6
Inserção Seletiva de Nulos

6.1
Introdução

Neste capítulo será apresentado o algoritmo ADDNULLS - Inserção Seletiva de Nulos. Este algoritmo usa a técnica da esteganografia para esconder os símbolos codificados em símbolos falsos. É baseado na inserção seletiva de um número variável de símbolos nulos após os símbolos codificados. É mostrado que as perdas nas taxas de compressão são relativamente pequenas.

Este trabalho foi publicado no *SSI - Computer Security Symposium*, Brazil, 2000 e *SPIRE - String Processing and Information Retrieval*, Chile, 2001.

6.2
Huffman Criptografado

Nesta sessão são propostos dois criptosistemas que acrescentam propriedades criptográficas aos códigos Huffman.

Um criptosistema é definido por uma tupla (P, C, K, E, D), onde as seguintes condições são observadas:

1. P é o conjunto finito dos possíveis textos originais;

2. C é o conjunto finito dos possíveis textos cifrados;

3. K, o espaço de chaves, é o conjunto finito das possíveis chaves;

4. para cada $k \in K$, existe uma regra de criptografia $e_k \in E$ e uma correspondente regra de decriptografia $d_k \in D$. Cada $e_k : P \rightarrow C$ e $d_k : C \rightarrow P$ são funções tais que $d_k(e_k(x)) = x$ para todo texto original $X \in P$.

6.2.1 Substituição Homofônica

O primeiro procedimento é baseado na cifra de substituição homofônica. São inseridos símbolos "nulos" no texto cifrado. Um símbolo "nulo" não possui significado algum, e sua inclusão tem por finalidade evitar uma fácil decodificação do texto por uma pessoa não autorizada. Esse símbolo "nulo" η é inserido no texto cifrado com múltiplos códigos, também chamados de *homofônicos*.

Um conjunto de símbolos falsos $\Delta = \{\delta_1, \delta_2, \ldots, \delta_m\}$ é gerado para dissimular na saída o símbolo nulo. Considere neste trabalho que os símbolos falsos são os homofônicos do nulo. O método é descrito a seguir.

O procedimento de substituição homofônica é um criptosistema (P, C, K, L, F, E, D) onde são definidos, adicionalmente:

1. L como o conjunto finito dos possíveis códigos falsos representando o símbolo falso η. Desta forma, tem-se $L \subset C$ e $\Sigma^+ = \Sigma \cup \{\eta\}$, onde Σ^+ é o alfabeto de símbolos acrescido do símbolo nulo;

2. $F = (f_1, f_2, \ldots)$ como o gerador de códigos falsos. f_i é a função geradora de homofônicos do símbolo i. Se for utilizada uma mesma função f para todos os símbolos, então $f_i = f, \forall i$.

Construção do Codebook

O conjunto dos códigos falsos Δ são de fato códigos homofônicos de η. Esses homofônicos podem ser gerados de acordo com várias alternativas tais como:

Alternativa 1: Após coletar os textos do alfabeto $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ com suas respectivas frequências, criar outros m símbolos definidos como símbolos falsos δ_j com $j \in [1, m]$ que sejam homofônicos de η. Gerar frequências aleatórias para cada símbolo δ_j e depois construir a árvore de Huffman com Σ e $\{\delta_1, \delta_2, \ldots, \delta_m\}$ símbolos juntos na mesma árvore;

Alternativa 2: Criar m símbolos δ_j onde $j \in [1, m]$ com frequências iguais às dos primeiros m símbolos de Σ na árvore de Huffman. Depois, construir uma segunda árvore de Huffman falsa;

Alternativa 3: Após a construção da árvore de Huffman com N símbolos, assumir que os primeiros m símbolos representam também os m códigos falsos.

Em ambas alternativas 2 e 3 se faz necessário um bit identificador extra (ID bit) para indicar quando o símbolo código é real (bit 0) ou falso (bit 1).

Seja α a taxa de geração da árvore falsa. Esse parâmetro controla a taxa de expansão, por exemplo 30% de símbolos falsos na árvore codificada. Com essa taxa é possível configurar o número de símbolos falsos distintos criados $m = \alpha * N$, como ilustrado na figura 6.1 que representa a alternativa 2.

