9
Códigos de Prefixo baseados em Substituição Homofônica com 2 homofônicos

9.1
Introdução

Neste capítulo será apresentado o algoritmo HSPC2 - Códigos de Prefixo baseados em Substituição Homofônica com 2 homofônicos. No processo de codificação, o algoritmo adiciona um bit de sufixo em alguns códigos. Uma chave secreta e uma taxa de homofônicos são parâmetros que controlam essa inserção. É mostrado que a quebra do HSPC2 é um problema NP-Completo.

Este trabalho foi publicado no *DCC 2006 - Data Compression Conference*, USA, 2006, *CIBSI 05 - 3o Congreso Iberoamericano de Seguridad Informática*, Chile, 2005, e nos relatórios técnicos *PUC-RioInf.MCC19/03* July, Brazil, 2003 e *PUC-RioInf.MCC50/04* December, Brazil, 2004.

9.2
Função de Substituição Homofônica

É proposto um algoritmo para os códigos de prefixo denominado *HSPC2 - Códigos de Prefixo baseados em Substituição Homofônica com 2 homofônicos*, cuja finalidade é melhorar a segurança dos códigos. HSPC2 pode ser incluído em qualquer código de prefixo tais como os códigos Huffman padrão e Huffman Canônico (22). É baseado em substituição homofônica.

Uma função de codificação é adicionada ao processo de codificação de prefixo, tornando difícil a quebra do código. Essa força criptográfica é analisada e o problema de decisão associado é reduzido ao problema do SUBSET-SUM (4) para demonstrar a NP-Completude como argumento da segurança do algoritmo. As funções de indexação e busca são mantidas (23). É feita uma comparação com o trabalho correlato desenvolvido por Klein et al. (9).

A seguir a explicação sobre a construção do algoritmo de substituição homofônica proposto, o HSPC2.

Tabela 9.1: Livro de códigos original.

Símbolo	Código
a	00
b	1
c	01

Tabela 9.2: Livro de códigos homofônicos.

Símbolos	Homofônicos
a	00, 000, 001
b	1, 10, 11
c	01

9.2.1 Taxa Homofônica

Considere um texto original T dado por `bcaabbbcc`. Para esse texto assuma um dicionário de 3 símbolos distintos $\{a, b, c\}$ e o livro de códigos de prefixo definidos pela tabela 9.1. Para definir quais símbolos têm homofônicos uma chave secreta K é utilizada. K é um vetor binário: se $k_i = 1$ então o símbolo i é codificado com a função de substituição homofônica, caso contrário, se $k_i = 0$, uma simples substituição é usada.

Por exemplo, assuma que sejam escolhidos os símbolos a e b para serem atribuídos aos códigos homofônicos, isto é, a chave secreta é $K = (1, 1, 0)$. O código homofônico estendido é dado pela tabela 9.2.

A introdução de códigos de prefixo homofônicos extra resulta em perda na compressão. É empregado um parâmetro de taxa homofônica com a finalidade de controlar esse efeito colateral. Nesse exemplo, considere a taxa como sendo $\frac{1}{3}$, ou seja uma em cada 3 ocorrências do símbolo deverá ser codificada com os códigos homofônicos.

A taxa homofônica de um dado símbolo i é definida por,

$$h_i = \lceil \frac{q}{m} . f_i \rceil$$

onde $i \in [0, n]$ e f_i representa a frequência do símbolo i com $f_i \in \aleph$.

Por exemplo, um possível texto cifrado do texto original `bcaabbbcc` é `101000011110101 01`.

k_i é o bit secreto da chave do símbolo i com $k_i \in \{0, 1\}$. Se $k_i = 0$ então a substituição simples é utilizada. Caso contrário, h_i é um inteiro $\frac{q}{m}$ de

Tabela 9.3: Transformação Homofônica.

Símbolo	Taxa Homofônica h_i	Homofônicos
a	1	00, 000, 001
b	2	1, 10, 11
c	0	01

Tabela 9.4: Taxas Homofônicas.

Símbolo	Chave k_i	Frequência f_i	Taxa Homofônica h_i
a	1	2	1
b	1	4	2
c	0	3	0

frequência f_i do símbolo i. Então, q pode ser usado para controlar a proporção entre símbolos que têm homofônicos e símbolos que não têm homofônicos. A chave K e a taxa homofônica h_i definem 3 possíveis codificações para um símbolo, a função de codificação HSPC2.

