University "Alexandru Ioan Cuza" of Iaşi
Faculty of Computer Science

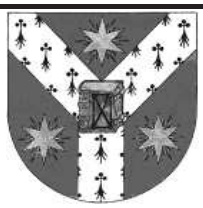
TECHNICAL REPORT

Applications of Parikh Matrices in Coding Theory

Radu-Florian Atanasiu

TR 10-03, May 2010

ISSN 1224-9327



Universitatea "Alexandru Ioan Cuza" Iaşi
Facultatea de Informatică
Str. Berthelot 16, 6600-Iaşi, Romania
Tel. +40-32-201090, email: bibl@infoiasi.ro

Contents

1 Introduction

In this report we will describe a possible application of Parikh matrices and amiability in the field of information security. The results presented were developed in [Atanasiu, A. & Atanasiu, R., 2008].

First we will start with some basic notations and definitions.

Let Σ be a nonempty and finite alphabet and $\mathbb{N}$ the set of nonnegative integers (or natural numbers). Mainly, we will work with an alphabet like $\Sigma = \{a_1, a_2, \ldots, a_s\}$, for which we define an order relation $<$. Without loss of generality, we consider $a_i < a_{i+1}$ for all $1 \leq i \leq s-1$ (if not specified otherwise, we will consider this order relation as implicit on every alphabet we work with from now on). The set of all words over Σ is Σ^*; if λ is the empty word, then the set of nonempty sequences is $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. For $\alpha \in \Sigma^*$, $|\alpha|$ denotes the length of α. Besides, for any finite set A we denote $|A|$ the number of elements contained by A.

The mirror image of a word $\alpha \in \Sigma^*$, denoted $\mathrm{mi}(\alpha)$, is defined as: $\mathrm{mi}(\lambda) = \lambda$, $\mathrm{mi}(x_1 x_2 \ldots x_n) = x_n \ldots x_2 x_1$, where $x_i \in \Sigma$, $1 \leq i \leq n$. A word α is a *palindrome* if and only if $\alpha = \mathrm{mi}(\alpha)$.

The number of occurrences of a letter $a \in \Sigma$ in a word $\alpha \in \Sigma^*$ is denoted by $|\alpha|_a$. If $u, v \in \Sigma^*$, then the word u is a scattered subword of v if $u = \beta_1 \beta_2 \ldots \beta_r$ and $v = \gamma_0 \beta_1 \gamma_1 \ldots \gamma_{r-1} \beta_r \gamma_r$, for some $r \geq 1$ and $\beta_i, \gamma_j \in \Sigma^*$. We denote by $|\alpha|_u$ the number of occurrences of u in α as a scattered subword. For instance, for $\Sigma = \{a_1, a_2\}$, we have $|a_1 a_2 a_1 a_2|_{a_1 a_2} = 3$. For all $1 \leq i \leq j \leq s$ we denote $a_{i,j} = a_i a_{i+1} \ldots a_j$.

Next we will give the definition of the Parikh mapping:

Definition 1.1. *[Mateescu, A., Salomaa, A., Salomaa, K. & Yu, S., 2001] Let $\Sigma_s = \{a_1, a_2, \ldots, a_s\}$ be an ordered alphabet. The Parikh mapping is a mapping*

$$\Psi : \Sigma^* \longrightarrow \mathbb{N}^s$$

defined as:

$$\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_s}) \text{ , for } w \in \Sigma^*.$$

We say that $(|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_s})$ is the Parikh vector of the word w.

It is important to notice that the Parikh mapping Ψ is a morphism from the monoid $(\Sigma^*, \cdot, \lambda)$ to the monoid $\left(\mathbb{N}^s, +, \underbrace{(0, 0, \ldots, 0)}_{s}\right)$.

