Introducing security into prefix-free encoding schemes

Ruy Luiz Milidiu
milidiu@inf.puc-rio.br

Carlos José P. de Lucena
lucena@inf.puc-rio.br

Claudio Gomes de Mello
cgmello@inf.puc-rio.br

PUC-RioInf.MCC19/03 July, 2003

Abstract: Minimum redundancy prefix-free codes are widely used to obtain high performance compression schemes. Given a prefix-free encoding for the symbols of a plain text, we propose a security enhancement by adding a multiple substitution algorithm with a key: the HSPC2 - Homophonic Substitution Prefix-free Codes with 2 homophones. Breaking the key when we are given a ciphertext, the dictionary, frequencies and codeword lengths, is a NP-Complete problem. In order to introduce security, some compression loss is generated. The compression loss is analysed and the data expansion per character is asymptotically smaller than 5% under usual parsing and coding assumptions. We also present some analytical results on the security impact of adding simple strategies to protect prefix-free encoded data.

Keywords: data compression, prefix-free codes, security, homophonic substitution.

Resumo: Códigos livres de prefixo de redundância mínima são largamente utilizados para se obter esquemas de compressão de alto desempenho. Dada uma codificação livre de prefixo para um texto original, nós propomos um aumento na segurança adicionando um algoritmo de substituição múltipla com a utilização de um chave: o HSPC2 - códigos livres de prefixo de substituição homofônica com 2 homofônicos. Quebrar a chave quando nos são dados o texto cifrado, o dicionário, as frequências e os comprimentos das palavras de código, é um problema NP-Completo. Com o objetivo de se introduzir segurança, alguma perda de compressão é gerada. A perda de compressão é analisada e a expansão dos dados por caracter é assintoticamente menor que 5% supondo codificação e varredura usuais. Nós também apresentamos resultados analíticos no impacto da segurança ao se adicionar estratégias simples para proteger dados codificados livres de prefixo.

Palavras-Chave: compressão de dados, códigos livres de prefixo, segurança, substituição homofônica.

1 Introduction

When using information retrieval systems, digital data usually goes through two separate processes: data compression to achieve low storage and transmission costs, and ciphering to provide security. Adding some additional strategies to data compression schemes, such as static Huffman [Huff52] coding, usually provides both compressed and encrypted data. By adding security into prefix-free encoding schemes, we reduce two different processes to a single one. That improves computational efficiency. Additionally, information retrieval features, such as indexing and searching [Moura97], are kept the same. The possible overhead is some data expansion due to the encryption approach.

Wayner [Wayn88] proposed a simple scheme for assigning a secret key to a Huffman tree. In his scheme, he obtains a new optimal tree by operating an exclusive-or (XOR) between each branch of the tree with a secret key. A simple version of this approach is to assign one bit of a secret key for each level of the tree. In this case, he operates an XOR for each Huffman code with the corresponding position of the key. The size of the key can be too small in this last, says O(log n). Klein et al. [Klein89a] analysed the cryptographic aspects of Huffman codes used to encode a large natural language on CD-ROM. And, in [Klein89b], Fraenkel and Klein show that this problem is NP-complete for several variants of the encoding process. Rivest et al. [Rive96] cryptoanalysed a Huffman encoded text assuming that the cryptoanalyst does not know the codebook. According to them, cryptoanalysis in this situation is surprisingly difficult and even impossible in some cases due to the ambiguity of the resulting encoded data. In [Mili00, Mili01a, Mili01b] data compression rates in the range of 40% to 50% are shown. Furthermore, observed coding and decoding times were very close to the standard Canonical Huffman codes. Moreover, experiments show that when using dyadic distributions we can even increase the encoding speed generating only a small loss in compression rates. A dyadic probability distribution is a distribution in which each probability is a negative integer power of 2. As an example, (2^{-2}, 2^{-2}, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-4}) is a dyadic distribution. Also, it is shown that dyadic distributions increase the secrecy of the ciphertext since prefix-free coding of a dyadic distribution source leads to a random stream of bits.