A Tabela 9.3 sumariza o esquema de codificação com $q = 1$ e $m = 3$, onde q é um valor inteiro de forma que $q \in (0, m]$, e m é o tamanho do texto original. A fim de se obter uma taxa de ocorrência inteira, e também reduzir a ocorrência de taxas nulas para os símbolos homofônicos correspondentes, é adotado um arredondamento para cima como mostrado na tabela 9.4.

9.2.2 Função de codificação HSPC2

Nesta seção a função de codificação HSPC2 é formalizada. Ela introduz segurança aos esquemas de códigos de prefixo. HSPC2 obscurece a redundância através da aplicação da substituição homofônica (confusão) sobre a redundância suprimida (compressão de dados).

Seja um código de prefixo $C = (c_1, ..., c_n)$ para um dicionário $\Sigma = (\sigma_1, ..., \sigma_n)$ de n símbolos distintos, um texto original $T = t_1...t_m$ com $t_i \in \Sigma$, um vetor binário $K = (k_1, ..., k_n)$ como a chave secreta com $k_i \in \{0, 1\}$, a frequência f_i de cada um dos símbolos distintos em Σ, um número real $q \in (0, m]$ com $m = f_1 + ... + f_n$ e um vetor binário $B = (b_1, ..., b_n)$ com $b_i \in \{0, 1\}$. HSPC2 gera um texto cifrado S, tal que $S = H_K(T) = H_{k_{t_1}}(t_1)...H_{k_{t_m}}(t_m)$, onde H é uma transformação dada por,

$$H_i = \begin{cases} c_i & \text{se } k_i = 0 \\ c_i & \text{se } k_i = 1 \text{ a uma taxa } f_i - h_i \\ c_i.b_i & \text{se } k_i = 1 \text{ a uma taxa } h_i \end{cases}$$

onde $H_i \equiv H_{k_{t_i}}(t_i)$ e $c_i \equiv C(t_i)$.

Observe que C deve ser um código de prefixo consistente, isto é, o conjunto $c_1, ..., c_n$ deve ser um conjunto de prefixo. Desta foma, nenhuma palavra código c_i é um prefixo de outra c_j, onde $i \neq j$, e se $c_i = c_j$ então $i = j$.

Quando um símbolo tem um conjunto de chaves ($k_i = 1$), o número de instâncias desse símbolo que deve ser codificada com o homofônico $c_i.b_i$, é igual a h_i. Por exemplo, a tabela 9.3 define que o símbolo a tem uma ocorrência codificada como $00.b_i$ e uma ocorrência codificada como 00.

Se $k_i = 1$, HSPC2 usa c_i ou $c_i.b_i$ para codificar o símbolo σ_i a uma taxa h_i.

9.2.3 Escolhendo quais símbolos são codificados com o bit extra

O problema que surge com a decodificação é que quando é realizada a varredura no código c_i não se sabe se o próximo bit na cadeia codificada é b_i ou é o primeiro bit do próximo símbolo. Assim, deve ser utilizada alguma função geradora de índice $D_i(\alpha)$ para resolver esse problema de indexação. $D_i(\alpha)$ define onde o próximo α-ésimo homofônico irá aparecer, com $\alpha \in [1,\ h_i]$ e $D_i(\alpha) \in [1,\ f_i]$. Ela gera indices que são utilizados pelo HSPC2 durante o tempo de codificação na saída de códigos. Por exemplo, pelas tabelas 9.3 e 9.4 tem-se que o símbolo a é codificado usando o código 000 ou 001 apenas uma vez ($h_a = 1$), e que o símbolo b é codificado como 10 ou 11 duas vezes ($h_b = 2$). Então, os possíveis vetores de indices que marcam quais símbolos devem ser codificados com o bit extra para o primeiro símbolo são $\{(0,1),(1,0)\}$, e para o segundo são $\{(0,0,1,1),(0,1,0,1),(0,1,1,0),(1,0,1,0),(1,0,0,1),(1,1,0,0)\}$. Por exemplo, o vetor $(0,0,1,1)$ significa que o símbolo b tem que ser codificado com homofônicos pela terceira e quarta vez que ele aparece, as duas primeiras codificações devem usar o código c_i. Por exemplo, `111010`, `111011`, `111111` e `111110` são codificações válidas para o texto original `bbbb`.

Esses vetores são gerados por $D_i(\alpha)$ e são regerados em tempo de decodificação, desta forma esses vetores não precisam ser armazenados. Eles são usados apenas para símbolos com chave onde $k_i = 1$. $D_i(\alpha)$ pode ser implementada como uma função hash, uma função randomizada ou por alguma

outra regra determinística.