We are now ready to formalize the definition of the Parikh matrix mapping:

Definition 1.2. *[Mateescu, A.* et al.*, 2001] Let $\Sigma = \{a_1, a_2, \ldots, a_s\}$ be an ordered alphabet and $\mathcal{M}_{s+1}$ be the multiplicative monoid of $(s+1)$ - dimensional upper-triangular matrices with nonnegative integer entries and unit diagonal. The Parikh matrix mapping, denoted Ψ_s, is the morphism*

$$\Psi_s : \Sigma^* \longrightarrow \mathcal{M}_{s+1}$$

defined by as follows:
if $k = 1, \ldots, s$ and $\Psi_s(a_k) = (m_{i,j})_{1 \leq i,j \leq s+1}$, then for each $1 \leq i \leq s+1$, $m_{i,i} = 1$, $m_{k,k+1} = 1$, all other elements of the matrix $\Psi_s(a_k)$ being 0.

Example 1.3. *Let $\Sigma_3 = \{a, b, c\}$ and $w = abacb$. Then $s = 3$ and $\Psi_3(w)$ is a 4×4 upper triangular matrix that can be computed as follows:*

$$\Psi_3(abacb) = \Psi_3(a)\Psi_3(b)\Psi_3(a)\Psi_3(c)\Psi_3(b)$$
$$= \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 2 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

If $|\Sigma| = s$ is fixed, we will denote $\Psi_s(\alpha)$ also by M_α. Also, for simplicity we will denote by Σ_s such an alphabet with an order relation defined as in the previous paragraphs.

A first remark regarding the properties of the Parikh matrix mapping is also a very important one:

Remark 1.4. *The Parikh matrix mapping is not injective. For instance, for $\Sigma_3 = \{a, b, c\}$, the words "*cab*" and "*acb*" have the same image through the Parikh matrix mapping:*

$$\Psi_3(cab) = \Psi_3(acb) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A natural question after this remark is: can one determine a natural criterion (non-numerical) for two words to have the same Parikh matrix mapping? Several conditions were discovered, but which do not cover all the existing cases, except for binary alphabets. In fact, it should be pointed out that concerning Parikh matrices, there are immense differences in the phenomena that arise in the case of binary alphabets, compared to higher-order alphabets.

Going further now, we will give a theorem that follows naturally from the definition:

Theorem 1.5. *[Mateescu, A.* et al.*, 2001] Consider $\Sigma_s = \{a_1, a_2, \ldots, a_s\}$ and $w \in \Sigma^*$. The matrix $M_w = \Psi_s(w) = (m_{i,j})_{1 \leq i,j \leq s+1}$ has the following properties:*

- $m_{i,j} = 0$ *for all* $1 \leq j < i \leq s + 1$,
- $m_{i,i} = 1$ *for all* $1 \leq i \leq s + 1$,
- $m_{i,j+1} = |w|_{a_{i,j}}$ *for all* $1 \leq i \leq j \leq s$.

This theorem practically points out the structure of a Parikh matrix. It says that given a word, we can find its corresponding Parikh matrix by calculating the number of appearances only of those subwords composed of consecutive letters in ascending order (bearing in mind the ordering of the alphabet). For instance, the Parikh matrix computed in Example 1.3 for the word $w = abacb$ is:

$$\Psi_3(w) = \begin{pmatrix} 1 & |w|_a & |w|_{ab} & |w|_{abc} \\ 0 & 1 & |w|_b & |w|_{bc} \\ 0 & 0 & 1 & |w|_c \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 1 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It is obvious that for any words u, v and w such that $w = uv$ we have that $\Psi_s(w) = \Psi_s(u) \cdot \Psi_s(v)$, which is equivalent to writing $M_w = M_u \cdot M_v$.

Now it is easy to make the following remark:

Remark 1.6. *[Mateescu, A.* et al.*, 2001] For any word $w \in \Sigma_s^*$, the Parikh matrix $\Psi_s(w)$ has the Parikh vector as the second diagonal, i.e.*
$(m_{1,2}, m_{2,3}, \ldots, m_{s,s+1}) =$
$(|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_s})$.