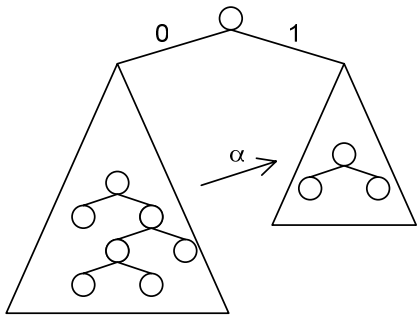

Figura 6.1: Geração de uma nova árvore de Huffman falsa a partir da taxa α.

Este trabalho adota a alternativa 3 em virtude da economia em espaço de memória e tempo de processamento menor do que seria necessário para a construção da árvore de Huffman com mais símbolos (alternativa 1) ou uma segunda árvore (alternativa 2).

Portanto, α representa a chave simétrica que define o número de símbolos falsos distintos usados conforme mostra a figura 6.1.

Codificação

Com os m códigos falsos definidos, uma função é utilizada para gerar a cadeia de códigos falsos para cada símbolo x_i de $X = x_1 x_2 \ldots$

Seja L_i a cadeia de homofônicos do nulo gerada para x_i e mais o próprio código x_i. Logo, $L_i = (\delta_1, \delta_2, \ldots, \delta_{mi}, x_i)$ como mostrado na figura 6.2. Para a codificação, basta então escolher um dentre os códigos falsos δ_i aleatoriamente, ou sequencialmente.

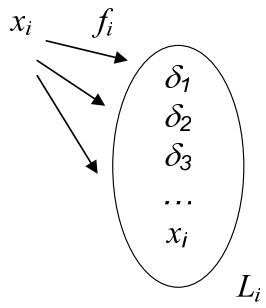

Figura 6.2: Geração dos homofônicos de η seguidos pelo símbolo efetivo.

Finalmente, seja β a taxa de geração de símbolos falsos, ou seja, a probabilidade de um código falso ser gerado a cada ciclo antes da saída do código efetivo.

O procedimento de codificação pode ser descrito pelo seguinte pseudo-código:

1. Escolha um número aleatório $p \in [0, 1]$
2. Se $p < \beta$ então
   $e_{Rs}(x_i) = \delta_i$
   retorne a 1
3. Senão
   $e_{Rs}(x_i) = x_i$
   $i \leftarrow i + 1$
4. Se $i \leq n$ então retorne a 1
5. Fim.

O parâmetro β é usado para estabelecer o número de símbolos falsos gerados entre as saídas de símbolos reais. Tem como finalidade balancear a difusão e a expansão de texto.

Desta forma, tem-se uma distribuição geométrica da geração dos m_i códigos falsos mais os símbolos efetivos definida pela seguinte expressão:

$$P[m_i + 1] = \beta^{(m_i+1)-1}.(1 - \beta) = \beta^{m_i}.(1 - \beta)$$

O procedimento de decodificação é definido abaixo:

1. Se a palavra-código lida é um símbolo S, então $d_k(s = x_i) = x_i$;
2. Senão $d_k(s = t_i) = \eta$ e o símbolo nulo é desprezado.

A geração dos símbolos falsos insere novos caracteres no texto codificado com a finalidade de uniformizar a distribuição total dos símbolos em uma dada linguagem.

Difusão

A análise estatística é um dos ataques de cifra mais conhecidos. Em qualquer que seja a linguagem sempre existirão caracteres mais utilizados que os demais. Desta forma, um ataque poderia ser feito pela contagem das frequências de cada símbolo no texto cifrado, e posterior correlação dos valores obtidos com a distribuição de frequência de cada caracter dentro da linguagem.

Com a geração de códigos falsos, é possível se obter uma certa difusão na distribuição de frequência dos códigos. Assim, a simples contagem das frequências dos códigos resultará em uma atribuição errada dos símbolos.

Uma desvantagem desse método é a expansão causada no texto. Entretanto, o método traz as seguintes vantagens: difusão na distribuição de frequências e ausência de correlação entre símbolos dentro da linguagem. Ambos atributos dificultam a análise estatística.