Note que os algoritmos têm que lidar com colisões. Exemplos de possíveis funções são $D_i(\alpha) = (x + D_i(\alpha - 1)) \bmod f_i$, $D_i(\alpha) = rand(x.D_i(\alpha - 1)) \bmod f_i$, ou $D_i(\alpha) = (2.\alpha) \bmod f_i$, para o símbolo i onde f_i é a frequência do símbolo i e x alguma entrada representado uma chave. Por exemplo, x pode ser a chave K concatenada, ou seja, x é o valor decimal do binário $k_1k_2k_3...k_n$.

Se a função $D_i(\alpha)$ for definida como $D_i(\alpha) = (x + D_i(\alpha - 1)) \bmod f_i$ com $D(-1) = 0$, a chave $K = (1, 1, 0)$ e assim $x = 110 = 6$, então o símbolo a tem indices $(1, 0)$ visto que $D(0) = 6 \bmod 2 = 0$ e o símbolo b tem índices $(1, 0, 1, 0)$ pois $D(0) = 6 \bmod 4 = 2$ e $D(1) = (6 + 2) \bmod 4 = 0$.

O sufixo binário b_i pode ser escolhido arbitrariamente. Algumas políticas para determinar b_i podem ser de forma aleatória, alterando 0´s e 1´s, isto é, $b_i = 0$ se i é par e $b_i = 1$ caso i seja ímpar, ou baseado em alguma regra estabelecida, por exemplo: $b_i = 0$ se $(i \bmod 10) = 0$, e 1 caso contrário.

9.2.4
A função de decodificação HSPC2

A função de decodificação HSPC2 utiliza o algoritmo de decodificação prefixo padrão mais o conhecimento da chave K e da função D para decodificar os bits extra inseridos no texto cifrado. Dado uma cadeia de bits no texto cifrado, a função de decodificação HSPC2 realiza os seguintes passos,

```
(1) Carregamento do dicionário e livro de códigos;
(2) Leitura da chave K;
(3) Geração do vetor de índices para cada símbolo
    usando a função D;
(4) Varredura em loop do texto cifrado:
    (4.1) Decodificação do próximo símbolo: varredura na cadeia
          de bits até encontrar um símbolo válido;
    (4.2) Atualização do contador de posição do vetor de índices
          do símbolo;
    (4.3) Se a chave e o índice estiverem estabelecidos então
          desconsidere o próximo bit (extra).
```

Em implementações práticas, a troca de chave K é realizada através de protocolos de chave de sessão ou chaves manuais pré-compartilhadas. Protocolos de chave de sessão usam chave pública e algoritmos simétricos: o servidor gera a chave de sessão K, criptografa a chave com a chave pública do cliente e a envia para o cliente, o cliente recebe a chave cifrada e a decifra com sua chave privada. No método de chaves pré-compartilhadas, a chave K é manualmente inserida pelo operador em ambos sistemas cliente e servidor.

9.3 Quebrando o Código

Agora que a função de codificação HSPC2 está definida, será realizada uma análise teórica a respeito da dificuldade em se decodificar o texto cifrado. Para demonstrar a segurança do algoritmo, esta seção inicia com a apresentação de um exemplo que ilustra o problema HSPC2 de quebra de código.

Exemplo 1: Suponha que seja dado $< \Sigma, F, R, S, q >$, *onde* $\Sigma = (a, b, c)$, $F = (2, 4, 3)$, $R = (2, 1, 2)$, $S =$ `11010010011110101`, $q = 3$ *e* $m = 9$. *A questão é a seguinte: existe um código de prefixo consistente* C, *um texto original* T *e uma chave* K *tal que* $H_K(T) = S$, *onde* $H_K(T)$ *é a função HSPC2? Assuma que* $D(i) = (x + D(i-1)) \ mod \ f$ *com* $D(-1) = 0$ *e* $b_i = 1$ *para todo* $i \in [1, n]$.

A resposta para essa questão é afirmativa, pois os itens necessários podem ser escolhidos da seguinte forma: $C = (00, 1, 01)$, $T = bcaabbbcc$ *e* $K = (1, 1, 0)$. *O ponto principal do desafio é encontrar a tripla* $< C, T, K >$ *que satisfaz os requisitos. O problema é realmente difícil ? Caso alguém tenha um texto cifrado* S *como esse, seria fácil obter o texto original* T *?*

Essas questões são analisadas nas próximas seções. Se for possível provar que este é um problema computacionalmente difícil (NP-Completo), então tem-se um algoritmo comprovadamente seguro para adicionar segurança aos códigos prefixo.

O problema HSPC2 é definido como o seguinte problema de decisão.