One might ask if the properties of the Parikh mapping are imported to the Parikh matrix mapping. The answer is generally "NO", and to illustrate this we will analyze perhaps the most famous and used property of the Parikh mapping:

> "If L is a context-free language, then its image through the Parikh mapping is a semilinear set."

Now, what if we take the context-free language $L = \{a^n b^n \mid n \geq 1\}$ over $\Sigma_2 = \{a, b\}$? We get:

$$\Psi_2(a^n b^n) = \begin{pmatrix} 1 & n & n^2 \\ 0 & 1 & n \\ 0 & 0 & 1 \end{pmatrix}$$

Hence, $\Psi_2(L)$ cannot be a semilinear set, therefore the property does not hold for the Parikh matrix mapping.

Having encoded words by matrices means that we have to draw some advantages from this algebraic theory. The road was opened with a series of properties that link the matrices theory to the mirror of the words. These early results represented the startup of the theory towards several directions, such as the study of injectivity of the Parikh matrix mapping or the relation to languages - to mention only such two possibilities.

We will start by noticing that Parikh matrices are nonsingular, thus any Parikh matrix M_w (over any ordered alphabet Σ_s) has an inverse matrix M_w^{-1}. This comes as no surprise, since the Parikh matrices of order higher than two is a noncommutative group with respect the operation of standard multiplication, and having the unit element the unit matrix I_{s+1}.

Definition 1.7. *[Mateescu, A.* et al.*, 2001] Let w be a word over the alphabet Σ_s and $M_w = (m_{i,j})_{1 \leq i,j \leq s+1}$ its Parikh matrix. The alternate Parikh matrix of w, denoted $\overline{M_w}$, is the matrix $(m'_{i,j})_{1 \leq i,j \leq s+1}$, where $m'_{i,j} = (-1)^{i+j} m_{i,j}$, for all $1 \leq i, j \leq s+1$.*

Before going further, please note that a word w and its mirror $\mathrm{mi}(w)$ always have the same Parikh vector, but this is not true for Parikh matrices. Only palindromes have this property.

Theorem 1.8. *[Mateescu, A.* et al.*, 2001] $[M_w]^{-1} = \overline{M}_{\mathrm{mi}(w)}$, for all words $w \in \Sigma_s^*$.*

Corollary 1.9. *[Mateescu, A.* et al.*, 2001] Let w be a word over Σ_s and $M_w = (m_{i,j})_{1 \leq i,j \leq s+1}$ its Parikh matrix. Assume that $[M_w]^{-1} = (m'_{i,j})_{1 \leq i,j \leq s+1}$. Then $|mi(w)|_{a_{i,j}} = |m'_{i,j+1}|$[1], for all $1 \leq i, j \leq s$.*

If Theorem 1.8 gives a method for computing the inverse of a Parikh matrix, Corollary 1.9 establishes a sharp relationship between the Parikh matrices of words and those of their mirrors. This relation will prove very useful when characterizing certain classes of words having the same Parikh matrix.

Example 1.10. *Let $\Sigma_3 = \{a, b, c\}$ and the word $w = cbbaa$. Obviously,* $\mathrm{mi}(w) = aabbc$. *We have that:*

$$M_{cbbaa} = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_{aabbc} = \begin{pmatrix} 1 & 2 & 4 & 4 \\ 0 & 1 & 2 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

By Theorem 1.8 the inverse matrix of M_{cbbaa} is:

[1] by $|m'_{i,j+1}|$ we understand the absolute value of $m'_{i,j+1}$

$$[M_{cbbaa}]^{-1} = \overline{M}_{aabbc} = \begin{pmatrix} 1 & -2 & 4 & -4 \\ 0 & 1 & -2 & 2 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

There has been another way of describing the Parikh matrix of the mirror of a word, by reversing the alphabet's ordering. Thus, we denote by $<^\circ$ the *dual order* of the order $<$, defined by $a <^\circ b$ iff $b < a$. In the same scope we define the *dual ordered alphabet*, $\Sigma_{s,\circ} = \{a_s < a_{s-1} < \ldots < a_1\}$ for an ordered alphabet $\Sigma_s = \{a_1 < a_2 < \ldots < a_s\}$. The Parikh matrix associated to a word $w \in \Sigma_s^*$ with respect to the dual order on Σ_s is denoted by $M_{s,\circ}(w)$.