In this paper, we present a theoretical study of the security impact of adding some strategies to prefix-free codes motivated by our previous empirical results. This work extends previous results [Mili00, Mili01a, Mili01b] on practical implementation of data ciphering-compressing algorithms using Canonical Huffman coding, dyadic distributions and approaches intended to secure the ciphertext against cryptanalysis. We propose a scheme that adds security into the compression process by using a homophonic substitution algorithm with a key: the HSPC2 - Homophonic Substitution Prefix-free Codes with 2 homophones. Multiple, or also called homophonic, substitution is an old technique that transforms a plain text sequence of symbols into a more random one. Each symbol has multiple homophones that can be chosen to represent it. This technique avoids statistical attacks to prefix-codes [Gunt88, Mass89]. We prove that the use of homophonic substitution, as in HSPC2, increases the security of the ciphertext. We assume that the cryptoanalyst is given a ciphertext, the dictionary, frequencies and codeword lengths. His goal is to break the key used during the encoding process. We show that the HSPC2 problem is a NP-Complete problem. In the encoding process, the HSPC2 function appends a bit suffix to some codes. A key and a relative frequency control this appending. According to the values of these two items, the algorithm chooses which instances of the symbol receive the bit suffix. Then, we can have different ciphertexts to the same plain text and key due this homophonic substitution approach. This work is similar to [Klein89b], but we use a different approach. Fraenkel and Klein use variations of the strategy of having a fixed suffix for

each symbol. In his work, the size and the presence or not of the suffix is the secret. On the other hand, we use homophonic substitution.

Section 2 describes the proposed HSPC2 function. Section 3 shows the related HSPC2 problem. The compression loss is analysed and the data expansion due to the homophonic substitution approach is asymptotically smaller than 5%, under usual parsing and coding assumptions. In section 4, we prove that HSPC2 is NP-Complete. In section 5, we present our conclusions and some guidelines to future work.

2 The HSPC2 Function

The issues that arise when using data compression schemes have been examined by cryptographers over the years. It is well known that compressing data is not secure enough against simple analysis such as statistical attacks. To protect data against statistical analysis, Shannon [Shan49] suggested that the language redundancy should be reduced before encryption. Data compression can be used to achieve this.

Homophonic Substitution Prefix-free Codes with 2 homophones (HSPC2) is a security enhancement to prefix-free coding. HSPC2 can be added to any prefix-free code such as static Huffman or Canonical Huffman [Moff94] codes, for instance. Next, we present an example that illustrates the HSPC2 approach.

Example 1. *Define a dictionary Σ = (a, b, c), a plain text T = abaabbccc, a binary vector key K = (1, 1, 0) and a prefix-free code C = (00, 01, 1). In this case, the HSPC2 function uses the M transformation given by*

$$M_i \equiv M_{k_i}(t_i) = \begin{cases} c_i , & \text{if } k_i=0 ; \\ c_i , & \text{if } k_i=1 \text{ with relative frequency 2/3 ;} \\ c_i.b_i , & \text{if } k_i=1 \text{ with relative frequency 1/3, where } b_i = 0 \text{ if } i \text{ is even and } b_i = 1 \text{ if } i \text{ is odd.} \end{cases}$$

Tabel 1 shows the M_i values that one can obtain when encoding the given text T.

i	t_i	c_i	k_i	b_i	M_i
1	a	00	1	1	00
2	b	01	1	0	01
3	a	00	1	1	001
4	a	00	1	0	00
5	b	01	1	1	011
6	b	01	1	0	01
7	c	1	0	1	1
8	c	1	0	0	1
9	c	1	0	1	1

Tabel 1 - M_i values

Therefore, the final ciphertext is the sequence of bits *00010010001101111*.

Observe that we use three homophones for each symbol. We also introduce a key k and a relative frequency p to control the substitution scheme. If k=0 then the first homophone c_1 is chosen, otherwise either the second homophone c_2 is chosen with relative frequency p, or the third homophone c_3 is chosen with relative frequency 1-p.