9.3.1 O problema de decisão HSPC2

Entrada:	dados $< \Sigma, F, R, S, q >$,
	um dicionário $\Sigma = (\sigma_1, ..., \sigma_n)$ de n símbolos,
	um vetor $F = (f_1, ..., f_n)$ de frequências,
	um vetor $R = (r_1, ..., r_n)$ de comprimentos de códigos,
	um texto cifrado S obtido por $S = H_K(T)$
	e $q \in (0, m]$ com $m = f_1 + ... + f_n$.
Pergunta:	existe $< C, T, K >$ com
	um código prefixo consistente C,
	um texto original T
	e uma chave K
	tal que $H_K(T) = S$,
	onde $H_K(T)$ é a função HSPC2 ?

Note que são fornecidos apenas os comprientos dos códigos. Na verdade, o vetor R é dado pois ele consiste numa informação intrínseca que pode ser obtida de Σ, F e do conhecimento do algoritmo de código prefixo adotado. Por exemplo, se HSPC2 usa Huffman então R não é secreto porque ele pode ser obtido pelo conhecimento de F. Uma propriedade interessante aqui é que Σ e F não são secretos, e sendo assim, não precisam ser protegidos. Entretanto, o código prefixo C não é completamente definido por R, devido a ambiguidade restante (28), então, escolhas feitas no processo de codificação (como a variação entre a codificação Huffman por ordem esquerda e direita) também contribuem para aumentar a segurança do texto cifrado.

9.3.2 Um problema mais simples: RHPC2

Para melhor entender o problema de decisão HSPC2 e realizar uma análise mais apropriada, considere agora um problema mais simples denotado por RHSPC2. RHSPC2 é obtido pela adição de duas simplificações sobre o HSPC2.

O texto original T é dado. O texto original T usado em RHSPC2 tem uma formatação definida pela concatenação de símbolos, isto é, $T = (\sigma_1^{f_1}.\sigma_1^{f_2}\sigma_1^{f_3}...\sigma_n^{f_n})$. Como exemplo, seja $T = aabbbbccc = a^2b^4c^3$. Como consequência, tem-se que a formatação de T e o vetor de frequências F definem T.

O livro de códigos C é dado. Pelo conhecimento do texto original, os comprimentos dos códigos e frequências, é possível construir um código prefixo consistente válido C. Logo, no RHSPC2, o livro de códigos é dado.

O problema RHSPC2 é definido como o seguinte problema de decisão.

Entrada:	dados $< \Sigma, F, R, L, q, C, T >$,
	um dicionário $\Sigma = (\sigma_1, ..., \sigma_n)$ com n símbolos,
	um vetor $F = (f_1, ..., f_n)$ de frequências,
	um vetor $R = (r_1, ..., r_n)$ de comprimentos de códigos,
	um inteiro L, com $L \geq f_1 r_1 + ... + f_n.r_n$
	$q \in (0, m]$ com $m = f_1 + ... + f_n$.
	um texto original $T = (\sigma_1^{f_1}.\sigma_1^{f_2}\sigma_1^{f_3}...\sigma_n^{f_n})$,
	e um livro de códigos $C = (c_1, ..., c_n)$,
Pergunta:	existe uma chave K tal que $\mid H_K(T) \mid = L$,
	onde $H_K(T)$ é a função HSPC2 ?

Aqui, apenas o tamanho de $H_K(T)$ é importante. Note que mais de um livro de códigos pode resultar no mesmo tamanho $H_K(T)$, ou seja, é possível existir dois livros códigos C_1 e C_2, com os mesmos comprimentos de código, ambos resultando em $\mid H_K(T) \mid = L$. Esta ambiguidade é destacada em (28). No problema RHSPC2 é dado um livro de códigos, porém outros podem também existir.

9.3.3 Avaliando a segurança do algoritmo

De acordo com os critérios de avaliação de segurança NIST usados no AES (24), a segurança do HSPC2 pode ser observada a partir das seguintes propriedades do algoritmo.

Segurança relativa a algoritmos similares

Este trabalho é similar ao (9), onde Klein et al. mostram que o problema da análise de códigos Huffman usados para codificar linguagens naturais grandes é NP-completo para diversas variações do processo de codificação. Em (9), Klein et al. analisam o esquema SPC onde eles utilizam a estratégia de ter um sufixo fixo de tamanho variável por símbolo no texto original. No trabalho deles, o tamanho e a presença (ou não) do sufixo são secretos. O tamanho da chave é igual ao número de símbolos no texto original.