Now we introduce the reverse of a triangle matrix: let $M = (m_{i,j})_{1 \leq i,j \leq s}$ be such a matrix. The *reverse* of M, denoted $M^{(rev)}$, is the matrix $M^{(rev)} = (m'_{i,j})_{1 \leq i,j \leq s}$, where $m'_{i,j} = m_{s+1-j,s+1-i}$, for all $1 \leq i < j \leq s$.

One easy way to obtain $M^{(rev)}$ is to reverse in M all diagonals that are parallel to the main diagonal.

Example 1.11. *For* $M = \begin{pmatrix} 1 & 1 & 2 & 6 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $M^{(rev)} = \begin{pmatrix} 1 & 5 & 4 & 6 \\ 0 & 1 & 3 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Theorem 1.12. *[Mateescu, A.* et al.*, 2001]* $[M_w]^{-1} = \overline{M}_{w,\circ}^{(rev)}$*, for all words* $w \in \Sigma_s^*$.

The next corollary shows the connection between words and their mirror:

Corollary 1.13. *[Mateescu, A.* et al.*, 2001]* $M_{\mathrm{mi}(w)} = M_{w,\circ}^{(rev)}$*, for all words* $w \in \Sigma_s^*$.

As one may see, this method of computing the Parikh matrix of a word's mirror is highly similar to the previous one.

This introductive section presented some basic notions and early results, aiming to familiarize the reader with the concepts of *Parikh matrix mapping* and *Parikh matrix*.

2 Message Authentication using Amiability

The idea of identifying binary sequences using as input data the number of 0s and 1s is quite old (in the following, for the sake of consistency, we will consider these sequences as being strings or words). Unfortunately, this information (given by the Parikh mapping associated to the sequence) is insufficient: there are $\begin{pmatrix} x+y \\ x \end{pmatrix}$ binary strings α, having $|\alpha|_0 = x$ and $|\alpha|_1 = y$ (we denote by $|\alpha|_s$ the number of appearances of the substring s within the string α). Once the Parikh matrix mapping [Mateescu, A. *et al.*, 2001] has been defined and especially since the Parikh matrix mapping associated with the binary sequences [Atanasiu, A., 2007] has been studied (where $|\alpha|_{01}$ is also taken into consideration), the number of sequences defined by the same characteristics has been drastically decreased. However, due to the inherent non-injectivity of the Parikh matrix mapping, the possibility of identifying the strings by using this procedure is reduced (especially for the balanced sequences, when $|\alpha|_0$ is almost equal to $|\alpha|_1$). A step forward seems to be the use of the Istrail morphism, which extends the binary strings over a 3 - letter alphabet: $\{0, 1, 2\}$. The number of substrings of 012's is separating many binary words that were identical by means of the Parikh matrix mapping. Thus, only 98 from 5392 binary sequences having $|\alpha|_0 = 10$, $|\alpha|_1 = 10$ and $|\alpha|_{01} = 48$ have $|\alpha|_{012} = 770$. One could see that 98 is the maximum number of sequences (from 184756 possible) that keep the values constant: $(|\alpha|_0, |\alpha|_1, |\alpha|_{01}, |\alpha|_{012})$ – for other details see Example 2.5 below. One problem raised once the Istrail morphism is applied seems to be the handling of the three characters (instead of two), which would create difficulties in the binary representation of the numbers. As it will be shown, the value $|\alpha|_{012}$ associated to the image of the binary word through the Istrail morphism can be determined without effectively constructing the image of the word; thus only the binary alphabet will be used.

