^{-1} = [1, 2, 3, 0]$. The Caley table of the operation $\cdot$, defined on Q, is given in Table 5.

TABLE 4. Some examples of the values of the mapping φ.

$\|Q\|$	Values of the mapping φ
4	[0, 4, 2, 3]
7	[2, 2, 0, 5, 6, 4, 1]
8	[0, 7, 5, 3, 4, 2, 6, 7]
12	[3, 13, 11, 6, 10, 2, 9, 7, 5, 0, 4, 8]

TABLE 5. The Caley table of the operation $\cdot$ defined on 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

According the Definitions 3 and 4 the equations $a \cdot x = b$ and $y \cdot a = b$, have exactly one solution, when any two elements $a, b \in Q$ are given. Let

$$a = 0 \quad \text{and} \quad b = 3.$$

Substituting these values into the previously mentioned equations we obtain

$$0 \cdot x = 3,$$

$$y \cdot 0 = 3.$$

Then the solutions are $x = 0$, or $x = 1$, and $y = 0$ (see Table 5), that is in contradiction with the Definition 3.

COROLLARY 1. *$(Q, \cdot)$ is not a finite quasigroup and the Caley table of $(Q, \cdot)$ is not a latin square.*

Although the mapping φ is not a permutation, computing a hash value is possible. We demonstrate it on the following example.

EXAMPLE. Let $\|Q\| = 4$ and $\theta = [2, 3, 0, 1]$, $\varphi = [0, 4, 2, 3]$, and $\psi^{-1} = [1, 2, 3, 0]$ be generated by the methods P1, P2, and P3, respectively, stated in Section 2. The Caley table of $(Q, \cdot)$ is given in Table 5. Let $a = 0$ and $(2, 1, 1, 3)$ be the message to be hashed. The hash value can be computed as follows:

$$H_0(2, 1, 1, 3) = \big(((0 \cdot 2) \cdot 1) \cdot 1\big) \cdot 3 = 3.$$

Let us create a new (false) message with the same hash value $d = 3$. Let this message be

$$(0, 2, 3, x_v) \quad \text{and} \quad d' = \big((0 \cdot 0) \cdot 2\big) \cdot 3 = 0.$$

Hence, it must hold $d' \cdot x_v = d$, i.e., $0 \cdot x_v = 3$. x_v can be calculated as follows:

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

However, in this case, one has to invert the mappings φ and ψ^{-1} to obtain x_v. The algorithms for the inverse mappings to θ, φ, and ψ^{-1} are stated below in C++ language with NTL data types.

Algorithm for the mapping ψ

The algorithm for the mapping ψ^{-1} is very simple (see Section 2) and finding the inverse mapping was easy.

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

Algorithm for the mapping φ^{-1}

The main problem concerning the φ mapping is that it is not a permutation. We will deal only with $\|Q\| = 2k$, $k \in \mathbb{N}$. Recall that the number of elements in the quasigroup is calculated from the bit length of the hash value, thus the number of elements in Q is, in fact, a power of 2. Some specific features of the φ mapping are shown in the Table 6.

TABLE 6. Some specific features of the φ mapping.

$\|Q\|$	Specific feature
4, 10, 16, 22, ...	1 is mapped to n
6, 12, 18, 24, ...	1 is mapped to $n + 1$
8, 14, 20, 26, ...	1 and $n - 1$ are mapped to $n - 1$

Some examples of the φ mapping values can be found in Table 4. Except the elements 1 and $n - 1$, it is possible to invert the mapping φ uniquely. The algorithm for calculating φ^{-1} is stated below in the C++ language using the NTL library.

```
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;
}
```

Algorithm for the mapping θ^{-1}

Although it is not necessary to invert the mapping θ in our construction of false messages, we give also the algorithm for calculating θ^{-1} for completeness (again in the C++ language using the NTL library).

```
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;
}
```