Neste trabalho é empregada a substituição homofônica; desta forma a ocorrência de um bit sufixo é definida por uma chave e por um parâmetro taxa. Se a chave símbolo não é definida o símbolo não terá o bit sufixo, caso contrário o símbolo poderá ter ou não o bit sufixo e sua presença é definida pela taxa homofônica. O tamanho da chave é igual ao número de símbolos distintos no texto original.

Resistência à Criptoanálise

Nesta seção é demonstrado que a quebra da chave utilizada no texto cifrado $S = H_K(T)$ é um problema NP-Completo. É empregada uma redução a partir do SUBSET-SUM, um problema NP-Completo conhecido (4), para mostrar que o HSPC2 é um algoritmo comprovadamente seguro.

O problema SUBSET-SUM (SUS) é definido da seguinte forma,

Entrada:	um vetor $A = (\alpha_1, ..., \alpha_n)$, $\alpha_i \in \aleph$, e um objetivo g, $g \in \aleph$, com $g \leq (\alpha_1 + ... + \alpha_n)$.
Pergunta:	existe algum vetor binário $K = (k_1, ..., k_n)$ tal que $(\alpha_1.k_1 + ... + \alpha_n.k_n) = g$?

Teorema 9.1 (SUS $\propto$ RHSPC2)

Deseja-se provar que RHSPC2 é NP-Hard, isto é, existe um algoritmo de redução Λ de SUS a RHSPC2. As três condições seguintes devem acontecer.

(i) Λ *constroi uma instância de RHSPC2 a partir de SUS.* Suponha que o problema SUS seja definido por $A = (\alpha_1, ..., \alpha_n)$ e objetivo g. Seja a construção do problema RHSPC2 a partir do problema SUS: geração de um dicionário $\Sigma = (\sigma_1, ..., \sigma_n)$ com n símbolos. Escolha de um inteiro $q \in (0, m]$ com $m = f_1 + ... + f_n$. É definida a frequência para cada símbolo como $f_i = \lfloor \frac{m}{q}.\alpha_i \rfloor$. O texto original é definido como $T = (\sigma_1^{f_1}.\sigma_1^{f_2}\sigma_1^{f_3}...\sigma_n^{f_n})$. Geração de um livro de códigos pela aplicação do algoritmo de código prefixo definido por HSPC2. L é definido por $L =\mid S_0 \mid +g$, onde $S_0 = H_{K_0}(T)$ with $K_0 = (0, .., 0)$, e assim $L \in \aleph$.

(ii) Λ *é polinomial.* As tarefas de geração do dicionário, escolha de q, cálculo de f_i, definição de T e obtenção de um livro de códigos (usando códigos de Huffman (7), por exemplo) são todas operações polinomiais. Logo, a complexidade total é polinomial.

(iii) Λ *está correto.* Seja K um dado vetor binário. Deve-se mostrar que K *é uma solução para SUS definida por* $< A, g >$ *se e somente se* K *for uma solução para RHSPC2 definida por* $< \Sigma, F, R, L, q, C, T >$ *com* $L = (\mid S_0 \mid +g)$. Primeiro, suponha que K seja uma solução para RHSPC2, então tem-se $\mid H_K(T) \mid = L$. Para o cálculo de $\mid H_K(T) \mid$,

$$
\begin{aligned}
\mid H_K(T) \mid &= \sum_{i=1}^{n} r_i.f_i.(1-k_i) + \sum_{i=1}^{n}(r_i+1).h_i.k_i + \sum_{i=1}^{n} r_i.(f_i-h_i).k_i \\
&= \sum_{i=1}^{n} r_i.f_i + \sum_{i=1}^{n}\lceil\frac{q}{m}.f_i\rceil.k_i = \mid S_0 \mid + \sum_{i=1}^{n}\lceil\frac{q}{m}.f_i\rceil.k_i \\
&= \mid S_0 \mid + \sum_{i=1}^{n}\lceil\frac{q}{m}.\lfloor\frac{m}{q}.\alpha_i\rfloor\rceil.k_i = \mid S_0 \mid + \sum_{i=1}^{n}\lceil\frac{q}{m}.((\frac{m}{q}.\alpha_i)-\epsilon)\rceil.k_i \\
&= \mid S_0 \mid + \sum_{i=1}^{n}\lceil\alpha_i - \frac{\epsilon}{\frac{m}{q}}\rceil.k_i = \mid S_0 \mid + \sum_{i=1}^{n}\lceil\alpha_i-\delta\rceil.k_i \\
&= \mid S_0 \mid + \sum_{i=1}^{n}\alpha_i.k_i
\end{aligned}
$$

onde $\mid S_0 \mid = \sum_{i=1}^{n} r_i.f_i$ e $0 \leq \epsilon < 1$, $0 \leq \delta \leq \epsilon < 1$, $\frac{m}{q} \in \Re$, $\frac{m}{q} \geq 1$ e $\alpha_i \in \aleph$.