We can define a message authentication code based on the Parikh matrix mapping, completed with the Istrail morphism. In this case the collision problems can be theoretically analyzed and their numbers is greatly reduced than in the case of known MACs. Although this code does not assure a 100% analysis of integrity check, if it is combined with a cryptographic system (we propose the AES system) then some satisfying results may be obtained.

2.1 Message Authentication Code (MAC)

A cryptographic message authentication code (MAC) is a short piece of information used to authenticate a message. A MAC algorithm accepts as input a secret code (or the more popular term, key) and an arbitrary-length message to be authenticated, and outputs a MAC (sometimes known as a tag). The MAC value protects both a message's data integrity as well as its authenticity, by allowing verifiers (who also possess the secret code) to detect any changes to the message content.

Usually, MACs are created using hash functions or block encryption algorithms. There are also dedicated MACs. For details Schneier, B. [1996] can be used.

2.2 Defining a MAC using the Parikh Matrix Mapping

The construction is based on the Istrail morphism.

As in the previous chapters of this work, we will work with ordered alphabets. In this section we will consider a binary ordered alphabet $\Sigma_2 = \{a, b\}$ and its extension $\Sigma_3 = \{a, b, c\}$. In practice we will use the natural substitution

$$a \longleftarrow 0, \quad b \longleftarrow 1, \quad c \longleftarrow 2$$

(as it will be shown, c is an empirical element, for which we do not effectively need a character).

Let $w \in \Sigma_2^*$ be a text. By making use of the tuple $(|w|_a, |w|_b, |w|_{ab})$ the elements of the associated Parikh matrix mapping M_w assure an initial level of authenticity and integrity.

Thus, when we receive a message, we can control if the number of binary characters of each type has been preserved, as well as the number of scattered strings of the form 'ab'.

The only issue that remains is solving the amiability problem: one can send the message w; then the receiver gets w' with $M_w = M_{w'}$.

It has been shown [Atanasiu, A., 2007] that w and w' are nodes in the Γ_M graph, connected through a finite sequence of $ab - ba$ transformations.

In order to separate the two words, we will use the Istrail morphism[1] (reduced to the binary case)

$$h : \Sigma_2 \longrightarrow \Sigma_3$$

defined through:

$$h(a) = abc, \qquad h(b) = ac$$

Theorem 2.1. *Let $w = ab\alpha ba$ and $w' = ba\alpha ab (\alpha \in \Sigma_2^*)$ be two amiable words. Then $h(w)$ is not amiable with $h(w')$.*

Proof. By applying the morphism h we get

$$h(w) = abcach(\alpha)acabc, \quad h(w') = acabch(\alpha)abcac$$

It follows immediately that

$$|h(\alpha)|_a = |h(\alpha)|_c = |\alpha|_a + |\alpha|_b, \quad |h(\alpha)|_b = |\alpha|_a$$

therefore – obviously – $|h(w)|_a = |h(w')|_a$, $|h(w)|_b = |h(w')|_b$, $|h(w)|_c = |h(w')|_c$. Then:
$|h(w)|_{ab} = |h(w')|_{ab} = 5 + |h(\alpha)|_a + 2|h(\alpha)|_b + |h(\alpha)|_{ab}$,
$|h(w)|_{bc} = |h(w')|_{bc} = 5 + |h(\alpha)|_c + 2|h(\alpha)|_b + |h(\alpha)|_{bc}$.
However:
$|h(w)|_{abc} = 8 + |h(\alpha)|_a + 4|h(\alpha)|_b + |h(\alpha)|_c + 2|h(\alpha)|_{ab} + 2|h(\alpha)|_{bc} + |h(\alpha)|_{abc}$
$|h(w')|_{abc} = 12 + 2|h(\alpha)|_a + 4|h(\alpha)|_b + 2|h(\alpha)|_c + 2|h(\alpha)|_{ab} + 2|h(\alpha)|_{bc} + |h(\alpha)|_{abc}$.
Therefore

$$|h(w)|_{abc} < |h(w')|_{abc}$$

□

A problem could be the use of a third character (for $'c'$), which would rule out the use of binary numbers. This can be easily avoided, because the $|h(w)|_{abc}$ value can be directly determined from w, without calculating the $h(w)$ morphism.