Expansão de texto

Para se estimar a expansão de texto causada pela substituição homofônica, observa-se que:

(i) O número médio de cifras geradas é a média da distribuição geométrica, ou seja, $1/(1 - \beta)$;

(ii) É necessário um bit adicional por caracter devido ao bit ID. Como $H(X) \leq l(\text{Huffman}) < H(X) + 1$ (11), tem-se agora no limite superior H(X)+2.

A expansão de texto pode ser estimada pelo número médio de caracteres gerados vezes o número médio de bits. Logo, o número médio de bits B por caracter é dado por:

$$B \leq [1/(1-\beta)].(H(X)+2)$$

Por exemplo, usando $\beta = 0.30$ e assumindo que $H(X) = 4.19$ para a varredura de caracter, e uma média de $H_L = 1.25$ para a entropia da língua inglesa, tem-se:

$1/(1-\beta) = 1/(1-0.30) = 1.43$, isto é, 43% de expansão de texto devido a códigos falsos.

$$B_l \leq 1.43\,(4.19+2) = 8.85$$

$$B_L \leq 1.43\,(1.25+2) = 4.65$$

Logo, é possível ainda atingir compressão usando varredura de palavra no Huffman criptografado quando comparado com a representação ASCII padrão (8 bits por caracter).

6.2.2 Cifra de Fluxo

Supondo que se tenha o texto codificado, a tabela de códigos Huffman e o número de códigos falsos definido por β, então, a decodificação é imediata.

Sendo assim, uma chave secreta se faz necessária para adicionar confusão ao processo. A chave secreta adotada nesse trabalho é completamente escalável, podendo ter qualquer tamanho desejado: 48 bits, 128 bits, 256 bits, etc.

Um segundo procedimento é então proposto: a cifra de fluxo.

Uma cifra de fluxo é um criptosistema (P, C, K, L, F, E, D) onde são adicionalmente definidos:

1. L como um conjunto finito chamado de alfabeto da cadeia de chaves;
2. $F = (f_1, f_2, \ldots)$ como o gerador da cadeia de chaves.

Cadeia de Chaves

Sejam $k \in K$ e $X = x_1x_2\ldots$ O procedimento da cifra de fluxo é definido a seguir:

(i) $Z = z_1z_2\ldots$ é a cadeia de chaves. Existe, então, uma função f_i que gera z_i a partir de k : $z_i = f_i(k, x_1, x_2, \ldots, x_{i-1})$;

(ii) z_i é usado para cifrar x_i tal que $y_i = e_{zi}(x_i)$.

Tem-se uma nova chave z_i para cada x_i, e essa chave z_i é gerada a partir dos últimos $z_1, y_1, z_2, y_2, \ldots, z_{i-1}, y_{i-1}$.

O método proposto consiste de uma única função constante tal que $z_i = f(k, i)$.

Se $|k| < |X|$, isto é, o tamanho da chave é menor que o tamanho do texto original, existem duas alternativas para se gerar a cadeia de chaves:

Alternativa 1: *Geração cíclica da cadeia de chaves* - quando a chave k termina o procedimento retorna ao início e assim por diante. Assim, o bit da chave é definido por $z_i = f(k, i) = k_{i \, mod \, \phi}$ onde $\phi = |k|$;

Alternativa 2: *Geração randômica da cadeia de chaves* - a função f gera as cadeias de bits usando k como semente.

A alternativa 1 é escolhida e ilustrada conforme a figura 6.3. Desta forma, as propriedades de sincronismo da codificação Hoffman são preservadas. A alternativa 2 não é empregada visto que ela causa dependência com o passado, e queremos manter a capacidade de indexação e busca.

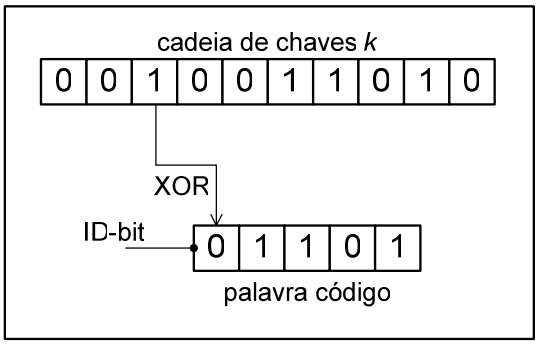

Figura 6.3: XOR entre o i-ésimo bit da chave k e o ID bit da palavra código.