Logo, $\mid S_0 \mid + \sum_{i=1}^{n} \alpha_i.k_i = L = \mid S_0 \mid + g$, então $\sum_{i=1}^{n} \alpha_i.k_i = g$. Assim K é uma solução para o problema SUS.

Agora, suponha que o problema RHSPC2 seja definido por $< \Sigma, F, R, L, q, C, T >$. Então, o problema SUS correspondente é definido por $A = (\alpha_1, ..., \alpha_n)$ e objetivo $g = L - \mid S_0 \mid$, onde α_i é definido por $\alpha_i = h_i = \lceil\frac{q}{m}.f_i\rceil$. Se K é uma solução para SUS, então $(\alpha_1.k_1 + ... + \alpha_n.k_n) = g$. Logo, $\mid H_K(T) \mid = \mid S_0 \mid + (\alpha_1.k_1 + ... + \alpha_n.k_n) = \mid S_0 \mid + g = \mid S_0 \mid + L - \mid S_0 \mid = L$. Sendo assim, K é uma solução para o problema RHSPC2.

De (i), (ii) e (iii), tem-se a prova que SUS $\propto$ RHSPC2.

Para se demonstrar o próximo teorema, considere agora que todos os textos originais são representados por um esquema de codificação do tipo *run-length*. Por exemplo, o texto original $T =$ `abaabbccc` deve ser representado como $\widetilde{T} =$ `1a1b2a2b3c`. Esta modificação no HSPC2 não introduz qualquer perda de generalização e a operação de mudança de representação é polinomial. A representação *run-length* reduz o tamanho do texto resultante se todos os símbolos em T estiverem agrupados. Observe que esta representação *run-length* utilizada pelo HSPC2 pode descrever qualquer texto original, especialmente aquele utilizado pelo RHSPC2. Por outro lado, pode dobrar o tamanho do texto se cada símbolo for distribuído pelo texto original. Assume-se que a função D e a política do bit extra são fixas para o HSPC2.

Teorema 9.2 (RHSPC2 $\propto$ HSPC2)

O Teorema 9.1 prova que RHSPC2 é NP-Hard. Agora, deseja-se provar que HSPC2 é NP-Hard, isto é, existe um algoritmo de redução Λ de RHSPC2 para HSPC2. As três condições seguintes devem acontecer.

(i) Λ *constrói uma instância de HSPC2 a partir de RHSPC2.* Suponha que o problema RHSPC2 seja definido pelo dicionário $\Sigma = (\sigma_1, ..., \sigma_n)$ de n símbolos, o vetor $F = (f_1, ..., f_n)$ de frequências, um texto original $T = (\sigma_1^{f_1}.\sigma_1^{f_2}\sigma_1^{f_3}...\sigma_n^{f_n})$, o livro de códigos $C = (c_1, ..., c_n)$, com comprimentos de código $R = (r_1, ..., r_n)$, um inteiro L, com $L \geq f_1 r_1 + ... + f_n.r_n$, e $q \in (0, m]$, com $m = f_1 + ... + f_n$.

A construção do problema HSPC2 a partir do problema RHSPC2 é imediata: considere o mesmo dicionário, frequências, palavra código, comprimentos de código e q. Agora, gere o texto original $\widetilde{T}$ de HSPC2 segundo uma representação *run-length*: $\widetilde{T} = f_1\sigma_1 f_2\sigma_2...f_n\sigma_n$.

(ii) Λ *é polinomial.* HSPC2 usa todos os vetores de entrada de RHSPC2, exceto o texto original $\widetilde{T}$ que é facilmente derivado. Assim, a complexidade total é polinomial.

(iii) Λ *está correto.* Seja K o vetor binário dado. Deve-se mostrar que K *é uma solução de RHSPC2 definida por* $< \Sigma, F, R, L, q, C, T >$ *com* $L =\mid S \mid= \mid H_K(T) \mid$ *se e somente se* K, C *e* $\widetilde{T}$ *forem solução para HSPC2 definida por* $< \Sigma, F, R, q >$. $\widetilde{T}$ *é a representação run-length de* T. Primeiro, suponha que K seja uma solução para RHSPC2. Então, $L =\mid S \mid= \mid H_K(T) \mid$. Além disso, para a chave K, palavra código C e texto original T, tem-se que $S = H_K(T)$. Devido ao fato de $\widetilde{T}$ e T serem representações equivalentes quando os símbolos estão agrupados e $H_K(\widetilde{T}) = H_K(T)$. Assim, K, C e $\widetilde{T}$ são solução de HSPC2.