In fact, HSPC2 uses two distinct homophones. Two homophones are defined by $c_2 = c_1$ and the other one by $c_3 = c_1.b$, where b is either 0 or 1. In example 1, the first symbol appears three times in T. Since $k_1=1$, we must have, in the ciphertext S, 1/3 of the occurrences being coded as $00.b_i$ and 2/3 of the occurrences being coded as 00. The choice of which homophone we use to represent the symbol can be made at random and does not matter to the decoding process, since the homophonic code is prefix-free. However, that is important to add secrecy to the encoding process. We have a similar situation for the second symbol, since $k_2=1$. And all the occurrences of the third symbol are coded as 1 since $k_3=0$.

Now we define the HSPC2 function, that introduces security into prefix-free encoding schemes.

HSPC2 Function. *Let us be given a prefix-free code $C = (c_1,...,c_n)$ for a dictionary $\Sigma = (\sigma_1,.., \sigma_n)$ of n distinct symbols, a plain text $T = t_1...t_s$ with $t_i \in \Sigma$, a key $K = (k_1,...,k_n)$ with $k_i \in \{0,1\}$, the number s of symbols in T, the frequency f_i of each distinct symbol σ_i in T, a real number q in the range [0, s], and a binary vector $B=(b_1,...,b_s)$. HSPC2 generates a ciphertext S, such that $S = M_K(T) = M_{k_1}(t_1)... M_{k_s}(t_s)$, where M is the transformation given by*

$$M_i \equiv M_{k_i}(t_i) = \begin{cases} c_i , & \text{if } k_i=0 ; \\ c_i , & \text{if } k_i=1 \text{ with relative frequency } 1-(q/s) ; \\ c_i.b_i , & \text{if } k_i=1 \text{ with relative frequency } q/s . \end{cases}$$

Observe that C must be a consistent prefix-free code, that is, the set $\{c_1, ..., c_n\}$ must be a prefix-free set. Hence, no codeword c_i is a prefix of another c_j, where $i \neq j$, and if $c_i = c_j$, then i=j.

If $k_i=1$, HSPC2 uses either c_i or $c_i.b_i$ at random to encode symbol σ_i with relative frequencies of occurrence 1-(q/s) and q/s. The binary suffix b_i can be arbitrarily chosen. Some simple policies to set b_i are

(i) at random;
(ii) alternating 0´s and 1´s, that is, $b_i = 0$ if i is even and $b_i = 1$ if i is odd;
(iii) based on some fixed rule, for example: $b_i=0$ if i mod 10 = 0, and 1 otherwise.

Note that, for each symbol σ_i that has $k_i = 1$, the number of instances of σ_i in the ciphertext S that must be encoded with homophone $c_i.b_i$ is equal to $(q/s).f_i$. In example 1, we have (1/3).(3) = 1 symbol encoded with the homophonic code $00.b_i$, and (2/3).(3) = 2 symbols encoded with the homophonic code 00. Since $(q/s).f_i$ is not always an integer, we set the ceiling $\lceil (q/s).f_i \rceil$ as the number of occurrences of the homophone $c_i.b_i$ in S when its key is 1. We set the ceiling operator to minimize zero results. The floor operator generates more zeros than ones, and so, fewer homophones $c_i.b_i$, and we think it weakens the HSPC2 scheme. In section 3.2 we show that either the floor or the ceiling value results in valid choices to our purposes.

3 The HSPC2 Problem

Next, we present an example that illustrates what we call the HSPC2 problem.

Example 2. *Given Σ, F, R, S and q, where Σ = (a, b, c), F = (3, 4, 2), s = 9, R = (2, 1, 2), S = 100001011011100, and q = 3. Are there a consistent prefix-free code C, a plain text T and a key K such that $M_K(T)$ = 100001011011100, where $M_K(T)$ is the HSPC2 function ?*

In example 2, the answer is yes, since the inquired items can be chosen as C = (00, 1, 01), T = baacbcbba and K = (1, 0, 0).

The related HSPC2 problem is defined as the following decision problem.

HSPC2 Problem
Input: a dictionary $\Sigma = (\sigma_1, ..., \sigma_n)$ of n symbols, the vector $F = (f_1, ..., f_n)$ of frequencies f_i, the vector $R = (r_1, ..., r_n)$ of codeword lengths, the ciphertext S obtained by $S = M_K(T)$, and $q \in [0, s]$, where $s = f_1 + ... + f_n$.
Question: Are there a consistent prefix-free code C, a plain text T and a key K such that $M_K(T) = S$, where $M_K(T)$ is the HSPC2 function ?

