Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Mestrado em Sistemas e Computação

Dissertação

PAPÍLIO: Proposta de um Algoritmo de Criptografia Baseado no Algoritmo Viterbi e Codificação Convolucional

Aluna: Karla Darlene Nepomuceno Ramos

Orientador: Ivan Saraiva da Silva

Co-Orientador: Benjamín René Callejas Bedregal

Natal-RN

Fevereiro de 2002.

PAPÍLIO: Proposta de um Algoritmo de Criptografia Baseado no Algoritmo de Viterbi e Codificação Convolucional

KARLA DARLENE NEPOMUCENO RAMOS

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte, como parte integrante dos requisitos necessários para obtenção do grau de mestre em sistemas e computação.

Aprovado por:

_____

Prof. Ivan Saraiva da Silva, D.Sc.
Presidente

_____

Prof. Benjamin René Callejas Bedregal, D.Sc.

_____

Prof. Ricardo Augusto da Luz Reis, D.Sc.

Natal-RN, fevereiro de 2002

Dedico a vitória da conclusão deste trabalho a minha mãe, Maria da Glória Nepomuceno Ramos, a quem sou eternamente grata por ter me dado a base necessária para mais uma realização; e ao meu esposo, Cleodon Martinho de Carvalho, pelo seu apoio, incentivo, compreensão e amor que me fortaleceram e conduziram-me para mais uma conquista de minha vida.

Agradecimentos

A Deus pela saúde, perseverança e iluminação durante a jornada para conclusão desta dissertação.

Ao meu orientador, Prof. Dr. Ivan Saraiva da Silva, por ter me ajudado a concluir uma etapa fundamental da minha vida.

Ao meu co-orientador, Prof. Dr. Benjamín René Callejas Bedregal, que muito contribuiu com seu conhecimento, interesse e dedicação para a definição e conclusão desta dissertação.

Ao Prof. Jossérgio Soares Antas de Gouveia por sua valiosa contribuição na implementação do algoritmo proposto nesta dissertação.

A minha inesquecível Profª Claudia Araújo Ribeiro por ter me motivado a participar do programa de mestrado da UFRN, e que mesmo de longe me acompanhou e incentivou.

A todos os professores do programa de mestrado de Sistemas e Computação da UFRN que, de uma forma ou de outra, contribuíram para o desfecho dessa dissertação.

A minha família pelo apoio e incentivo a consecução dos meus objetivos, em especial ao meu sogro, Francisco Pereira de Carvalho, e minha sogra, Francisca Salene Macedo de Carvalho, que me acolheram tão maravilhosamente bem, durante dois anos de estudo e pesquisa dedicados a esta dissertação.

A todos os amigos, em especial a Raimundo Nonato Camelo Parente, pela colaboração e apoio na caminhada desta pesquisa.

Sumário

Índice de Figuras

Índice de Tabelas

Resumo da dissertação apresentada como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M. Sc.)

PAPÍLIO: Proposta de um Algoritmo de Criptografia Baseado no Algoritmo de Viterbi e Codificação Convolucional

Karla Darlene Nepomuceno Ramos

Fevereiro de 2002

Orientador: Ivan Saraiva da Silva

Co-Orientador: Benjamín René Callejas Bedregal

Programa: Mestrado em Sistemas e Computação/DIMAP/UFRN

Linha de Pesquisa: Concepção de Sistemas Digitais

Palavras Chaves: criptografia, algoritmo Viterbi, codificação convolucional, codificador Feistel, algoritmo simétrico, efeito avalanche, difusão, confusão, codificador de bloco.

A criptografia é uma técnica básica no mundo digital que, apesar de essencial, não resolve por si só o problema da segurança. Existem vários algoritmos criptográficos usados para proteger e-mail privado, arquivos de computadores pessoais, transações bancárias eletrônicas, garantir a autenticidade de dados e mensagens, entre outras aplicações. Esta dissertação apresenta uma proposta de algoritmo de criptografia denominado PAPÍLIO, cujo processo de codificação baseia-se no algoritmo Viterbi. O algoritmo Viterbi foi inicialmente proposto como uma solução para a decodificação convolucional, desde então pesquisadores têm encontrado outras áreas de aplicação para o referido algoritmo. Para o algoritmo Viterbi ser utilizado na criptografia, foram necessárias algumas modificações que deram origem ao algoritmo Viterbi Modificado. PAPÍLIO é um codificador de bloco (criptografa dados em blocos de 8 bytes) simétrico (usa a mesma chave secreta para ambos criptação e decriptação) que usa uma chave de 128 bits e tem por objetivo a codificação de dados. Diferentemente de outros algoritmos de criptografia que utilizam uma matemática avançada ou operações com matrizes de