Theorem 2.2. *Let $w \in \Sigma_2^*$ and $I = \{i \mid pr_i(w) = a\}$ ($pr_i(w)$ represents the i-th character from the w sequence). Then*

$$|h(w)|_{abc} = \sum_{i \in I} i \cdot (n + 1 - i).$$

Proof. The b characters from $h(w)$ appear only through the image of a's from w (because $|h(w)|_b = |w|_a$). Let us presume that $w = \alpha a \beta$ with $\alpha, \beta \in \Sigma_2^*$; therefore $h(w) = h(\alpha)abch(\beta)$. Then $|h(w)|_{abc}$ generated by this b is given by the product $(1+|h(\alpha)|_a)\cdot(1+|h(\beta)|_c)$. But $|h(x)|_a = |h(x)|_c = |x|$, and $|\alpha|+|\beta| = |w|-1$; so, by denoting $|\alpha| = i - 1$, we obtain the desired relation. □

From this theorem, other interesting collateral results can be proven; for example:

Corollary 2.3. $|(abc)^n|_{abc} = \begin{pmatrix} n+2 \\ 3 \end{pmatrix}$.

Example 2.4. *We will analyze the words having the Parikh vector $\Psi = (19, 2)$; we will observe that there are no more amiable words after applying the Istrail morphism ($\alpha \sim_a \beta$ but $h(\alpha) \not\sim_a h(\beta)$). Indeed:*

q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\lvert C_\alpha \rvert$	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10
$\#_h$	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10
$\#_{max}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The table reviews all the binary words having the Parikh mapping $\Psi = (19, 2)$. For each value of $q = |\alpha|_{ab}$, the second line shows the number of amiable words from C_α – thus having the following corresponding Parikh matrix mapping:

$$M = \begin{pmatrix} 1 & 19 & q \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

[1]The Istrail morphism is defined for an ordered alphabet $\Sigma_3 = \{a, b, c\}$ in this way: $h(a) = abc$, $h(b) = ac$, $h(c) = b$.

The third line gives the number of classes in which the set $X = \{h(w) \mid w \in C_\alpha\}$ is separated, and the last line gives the number of elements present in the most numerous such set.

Example 2.5. *The result obtained in the previous Example is an ideal case, situation which – unfortunately – is not always possible. So, if we take the case of the Parikh mapping $\Psi = (10, 10)$, the class repartition would be as follows:*

q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\lvert C_\alpha\rvert$	1	1	2	3	5	7	11	15	22	30	42	54	73	93	121	152	193	237	295
$\#_h$	1	1	2	3	5	7	9	15	19	25	31	45	41	67	65	65	79	99	93
$\#_{max}$	1	1	1	1	1	1	2	1	2	2	3	2	5	3	6	6	9	6	10

q	19	20	21	22	23	24	25	26	27	28	29	30	31
$\lvert C_\alpha\rvert$	356	433	515	615	720	847	978	1131	1289	1420	1652	1860	2065
$\#_h$	107	107	121	125	139	135	135	147	159	155	163	157	171
$\#_{max}$	8	14	17	16	13	23	24	25	28	34	25	47	35

q	32	33	34	35	36	37	38	39	40	41	42	43
$\lvert C_\alpha\rvert$	2293	2517	2761	2994	3246	3481	3729	3956	4192	4397	4609	4784
$\#_h$	171	183	181	175	187	195	189	195	187	199	197	207
$\#_{max}$	42	49	52	47	70	56	59	60	84	59	83	70

q	44	45	46	47	48	49	50
$\lvert C_\alpha\rvert$	4959	5095	5226	5311	5392	5424	5448
$\#_h$	203	195	205	211	203	207	197
$\#_{max}$	74	89	92	65	98	78	89

Several observations:

- *There are great fluctuations for the low-ranked classes regarding the maximum number of amiable words through the Istrail morphism.*

  Next – once the number of elements of a class is increased – the performance order is approximately 2% (this means that a maximum of 2% of the elements of a class of amiable words remain amiable through the Istrail morphism).