Note that only the codeword lengths are provided. In fact, the vector R is given because it is an intrinsic information that can be derived from Σ, F and the knowledge of the adopted prefix-free coding algorithm. An interesting feature here is that Σ and F are not secrets, and hence, do not need to be protected. Moreover, the prefix-free code C is not completely defined by R, due to the remaining ambiguity [Rivest96]. Furthermore, the order the symbols with the same frequencies are taken, and the unknown choices made in the encoding process (like varying between left-order and right-order Huffman coding) also contribute to increase the security of the ciphertext

3.1 The RHSPC2 Problem

Now, let us consider a simpler problem denoted by RHSPC2. RHSPC2 is obtained by adding the following two restrictions to HSPC2.

(i) *The text layout is known*. The plain text T that we use in RHSPC2 is defined by a concatenation of groups of repeated symbols, that is, $T = \sigma_1^{f_1}.\sigma_2^{f_2}.\sigma_3^{f_3} ....\sigma_n^{f_n}$. As an example, let $T = aabbbbcc = a^2b^4c^2$. As a consequence, we have that the layout of T and the vector of frequencies F define T.

(ii) *The codebook is known*. From the knowledge of the text layout, the codeword lengths and frequencies, it is possible to build a valid consistent prefix-free code C. Hence, in RHSPC2, the codebook is given.

The RHSPC2 problem is defined as the following decision problem.

RHSPC2 Problem
Input: a dictionary $\Sigma = (\sigma_1, ..., \sigma_n)$ of n symbols, the vector $F = (f_1, ..., f_n)$ of frequencies f_i, a plain text defined by $T = \sigma_1^{f_1}.\sigma_2^{f_2}.\sigma_3^{f_3} ....\sigma_n^{f_n}$, the codebook $C = (c_1, ..., c_n)$, with codeword lengths $R = (r_1, ..., r_n)$, an integer L, with $L \geq f_1r_1 + ... + f_n.r_n.$, and $q \in [0, s]$, with $s = f_1 + ... + f_n$.
Question: Is there a key K such that $|M_K(T)| = L$, where $M_K(T)$ is the HSPC2 function ?

Only the size of $M_K(T)$ is important. Note that more than one codebook can result in the same size of $M_K(T)$, that is, we can have codebooks C_1 and C_2, with the same codeword lengths, both resulting in $|M_K(T)| = L$. This ambiguity is pointed out by [Rive96]. In RHSPC2 problem, one valid codebook is given, but others can exist.

3.2 Ciphering data expansion

In RHSPC2, the plain text is not a secret, and its layout is given by $T = \sigma_1^{f_1}.\sigma_2^{f_2}.\sigma_3^{f_3} ....\sigma_n^{f_n}$. L is an integer obtained by $L = |M_K(T)| = |S|$, with $L \geq \Sigma r_i.f_i$. First, let us calculate the size of $M_K(T)$.

$$|M_K(T)| = |S| = \Sigma r_i.f_i.(1-k_i) + \Sigma (r_i+1).\lceil (q/s) \cdot f_i \rceil.k_i + \Sigma r_i.(f_i - \lceil (q/s) \cdot f_i \rceil).k_i =$$
$$= \Sigma r_i.f_i + \Sigma \lceil (q/s) \cdot f_i \rceil.k_i =$$
$$= |S_0| + \Sigma \alpha_i.k_i$$

where $|S_0| = \Sigma r_i.f_i$ *and* $\alpha_i = \lceil (q/s) \cdot f_i \rceil$, $\alpha_i \in I$.

Hence, $L = |M_K(T)| = |S| = |S_0| + \Sigma\alpha_i.k_i$. Finding the solution of the RHSPC2 problem is equivalent to finding K such that $|S_0| + \Sigma\alpha_i.k_i = L$.

Note that defining the floor or the ceiling value for the number of symbols that must be encoded with homophone $c_i.b$ are both valid choices related to L.