dados indexados (caixas-S), o algoritmo PAPÍLIO utiliza tabelas que além de serem geradas de forma simples, podem ser livremente divulgadas. A utilização do algoritmo Viterbi Modificado confere ao PAPÍLIO as propriedades conhecidas como efeito avalanche, difusão e confusão, desejadas em qualquer algoritmo de criptografia.

Abstract of the dissertation presented as partial fulfillment of the requirements for obtaining the Master's Degree in Sciences (M. Sc.)

PAPÍLIO: Proposal of Cryptography Algorithm Based in the Viterbi Algorithm and Convolutional Coding

Karla Darlene Nepomuceno Ramos

February, 2002

Advisor: Ivan Saraiva da Silva

Co-Advisor: Benjamín René Callejas Bedregal

Program: Graduation on Systems and Computation/DIMAP/UFRN

Research Line: Conception of Digital Systems

Keywords: cryptography, Viterbi algorithm, convolutional coding, Feistel Cipher, encryption, decryption, diffusion, confusion, avalanche effect, block cipher.

This M.Sc. thesis presents a conventional encryption algorithm called PAPÍLIO (Prepona, in english). PAPÍLIO is an symmetric (i.e. uses the same secret key for both encryption and decryption) block cipher encryption algorithm. PAPÍLIO encrypts 64-bit blocks of plaintext into 64-bit blocks of ciphertext. The key lenght is 128-bit. The coding's process is based in the Viterbi algorithm. The Viterbi algorithm was first proposed as a solution to the decoding of convolutional codes, since then others researchers have expanded for others areas. To use Viterbi algorithm in the cryptography, some modifications were realized and the Modified Viterbi algorithm was created. Unlike most cryptography algorithms that use advanced mathematics and lookup table in S-boxes, PAPÍLIO uses tables created by the Modified Viterbi and it has the desirables properties of any encryption algorithm: diffusion, confusion and avalanche effect.

Capítulo 1

Introdução

Há muitos anos a humanidade utiliza códigos para se comunicar. Entretanto, quando o objetivo é manter segredo, o código utilizado deve ser ininteligível a terceiros. A técnica de escrever de forma ininteligível, ou seja, em cifras ou códigos chama-se criptografia. O primeiro uso documentado da criptografia foi em torno de 1900 a.C., no Egito, quando um escriba usou hieróglifos fora do padrão em uma inscrição. Uma técnica clássica de criptografia é o codificador de Júlio César de 50-60 a.C. [Stallings 98]. Ao longo da história, muitas outras técnicas foram criadas com a finalidade de manter a privacidade.

A criptografia exerce um papel importante na privacidade de informações eletrônicas contra diversos tipos de ameaças. A preocupação com a privacidade das informações, está cada vez mais em evidência. O crescimento dos sistemas computacionais, e a interconexões via rede têm aumentado a dependência dos usuários em relação aos dados armazenados nesses sistemas. Outro aspecto que gera preocupação, além do armazenamento dos dados, é a comunicação (transmissão) desses dados no sistema. Existem serviços, tais como autenticação, integridade, dentre outros, que objetivam proteger os dados armazenados e transmitidos. Tais serviços utilizam mecanismos de segurança para atingirem seus objetivos. Um dos mecanismos mais utilizados para fornecer segurança é a criptografia.

Um algoritmo criptográfico constitui-se de uma função matemática bijetiva que codifica e decodifica uma mensagem. Para uma maior segurança os algoritmos codificadores utilizam uma chave que pode assumir diferentes valores.

Esta dissertação apresenta um algoritmo de criptografia convencional ou de chave privada denominado *PAPÍLIO*, o qual foi baseado no codificador Feistel, no algoritmo de Viterbi e na Codificação Convolucional. O algoritmo de Viterbi deu

origem ao Decodificador Viterbi, que junto com a Codificação Convolucional constituem uma técnica de Correção Antecipada de Erro.