We can finish our construction of a false message from the example above. Now,

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

Using the algorithms for calculating φ^{-1} and ψ we obtain

$$x_v = \varphi^{-1}\big(2 + (4 - 2) \bmod 4\big) = \varphi^{-1}(0) = 0.$$

Thus we have found the 2nd preimage to the message (2, 1, 1, 3):

$$H_0(2,1,1,3) = \big(((0 \cdot 2) \cdot 1) \cdot 1\big) \cdot 3 = 3,$$
$$H_0(0,2,3,0) = \big(((0 \cdot 0) \cdot 2) \cdot 3\big) \cdot 0 = 3.$$

Remark 1. Note that the previously described construction of false messages (2nd preimages) can be used to construct collisions and preimages in a similar way as well.

Corollary 2. *The hash function H_a is neither collision resistant, nor 2nd preimage resistant, and preimage resistant. Thus the hash function H_a is totally broken and is insecure for cryptographic use.*

4. Conclusions

We described the cryptanalysed hash function, that was proposed in [2], [3]. The studied hash function uses a quasigroup isotopic to the quasigroup of modular subtraction. This paper enhances the paper [9], where the simplified studied hash function, based only on the quasigruop of modular subtraction, was successfully cryptanalysed. We have found a significant problem concerning the mapping φ that is one of the mappings defining the isotopy. It is not a permutation what contradicts the design ideas. Moreover, we have presented the construction of false messages (collisions, 2nd preimages, and also preimages) for the hash function H_a. The attack is based on inverting the mappings defining the isotopy.

It still remains an open question whether such a construction of a hash function is secure if "hard-to-invert" mappings are used for the isotopy. The structure of the underlying quasigroup of modular subtraction could be useful in the attack.

REFERENCES

[1] DÉNES, J. — KEEDWELL, A. D.: *Latin Squares and their Applications*, Acad. Press, New York, 1974.

[2] DVORSKÝ, J. — OCHODKOVÁ, E. — SNÁŠEL, V.: *Hash function based on quasigroups*, in: Proc. of Mikulášska kryptobesídka, Praha, 2001, pp. 27–36. (In Czech)

[3] DVORSKÝ, J. — OCHODKOVÁ, E. — SNÁŠEL, V.: *Hash functions based on large quasigroups*, in: Proc. of Velikonoční kryptologie, Brno, 2002, pp. 1–8. (In Czech)

[4] GLIGOROSKI, D. — MARKOVSKI, S. — KNAPSKOG, S. J.: *The stream cipher Edon80*, in: New Stream Cipher Designs: The eSTREAM Finalists, Lecture Notes in Comput. Sci., Vol. 4986, Springer-Verlag, New York, 2008, pp. 152–169.

[5] GLIGOROSKI, D. — MARKOVSKI, S. — KOCAREV, L.: *Edon-R: An infinite family of cryptographic hash functions,* Internat. J. of Network Security, **8(3)** (2009), pp. 293–300.
[6] GROŠEK, O. — SATKO, L. — NEMOGA, K.: *Ideal difference tables from an algebraic point of view*, in: Proc. of VI RECSI, Cryptology and Information Security (P. C. Gil, C. H. Goya, eds.) Teneriffe, Spain, 2000, RA-MA, Madrid, 2000, pp. 453–454.
[7] OCHODKOVÁ, E.: e-mail communication.
[8] PRENEEL, B.: *The state of hash functions*, in: Proc. of VI RECSI, Cryptology and Information Security (P. C. Gil, C. H. Goya, eds.) Teneriffe, Spain, 2000, RA-MA, Madrid, 2000, pp. 3–27.
[9] VOJVODA, M.: *Cryptanalysis of one hash function based on quasigroup*, Tatra Mt. Math. Publ. **29** (2004), 173–181.

Received May 21, 2010

Department of Applied Informatics and Information Technology
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Ilkovičova 3
SK–812-19 Bratislava
SLOVAKIA

E-mail: milan.vojvoda@stuba.sk
ivana.slaminkova@gmail.com