The value $D = (|S| - |S_0|) / |S_0| = \Sigma\alpha_i.k_i / \Sigma r_i.f_i$ represents the ciphering data expansion induced on the ciphertext S. Since $\Sigma r_i.f_i \geq \Sigma f_i = s$, we have

$$D = \Sigma\alpha_i.k_i / \Sigma r_i.f_i \leq \Sigma \lceil (q/s) \cdot f_i \rceil.k_i / s \leq \Sigma (1+ (q/s) \cdot f_i).k_i / s \leq (n+q)/s,$$ *since* $k_i \leq 1$ *for* $i=1,...,n$.

Observe that D is an increasing function of q. Hence, q can be used to control the extra bits overhead due to ciphering. The data expansion depends also on the vectors K and F. To calculate the expected data expansion let us introduce some probabilistic assumptions. First, let us assume that q, K and F are independent. This is true for RHSPC2 since given F, the key K and the number q are chosen at random, independently of F. In this case, $E[q] = s/2$ and $E[k_i] = p[k_i=1] = 1/2$.

For a fixed plain text size, we have

$$E[D] \leq E[\Sigma (1+ (q/s) \cdot f_i).k_i / s] = E[\Sigma k_i / s + \Sigma (q/s) \cdot f_i \cdot k_i / s]$$
$$= E[\Sigma k_i] / s + E[(q/s^2) \cdot \Sigma f_i \cdot k_i]$$
$$= n/2s + 1/s^2 \cdot E[q] \cdot E[\Sigma k_i.f_i] =$$
$$= n/2s + 1/s^2 \cdot (s/2) \cdot \Sigma (E[k_i] \cdot E[f_i]) =$$
$$= n/2s + 1/2s \cdot \Sigma (1/2 \cdot E[f_i]) =$$
$$= n/2s + 1/4s \cdot \Sigma E[f_i]$$
$$= n/2s + 1/4s \cdot E[\Sigma f_i]$$
$$= n/2s + 1/4s \cdot s$$
$$= n/2s + 1/4 \approx 1/4 = .25,$$ *when* $s >> n$.

The expected ciphering data expansion E[D] is asymptotically smaller than 25% bits per symbol for usual parsing and coding assumptions. Note that if the symbol is a character then the data expansion is asymptotically smaller than 25%. On the other hand, if the symbols are words or n-grams, the data expansion is considerably lower. Hence, expected data expansion per character can be asymptotically bounded as $E[D] \leq .25/t$ for t-gram parsing codes. As an example, for a 5-gram parsing, or similarly for word parsing, the bound is 5%. Moreover, since D depends on q, K and F, one can

decrease data expansion adopting other policies to choose q and K. For instance, for $E[k_i] = \beta$ we have $E[D] = \beta/2$ bits per symbol, where β is in the range [0, 1/2). The disadvantage here is a lack of secrecy, since this different probabilistic distribution for K can be used by cryptoanalysts.

4 HSPC2 is NP-Complete

In this section we show that breaking the key used in the ciphertext $S = M_K(T)$ is a NP-Complete problem.

4.1 Reductions

Now, we use a reduction from SUBSET-SUM, a well-known [Cormen90] NP-Complete problem.

The SUBSET-SUM (SUS) Problem
Input: a vector $A = (\alpha_1, ..., \alpha_n)$, $\alpha_i \in N$, and a goal g, $g \in N$, with $g \leq (\alpha_1 + ... + \alpha_n)$.
Question: Is there a binary vector $K = (k_1, ..., k_n)$ such that $\alpha_1.k_1 + ... + \alpha_n.k_n = g$?

Theorem 1. *SUS $\propto$ RHSPC2.*

Proof: We want to prove that RHSPC2 is NP-Hard. The following three conditions must hold.