- *Case Study: for*

$$M = \begin{pmatrix} 1 & 10 & 20 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{pmatrix}$$

  the corresponding class of amiable words has 433 elements. The biggest set of words that stays amiable after applying the Istrail morphism is $C = \{w \in \Sigma_2^* \mid M_w = M,\ |h(w)|_{abc} = 750\}$.

  This set has 14 elements:
  $C = \{abbbbbbbabbaaaaaaaba, babbbbbabbabaaaaabaa, babbbbbbabaaabaabaaa,$
  $bbabbabbbbbaaaaaabaa, bbabbbabbbaabaaabaaa, bbabbbbaabbbaaaabaaa,$
  $bbabbbbababaabababaaaa, bbabbbbbaaaabbbaaaaa, bbbabababbbaaababaaaa,$
  $bbbabbababbabaabaaaa, bbbabbabbaababbaaaaa, bbbbaabbabbaabbaaaaa,$
  $bbbbabaabbabbabaaaaa, bbbbbaaaabbbbbaaaaaa\}$

In order to construct a MAC for a binary sequence $w \in \{0,1\}^*$, we will use the Parikh matrix mapping of w, to which we add the number $|h(w)|_{012}$. The obtained values are then encrypted using the AES system.

We shall use the Istrail morphism $h : \{0,1\} \longrightarrow \{0,1,2\}^*$ defined as follows:

$$h(0) = 012, \quad h(1) = 01$$

Also, we denote by $x\|y$ the concatenation of string x and y.

The algorithm for generating the code is the following:

Algorithm 2.6.
Input*:* The text $w \in \{0,1\}^*$.

1. Determine $A = |w|_0,\ B = |w|_1,\ C = |w|_{01}$;
2. Compute $h(w)$;
3. Determine $D = |h(w)|_{012}$;
4. Output $[A\|B\|C\|D]_{16}$.

As a remark, the output sequence is a number written in the hexadecimal base.

For the construction of the authentication code $MAC(w)$ several methods may be used. We suggest the following:

Algorithm 2.7.
Input*:*
- $X = [A\|B\|C\|D]_2 = b_1 b_2 \dots b_{4p}$, considered as a binary vector.
- IV – the binary (secret) sequence on 128 bits
(if the $AES-128$ system is used for encryption).

1. Construct the sequence $X' = b_1 b_3 \dots b_{4p-1} b_2 b_4 \dots b_{4p}$;
2. X' is divided in subwords of 128 bits: $X' = Y_1 Y_2 \dots Y_n$.
3. Iteratively construct the encryption sequence
$$Z_1 = e_{Y_1}(Y_2 \oplus IV),\ Z_i = e_{Z_{i-1}}(Y_{i+1}),\ i = 2, \dots, n-1$$
4. $MAC(w) \longleftarrow Z_{n-1}$.

where $e_K(\cdot)$ is the $AES-128$ encryption having the key K, and $\oplus$ is the bitwise XOR operation.

Remark 2.8. *If $|Y_n| = k < 128$, then it will be left-bordered by zeros. Therefore $Y_n \longleftarrow 0^{128-k} Y_n$.*

Remark 2.9. *If $AES-256$ system is used for encryption instead $AES-128$, then the lengths of binary strings used for the key and for IV are modified according to this parameter.*

Upon transmission, the message w is accompanied by the $MAC(w)$. For verifying the authenticity and the integrity of w, the receiver (who owns the secret value IV) remakes the calculations and verifies if the obtained authentication code matches the attached MAC.