Agora, suponha que K, C e $\widetilde{T}$ sejam solução para HSPC2. Então, $S = H_K(T)$ e $L =\mid S \mid= \mid H_K(\widetilde{T}) \mid$. Além disso, note que símbolos em $\widetilde{T}$ não estão necessariamente agrupados, mas $\mid H_K(\widetilde{T}) \mid= \mid H_K(T) \mid= L$. Assim, não é preciso agrupar os símbolos em T para verificar que K é uma solução para RHSPC2.

De (i), (ii) and (iii), pode-se provar que HSPC2 é NP-Hard.

Observe que em HSPC2, o criptoanalista não sabe a formatação do texto, então é necessário advinhar as partições do texto cifrado S através de busca exaustiva consistindo da divisão da cadeia de bits S em m códigos não-vazios, onde $m = f_1 + ... + f_n$, é o número de símbolos em T. A análise força bruta do número de possíveis combinações diferentes foi primeiramente destacada por (8) e tem que considerar um número exponencial de partições em S, sob as seguintes restrições: o conjunto de diferentes palavras código na sequência deve ser não vazio e deve-se eliminar os conjuntos que não são códigos de prefixo consistentes.

Além disso, pode ocorrer o problema da ambiguidade na procura pelo livro de códigos quando é realizada a criptoanálise no HSPC2. Os códigos ambíguos aparecem quando é possível decodificar e codificar dados usando dois ou mais livros de códigos válidos, isto é, se uma cadeia de bits puder ser decodificada usando o livro de códigos C_1 ou C_2, resultando em textos originais igualmente válidos. Então, códigos de prefixo como os códigos Huffman levam a dados codificados ambíguos. Ambiguidade de código foi destacada por Rivest et al. (28), que criptoanalisaram dados codificados com Huffman assumindo que o criptoanalista não conhecia o livro de códigos. De acordo com o trabalho deles, a análise criptográica nessa situação é surpreendentemente difícil e até mesmo impossível em alguns casos devido à ambiguidade nos dados codificados resultantes.

Teorema 9.3 (HSPC2 é NP-Completo)

HSPC2 é NP-Completo se as duas seguintes condições forem verificadas: (i) HSPC2 estiver em NP; (ii) HSPC2 estiver em NP-Hard. Para provar (i), assume-se que é dado um certificado C, T e K. Então, a verificação é imediata caso seja verdadeiro ou não que $S = H_K(T)$. A condição (ii) é verificada pelo Teorema 9.1. Logo, HSPC2 é NP-Completo e HSPC2 é comprovadamente seguro.

9.4 Estimando a Expansão do Texto Cifrado

A inserção de bits na cadeia de saída dificulta o trabalho da criptoanálise, porém reduz a eficiência da compressão. É importante então estimar a expansão esperada no texto cifrado.

No RHSPC2, o texto original não é secreto, e sua formatação é dada por $T = (\sigma_1^{f_1}.\sigma_1^{f_2}\sigma_1^{f_3}...\sigma_n^{f_n})$. L é o inteiro definido por

$$L =\mid H_K(T) \mid=\mid S \mid=\mid S_0 \mid + \sum_{i=1}^{n} h_i.k_i$$

A expansão dos dados cifrados Ψ de S é definida como

$$\Psi = \frac{(\mid S \mid - \mid S_0 \mid)}{\mid S_0 \mid} = \frac{\sum_{i=1}^{n} h_i.f_i}{\sum_{i=1}^{n} r_i.f_i}$$

e como $\sum_{i=1}^{n} r_i.f_i \geq \sum_{i=1}^{n} f_i = m$, tem-se

$$\Psi \leq \frac{\sum_{i=1}^{n} \lceil \frac{q}{m}.f_i \rceil.k_i}{m} \leq \frac{\sum_{i=1}^{n} (1 + \frac{q}{m}.f_i).k_i}{m} \leq \frac{n+q}{m}$$

visto que $k_i \leq 1, \forall i = 1, ..., n$.

Observe que D é uma função crescente de q. Logo, q pode ser utilizado para controlar o overhead de bits extra devido a cifragem. A expansão de dados depende também dos vetores K e F. Para calcular a expansão de dados esperada são tomadas algumas considerações probabilísticas. Primeiro, assume-se que q, K e F são independentes. Isso é verdade para RHSPC2 visto que dado F, a chave K e o número q são escolhidos de forma aleatória e independente de F. Nesse caso, $E[q] = \frac{m}{2}$ and $E[k_i] = p[k_i = 1] = \frac{1}{2}$.