(i) *There exists a reduction algorithm Δ from SUS to RHSPC2.* Suppose that the SUS problem is defined by $A=(\alpha_1, ..., \alpha_n)$ and goal g. Let us build the RHSPC2 problem from the SUS problem: generate an alphabet $\Sigma = (\sigma_1,...,\sigma_n)$ with n symbols. Choose an integer q in the range [0, s]. The frequency of each symbol is defined by $f_i = \lfloor (s/q).\alpha_i \rfloor$. T is defined by $T = \sigma_1^{f_1}.\sigma_2^{f_2}.\sigma_3^{f_3} .... \sigma_n^{f_n}$. Obtain a codebook applying the prefix-free coding algorithm defined for HSPC2. L is defined by $L = |S_0| + g$, hence $L \in N$.
(ii) *Δ is polynomial.* Generating the dictionary, choosing q, calculating f_i, defining T and obtaining a codebook (using Huffman [Huff52] codes, for instance) are all polynomial operations. Hence, the total complexity is polynomial.

(iii) *Δ is correct.* Let K be a given binary vector. K is a solution to SUS defined by A and g if and only if K is a solution to RHSPC2 defined by Σ, F, T, C, q and $L = (|S_0| + g)$. First, suppose that K is a solution to RHSPC2, so we have $|M_K(T)| = L$.

$$
\begin{aligned}
|M_K(T)| &= |S_0| + \sum \lceil (q/s) \,.\, f_i \rceil k_i = |S_0| + \sum \lceil (q/s) \,.\, \lfloor (s/q) \,.\, \alpha_i \rfloor \rceil k_i \\
&= |S_0| + \sum \lceil (q/s) \,.\, (((s/q) \,.\, \alpha_i) - \varepsilon) \rceil k_i,\ 0 \leq \varepsilon < 1,\ (s/q) \in \Re,\ (s/q) \geq 1,\ \alpha_i \in N \\
&= |S_0| + \sum \lceil \alpha_i - \varepsilon/(s/q) \rceil k_i,\ 0 \leq \varepsilon < 1 \\
&= |S_0| + \sum \lceil \alpha_i - \delta \rceil k_i,\ 0 \leq \varepsilon < 1,\ 0 \leq \delta \leq \varepsilon < 1 \\
&= |S_0| + \sum \alpha_i.k_i = L = |S_0| + g
\end{aligned}
$$

Hence, $\Sigma \alpha_i.k_i = g$ and K is a solution to the SUS problem.

Now, suppose that the RHSPC2 problem is defined by Σ, F, R, L and q. Then, the corresponding SUS problem is defined by $A=(\alpha_1, ..., \alpha_n)$ and goal $g = L - |S_0|$, where α_i is defined by $\alpha_i = \lceil (q/s) \,.\, f_i \rceil$. If K is a solution to SUS, then $\alpha_1.k_1 + ... + \alpha_n.k_n = g$.

Hence, $|M_K(T)| = |S_0| + \alpha_1.k_1 + ... + \alpha_n.k_n = |S_0| + g = |S_0| + L - |S_0| = L$. Hence, K is a solution to the RHSPC2 problem.

From (i), (ii) and (iii), we prove that SUS $\propto$ RHSPC2.

In order to prove the next reduction, we consider now that all plain texts are represented in a run-length encoding scheme. For instance, the plain text *T = abaabbccc* of example 1 is represented as *T = 1a1b2a2b3c*. This modification in HSPC2 does not generate any lack of generality, and the operation of changing representation is polynomial. This run-length representation can reduce the size of the resulting plain text if all symbols in T are grouped. On the other hand, can double the size if the symbols are distributed over the plain text.

Theorem 2. *RHSPC2 $\propto$ HSPC2.*

Proof: Theorem 1 proves that RHSPC2 is NP-Hard. Now, we want to prove that HSPC2 is NP-Hard. The following three conditions must hold.

(i) *There exists a reduction algorithm Δ from RHSPC2 to HSPC2.* Suppose that the RHSPC2 problem is defined by a dictionary $\Sigma = (\sigma_1, ..., \sigma_n)$ of n symbols, the vector $F = (f_1, ..., f_n)$ of frequencies f_i, a plain text defined by $T = \sigma_1^{f_1}.\sigma_2^{f_2}.\sigma_3^{f_3} ....\sigma_n^{f_n}$, the codebook $C = (c_1, ..., c_n)$, with codeword lengths $R = (r_1, ..., r_n)$, an integer L, with $L \geq f_1r_1 + ... + f_n.r_n.$, and $q \in [\ 0, s\ ]$, with $s = f_1 + ... + f_n$. . Building the HSPC2 problem from the RHSPC2 problem is immediate: consider the same dictionary, frequencies, codeword, codeword lengths and q. Now, generate the plain text T´ of HSPC2 in run-length representation: $T' = f_1\sigma_1\ f_2\sigma_2\ f_3\sigma_3 ... f_n\sigma_n$. Note that this run-length representation used by HSPC2 can represent any plain text, especially the one used by RHSPC2.