2.3 Complexity

The elements of the Parikh vector (A, B) are determined through a linear algorithm.

Knowing A, it is straightforward to determine C too. For calculating D we use Theorem 2.2, which also leads to a linear algorithm.

Finally the algorithm for generating the code has time complexity $\mathcal{O}(n)$ and space complexity $\mathcal{O}(0)$. A detailed form of this algorithm is given below:

Algorithm 2.10.
Input*:*
- The current character $x \in \{0,1\}$ (an element from the message $w \in \{0,1\}^*$).
- Length $n = |w|$ of the message.

1. $A \longleftarrow 0,\ C \longleftarrow 0,\ D \longleftarrow 0;$

2. for $i \longleftarrow 1$ to n do

2.1. read x;

2.2. if $x = 0$ then

begin

$A \longleftarrow A + 1,\ \ D \longleftarrow D + i \cdot (n + 1 - i);$

end

else $C \longleftarrow C + A;$

3. $B \longleftarrow n - A.$

4. output $A\|B\|C\|D.$

2.4 Performance, Implementation Details

Since we are dealing with a linear algorithm, the performance (as expected) is high. The attempted implementation was made using Visual C++ on a 1,5 Ghz Mobile processor with 1,5 GB of RAM. The algorithm was applied to various input files - from text files (txt, pdf, doc) to several types of media files (jpg, bmp, mp3).

The algorithm performed as follows:

- For at most 1 KB files used as input, the algorithm was executed in less than 0,001 seconds;
- For around 0,5 MB files used as input, the algorithm was executed in at most 1,5 seconds;
- For at most 2,5 MB files used as input, the algorithm was executed in less that 6 seconds.

It should be noted that the algorithm might theoretically have $\mathcal{O}(0)$ size complexity. However, we have discovered that it is more time efficient to store the data in blocks and then parse them. The reason for this is very simple: it is more costly (regarding time) to access bitwise a file stored on a disk, rather than in data chunks. However, if other data channels are used, it might be the case that this approach is more costly than the one originally proposed by the algorithm. Such data channels might include traffic sent across a network or storage on high-speed access media. Since the algorithm is designed to be used for authenticating messages, then the $\mathcal{O}(0)$ approach should theoretically prove ideal, due to the fact that we read the information as it arrives through the transmission channel, making buffer storage not necessary. But in this case the message's length has to be provided before the text transmission.

As a conclusion, the implementation of the algorithm is useful for channels using streams of data, rather than for conventional storage.

3 Conclusions

The word analysis theory concerning Parikh Matrices is part of the bigger field of *combinatorics on words*. The theoretic aspects of this domain have quite a few practical applications. The software industry in the latest years has largely benefited from the need of searching and structuring the information. Injectivity of the Parikh matrix mapping has raised several challenges, as this aspect was framed under the notion of *amiability*.

In this paper we have proposed an application of amiability. This application is represented by a MAC(Message Authentication Code), constructed through the aid of the Istrail morphism. We aimed at both offering the theoretical support – in this sense we have given an algorithm for messages authentication and analyzed it from several points of view, such as complexity – and the practical counterpart, taking the form of implementing and testing the algorithm.

References

Atanasiu, A. (2007). Binary amiable words. *Int. J. Found. Comput. Sci. vol 18, no. 2, pp. 387-400*.

Atanasiu, A. and Atanasiu, R. (2008). Message authentication code based on parikh matrices. In *International Conference on Security for Information Technology and Communication*.

Mateescu, Al. and Salomaa, A. and Salomaa, K. and Yu, S. (2001). A sharpening of the Parikh mapping. *Theoret. Informatics Appl. 35, 551-564*.

Schneier, B. (1996). *Applied Cryptography*. John Wiley and Sons, Inc., 2nd edn.