Codificação

Sejam os procedimentos de codificação e decodificação definidos por $\otimes$ - operações XOR (ou-exclusivo) entre o bit z_i e todos os demais bits da palavra código. Isso equivale a trocar as posições de um símbolo na árvore de Huffman:

$$\begin{aligned} Y_i &= e_k(X_i) = (z_i \otimes x_{i,1}, z_i \otimes x_{i,2}, \ldots, z_i \otimes x_{i,|Xi|}) \\ X_i &= d_k(Y_i) = (z_i \otimes y_{i,1}, z_i \otimes y_{i,2}, \ldots, z_i \otimes y_{i,|Yi|}) \end{aligned}$$

Onde, $X_i = x_{i,1}x_{i,2}\ldots x_{i,|Xi|}$ e $Y_i = y_{i,1}y_{i,2}\ldots y_{i,|Yi|}$

Entretanto, um procedimento mais simples pode ser definido por uma operação de XOR entre o z_i bit e o primeiro bit da palavra código, o que é equivalente a disfarçar apenas o bit ID:

$$\boxed{\begin{array}{l} Y_i = e_k(X_i) = (z_i \otimes h_1, x_{i,2}, \ldots, x_{i,|Xi|}) \\ X_i = d_k(Y_i) = (z_i \otimes h'_1, y_{i,2}, \ldots, y_{i,|yi|}) \end{array}}$$

Onde, h_1 e h'_1 são os bits ID de x_i e y_i.

Confusão

Com a operação de XOR é possível ocultar o método proposto. Ela é simples e tem baixo custo de processamento. A chave secreta pode ter tamanho variado e qualquer ataque baseado em tentativa e erro seria tão custoso que faria com que a análise fosse inviável.

Neste esquema, tem-se de fato uma chave composta que contém α e a semente da cadeia de chaves k como mostra a figura 6.4.

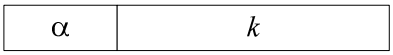

Figura 6.4: Chave composta.

Com a chave secreta incluída no processo, o sistema criptográfico pode ser aplicado em canais de comunicação inseguros.

Expansão de texto

A cifra de fluxo não causa qualquer expansão adicional de texto.

6.3 Experimentos

A tabela 6.1 apresenta alguns resultados decorrentes da implementação em linguagem C++ da codificação e decodificação do Huffman criptografado. Foram usadas a Constituição Brasileira e a coleção do Projeto Gutenberg (6).

Foi utilizado um $\alpha = 0.3$, varredura de caracter e todos os tamanhos foram expressos em bytes.

Primeiramente é medido o espaço de armazenamento necessário para comprimir a coleção de documentos usando o código Huffman padrão.

Depois o código de Huffman criptografado é utilizado e a medida da diferença entre o novo espaço requerido para as funções de criptografia é obtida, isto é, o espaço-extra necessário.

Observa-se que a expansão de texto está bem próxima do seu valor esperado. Além disso, o tempo adicional relativo para inserir a criptografia ao processo de compressão é de aproximadamente 5%.

Tabela 6.1: Resultados da expansão de texto(tamanhos em bytes).

Coleção	Tamanho texto	Tamanho texto cod.	Tamanho texto cifr.	Expansão texto	Expansão texto cod.
Constituição Brasileira	7.004.160	4.078.717	8.309.822	19%	104%
Projeto Gutenberg	39.059.456	22.813.331	52.539.412	34%	130%

6.4 Conclusões

Geralmente é necessário realizar procedimentos em série para comprimir e depois criptografar um texto. Nesse capítulo foi proposta uma simples modificação no código de Huffman no sentido de se adicionar criptografia aos procedimentos usuais de compressão.

O resultado foi a rapidez na descompressão e menor taxa de expansão. Observou-se também confusão e difusão no texto codificado, conforme análise teórica e prática apresentada nesse trabalho.

A difusão e confusão são também parâmetros controlados. O fator β controla a taxa de geração de códigos falsos. Com β é possível estabelecer a expansão do texto cifrado devido à inserção de símbolos nulos, o que está diretamente associada à segurança do arquivo. Enquanto que o tamanho da chave k está associado à segurança do processo de confusão.