(ii) *Δ is polynomial.* HSPC2 uses all input vectors of RHSPC2, except the plain text that is easily derived. Hence, the total complexity is polynomial.

(iii) *Δ is correct.* Let K be a given binary vector. K is a solution to RHSPC2 defined by Σ, F, T, C, q and $L = |S| = |M_K(T)|$ if and only if K, C and T' are a solution to HSPC2 defined by Σ, F, R and q. T´ is the run-length representation of T. First, suppose that K is a solution to RHSPC2. Hence, $L = |S| = |M_K(T)|$. Moreover, for the codeword C and plain text T, we have that $S = M_K(T)$. Since T´ and T are equivalent representations when the symbols are grouped, we have $M_K(T') = M_K(T)$. Hence, K, C and T´ are a solution to HSPC2.

Now, suppose that K, C and T' are solution to HSPC2. Hence, $S = M_K(T)$ and $L = |S| = |M_K(T')|$. Moreover, note that symbols in T´ are not necessarily grouped, but we have $|M_K(T')| = |M_K(T)| = L$. Hence, we do not need to group symbols in T to verify that K is a solution to RHSPC2.

From (i), (ii) and (iii), we prove that HSPC2 is NP-Hard.

Observe that in HSPC2, the cryptoanalyst do not know the text layout, that is, it is necessary to guess the partitions of the ciphertext S by exhaustive search consisting in dividing the stream of bits S into s non-empty codes, where s, defined by $s = f_1 + f_2 + ... + f_n$, is the number of symbols in T. The brute force analysis of the number of different combinations is first pointed out by [Klein89a] and has to consider an exponential number of possible partitions in S. And is under the following constraints: the set of the different codewords in the sequence must be non-empty and we must eliminate the sets that are not consistent prefix-free codes.

Moreover, we can have the ambiguity problem in finding the codebook when cryptoanalysing HSPC2. We have ambiguous codes when it is possible to decode an encoded data using two or more valid codebooks, that is, if a bit stream can be decoded using either codebook C_1 or C_2, resulting in equally valid possible plain texts. Therefore, prefix-free codes like Huffman codes lead to ambiguous encoded data. Code ambiguity is pointed out by Rivest et al. [Rive96], which have cryptoanalysed a Huffman encoded data assuming that the cryptoanalyst does not know the codebook. According to them, cryptoanalysis in this situation is surprisingly difficult and even impossible in some cases due to the ambiguity of the resulting encoded data.

4.2 HSPC2 is NP-Complete

Theorem 3. HSPC2 is NP-Complete.

Proof: HSPC2 is NP-Complete if the following two conditions hold: *(i) HSPC2 is in NP; (ii) HSPC2 is in NP-Hard.*

To prove (i), we assume that we are given a certificate C, T and K. Then, it is immediate to verify if it is true or not that $S = M_K(T)$. We get (ii) from theorem 2. Hence, HSPC2 is NP-Complete.

5 Conclusions and future work

The issues that arise when using data compression schemes have been examined by cryptographers over the years. It is known that compressing data is not secure enough against simple analysis such as statistical attacks. In this paper, we propose a security enhancement to the encoding process by using a homophonic substitution algorithm with a key: the HSPC2 - Homophonic Substitution Prefix-free Codes with 2 homophones.

This work shows how to add a homophonic strategy to prefix-free data compression algorithms. This enhancement aims to increase the security of information retrieval systems. Through the results presented here, we obtain a theoretical analysis of the security feature that is added to a modified prefix-free data compression code. It also provides simple guidelines to practical implementations of data ciphering-compressing algorithms using Canonical Huffman coding, dyadic distribution and other experimental strategies intended to secure the ciphertext against cryptanalisys. We plan to integrate this new strategy to a practical implementation and test its empirical performance.