Inspirando-se na treliça gerada pela Decodificação Viterbi, que sugere formas de borboletas, o algoritmo proposto foi denominado *PAPÍLIO*. *PAPÍLIO thoas brasiliensis* é o nome da espécie de borboletas muito comum no Brasil [Lasertogo 97] e em todo o estado do Rio Grande do Norte.

Esta dissertação será apresentada em seis capítulos. O capítulo 2 apresentará as classes da criptografia, os principais algoritmos de cada classe, os principais tipos de ataque aos sistemas criptográficos, bem como os modos de operação dos codificadores de bloco. A seção 2.1 trará informações sobre criptografia convencional ou simétrica. A seção 2.2 comentará sobre a criptografia assimétrica ou de chave pública. A seção 2.3 comentará sobre o sistema criptográfico híbrido e a seção 2.4. apresentará os principais tipos de ataques aos sistemas de criptografia. Os modos de operação dos codificadores em bloco serão comentados na seção 2.5.

O capítulo 3 apresentará as técnicas que compõem o sistema proposto. Esse capítulo estará subdividido em duas seções. A seção 3.1 tratará da técnica de Codificação Convolucional, enquanto que a seção 3.2 apresentará os pontos relevantes da Decodificação Viterbi, no tocante à área em que serão utilizados.

O capítulo 4 apresentará a motivação para utilização do algoritmo de Viterbi em criptografia, e as contribuições inovadoras desta dissertação. A primeira delas é a geração do algoritmo Viterbi Modificado, o qual gera tabelas de codificação de dados. A segunda é o desenvolvimento do algoritmo de criptografia *PAPÍLIO* baseado em tabelas de codificação geradas a partir do algoritmo Viterbi Modificado.

No capítulo 5, os resultados obtidos serão apresentados, bem como a resistência a ataques e a implementação do algoritmo *PAPÍLIO*. A conclusão da dissertação e as propostas de trabalhos futuros farão parte do capítulo 6.

Capítulo 2

Classes da Criptografia e Principais Algoritmos

A criptografia é uma das ferramentas mais utilizadas para fornecer segurança. A idéia é permitir ao usuário o uso privado de um conhecimento que torna a informação secreta, evitando que pessoas indesejáveis conheçam a informação.

Segundo Bruce Schneier [Schneier00], a criptografia é um punhado de acrônimos que realiza várias tarefas de segurança. IP Security (IPSec), por exemplo, protege o tráfego de IP pela internet, Secure Sockets Layer (SSL) protege as conexões da WWW, Pretty Good Privacy (PGP) e S/MIME protegem o e-mail de forma a impedir que terceiros leiam o e-mail, bem como forjem um e-mail para que pareça ter vindo de mais alguém. Todos esses são protocolos. Para se montar esses protocolos, os criptógrafos usam diferentes algoritmos: algoritmos de codificação, algoritmos de assinatura digital, etc.

Um algoritmo criptográfico é uma função matemática usada para codificar e decodificar informações. Em geral, os algoritmos de criptografia modernos – para tornar a codificação realmente segura – utilizam uma chave, *K*. A chave criptográfica foi inventada pelo italiano Leon Battista Alberti em 1466 [Schneier00].

A criptografia moderna está dividida em duas classes: criptografia convencional ou simétrica, e criptografia assimétrica ou de chave pública.

2.1 Criptografia Convencional

A criptografia convencional ou simétrica recebe esta denominação devido ao emissor e receptor compartilharem a mesma chave de codificação/decodificação de dados. Os algoritmos simétricos são utilizados mundialmente, os mais comuns são Data Encryption Standard – DES [NBS 77] e DES triplo, RC5 [Rivest 94], International Data Encryption – IDEA [Lai 90] e Blowfish [Scheneir 93].

A figura-2.1 exibe o processo de criptografia convencional. A mensagem original ou texto claro é convertido em um texto codificado através de um processo de codificação. Este processo de codificação consiste de um algoritmo e uma chave compartilhada pelo emissor e receptor.