Para um texto original tem-se

$$\begin{aligned}
E[\Psi] &\leq E[\frac{\sum_{i=1}^{n}(1 + \frac{q}{m}.f_i).k_i}{m}] = E[\sum_{i=1}^{n} \frac{k_i}{m} + \sum_{i=1}^{n} \frac{\frac{q}{m}.f_i.k_i}{m}] \\
&= \frac{E[\sum_{i=1}^{n} k_i]}{m} + E[\frac{q}{m^2}.\sum_{i=1}^{n} f_i.k_i] = \frac{\frac{n}{2}}{m} + \frac{1}{m^2}.E[q].E[\sum_{i=1}^{n} k_i.f_i] \\
&= \frac{n}{2.m} + \frac{1}{m^2}.\frac{m}{2}.\sum_{i=1}^{n}(E[k_i].E[f_i]) = \frac{n}{2.m} + \frac{1}{2.m}.\sum_{i=1}^{n}(\frac{1}{2}.E[f_i]) \\
&= \frac{n}{2.m} + \frac{1}{4.m}.\sum_{i=1}^{n} E[f_i] = \frac{n}{2.m} + \frac{1}{4.m}.E[\sum_{i=1}^{n} f_i] \\
&= \frac{n}{2.m} + \frac{1}{4.m}.m \approx \frac{1}{4}
\end{aligned}$$

visto que $m \gg n$ para grandes textos.

Assim, a expansão esperada de dados cifrados $E[\Psi]$ é assintoticamente menor que $\frac{1}{4}$ bit por símbolo para varreduras usuais e considerações de codificação. Se os símbolos forem considerados t-grams a expansão de dados é assintoticamente limitada por $\widetilde{E}[\Psi] \leq \frac{1}{4.t}$ para códigos de varredura t-gram. Como exemplo, para uma varredura 5-gram, ou de maneira similar para uma varredura em palavra, o limite é 5%. Além disso, visto que Ψ depende de q, K e F, pode-se diminuir a expansão de dados através da adoção de políticas para a ecolha de q e K. Por exemplo, para $E[k_i] = \beta$ tem-se $E[\Psi] = \frac{\beta}{2}$ bits por símbolo, onde $\beta \in [0, 1]$. A desvantagem aqui é a falta de sigilo, visto que essa distribuição de probabilidade diferente para K pode ser empregada pelos criptoanalistas.

9.5 Conclusões

As questões envolvendo o uso de esquemas de compressão de dados vêm sendo examinadas por criptógrafos ao longo dos anos. É sabido que a compressão de dados não oferece segurança contra simples análises tais quais os ataques estatísticos. Neste capítulo foi proposto um algoritmo que aumenta a segurança do processo de codificação através da substituição homofônica com uma chave: o *HSPC2 - Códigos de Prefixo baseados em Substituição Homofônica com 2 homofônicos*.

Através dos resultados apresentados no capítulo, foi obtida um análise teórica das propriedades de segurança que são adicionadas ao código de compressão de dados prefixos modificados. Foram também apresentadas orientações simples para implementações práticas de algoritmos de criptocompressão de dados usando códigos Huffman Canônicos, distribuição diática e outras estratégias experimentais com a finalidade de aumentar a segurança dos textos cifrados contra ataques de criptoanalistas.

Uma das maiores vantagens da função HSPC2 é que as propriedades dos sistemas de recuperação de dados, tais como indexação e busca (23) permanecem inalteradas. O possível custo seria algum tipo de expansão de dados devido ao procedimento de criptografia, mas a análise em condições normais mostrou que, para varredura em palavra, a perda devido a compressão é assintoticamente menor que 5% por caracter.

Outras possíveis estratégias podem ser adicionadas ao esquema com a finalidade de diminuir a expansão de dados e aumentar o desempenho. Entretanto, o impacto teórico e prático na segurança devido as modificações no algoritmo deve ser analisado. Por exemplo, se a chave secreta K for dependente

to texto original T, tal qual $k_i = \frac{1}{f_i}$, então pode-se diminuir a expansão de dados, mas tem-se ainda que avaliar o impacto desta modificação nas propriedades de segurança do algoritmo. Ainda, outras técnicas correlatas podem ser empregadas para otimizar o algoritmo como um todo, tipo árvores esqueleto (10) e esquemas de redução de dicionário para lidar com textos em larga escala (35).