One mayor advantage of the HSPC2 function is that information retrieval features, such as indexing and searching [Moura97], are kept the same. The possible overhead is some data expansion due to the encryption approach, but analyses under usual assumptions show that, for word parsing, the compression loss is asymptotically smaller than 5%.

Possible other strategies can be added to this scheme to achieve lower data expansion and better performance, therefore, the theoretical and practical impact in security due to modifications in the algorithm must be analysed. For instance, if the secret key K is dependent on the plain text T, say $k_i = 1/f_i$, then we can have lower data expansion, but it is an open problem the impact of this modification in secrecy. Also, other related techniques can be used to optimize the overall algorithm like skeleton trees [Klein97] and dictionary reducing schemes to large scale texts [Zobel99].

References

[Cormen90] Cormen, T. H., Leiserson, C. E., Rivest, R. L. 1990. *Introduction to Algorithms.* The MIT (The Massachusetts Institute of Technology) Press.

[Gunt88] Gunter, C.G. 1988. *An Universal Algorithm for Homophonic Coding in Advances in Cryptology*. Eurocrypt-88, LNCS, vol. 330.

[Huff52] Huffman, D. 1952. *A Method for the Construction of Maximum of Minimum Redundancy Codes*. Proc. IRE, 1098-1101.

[Klein89a] Klein, S. T., Bookstein, A., Deerwester, S. 1989. *Storing Text Retrieval Systems on CD-ROM: Compression and Encryption Considerations*. ACM Transactions on Information Systems, vol. 7, no. 3

[Klein89b] Klein, S.T., Fraenkel, A.S. 1989. *Complexity Aspects of Guessing Prefix Codes*. Algorithmica 12 409-419.

[Klein97] Klein, S.T. 1997. *Skeleton Trees for the Efficient Decoding of Huffman Encoded Texts*. 8th Annual Symposium on Combinatorial Pattern Matching (CPM'97), 65-75.

[Mass89] Massey, J.L., Kuhn, Y.J.B., Jendal, H.N. 1989. *An Inf*ormation-*Theoretic Treatment of Homophonic Substitution*. in Advances in Cryptology Eurocrypt-89, LNCS, vol. 434.

[Mili01a] Milidiú, R.L., Mello, C.G, Fernandes J.R., Nov 2001. *Adding security to compressed information retrieval systems.* SPIRE 2001, Chile.

[Mili01b] Milidiú, R.L., Mello, C.G, Fernandes J.R., Mar 2001. *Substituição Homofônica Rápida via Códigos de Huffman Canônicos*. Wseg 2001 (Workshop on Computer Systems Security), Florianópolis, SC, Brazil.

[Mili00] Milidiú, R.L., Mello, C.G., Fernandes J.R., Nov 2000. *A Huffman-based text encryption algorithm*. SSI 2000 (Computer Security Symposium), pp. 11-17, São José dos Campos, SP, Brazil.

[Moff94] Moffat, A., Witten, I.I., Bell and Timothy C. Bell. 1994. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold.

[Moura97] Moura, E., Navarro, G., and Ziviani, N., 1997. *Indexing compressed text.* In N. Ziviani, R. Baeza-Yates, and K. Guimaraes, editors, Proceedings of the 4th South American Workshop on String Processing. Carleton University Press.

[Rive96] Rivest, R.L., Mohtashemi, M., Gillman, David W. 1996. *On Breaking a Huffman Code*. IEEE Transactions on Information Theory, vol. 42, no. 3.

[Shan49] Shannon, C. 1949. *Communication Theory of Secrecy Systems*. Bell Syst. Tech., vol. 28, no. 4, pp. 656-715.

[Wayn88] Wayner, P. 1988. *A Redundancy Reducing Cipher*. Cryptologia, 107-112.

[Zobel99] Zobel, J., Williams, H.E. 1999. *Compact In-Memory Models for Compression of Large Text Databases*. SPIRE´99 (String Processing and Information Retrieval), 224-231