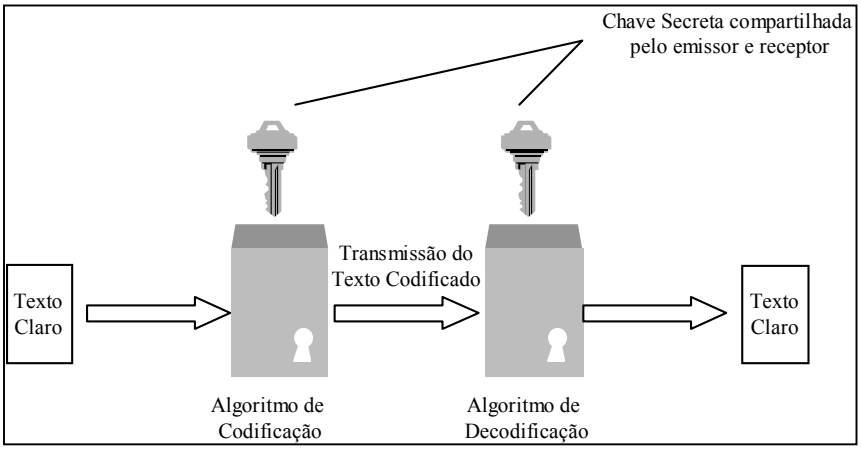

Figura-2.1: Processo de Criptografia Convencional

As notações abaixo representam as funções de codificação e decodificação, onde E representa o algoritmo de criptografia, D o algoritmo de decriptação, P o texto claro, C o texto codificado e K a chave.

(1) $$E_K (P) = C$$

(2) $$D_K (C) = P \text{ e}$$

(3) $$D_K (E_K (P)) = P$$

A notação (1) indica que C é o resultado do algoritmo de criptografia E dado o texto claro (P) e a chave K. A notação (2) representa o processo de reversão, cujo algoritmo de decriptação D atua em C e gera P, através da utilização de uma chave. Para recuperação do texto claro a identidade (3) deve ser verdadeira.

A segurança de um algoritmo não deve estar baseada no segredo do algoritmo. Se um algoritmo só for seguro se permanecer secreto, então ele só será seguro até que

alguém utilize engenharia reversa e o publique [Schneier 00]. Os algoritmos públicos são projetados para serem seguros, por isto não há riscos em torná-los públicos. A segurança da criptografia convencional está na chave. Um intruso conhecendo os algoritmos E e D e de posse do texto codificado, mas não tendo acesso à chave ou ao texto claro, poderá tentar descobrir o texto claro ou a chave, ou ambos. Raramente um intruso está interessado em descobrir apenas uma mensagem, em geral o intruso está interessado em descobrir a chave, pois conhecendo a chave ele poderá recuperar mensagens futuras. O processo de tentativas para descobrir o texto claro ou a chave, ou ambos, é conhecido como criptoanálise. A seção 2.4, deste capítulo, apresenta alguns tipos de ataques criptográficos.

Quanto à forma de processamento do texto claro, os algoritmos simétricos são classificados em duas categorias:

- codificadores de bloco: processa um bloco de elementos a cada momento, produzindo um bloco de saída para cada bloco de entrada.
- codificadores de fluxo: processa continuamente a entrada de elementos, produzindo ao longo da saída um elemento de cada vez.

Virtualmente, todos os modernos codificadores simétricos de bloco são semelhantes, em muitos aspectos, ao DES e a estrutura do codificador de bloco Feistel. O processo de codificação desses algoritmos está baseado em ferramentas de substituição e permutação.

Codificador Feistel

Horst Feistel, em 1973, propôs [Feistel73] um codificador de blocos que alterna funções de substituição e permutação. A estrutura do codificador de Feistel baseia-se na proposta de Shannon [Shannon 49] e constitui-se a base dos modernos codificadores simétricos ou de chave privada. Shannon usou pela primeira vez os termos *difusão* e *confusão*. O termo difusão foi utilizado com o objetivo de tornar a relação estatística entre texto claro (original) e texto codificado a mais complexa possível, a fim de evitar a descoberta da chave. Assim como a *difusão*, o termo *confusão* foi empregado para evitar a descoberta da chave de codificação, porém a confusão busca tornar o mais

complexo possível a relação estatística entre o texto codificado e a chave de codificação.

No codificador Feistel, a confusão é obtida através da função F. A grande maioria dos codificadores simétricos, baseados em Feistel, utiliza caixas-S para atingir o elemento de confusão. O objetivo das caixas-S é provocar uma mudança aleatória no vetor de saída da caixa-S, caso haja qualquer tipo de modificação no vetor de entrada. Em outras palavras, uma caixa-S consiste de uma substituição de *m* bits de entrada em *n* bits de saída. Em geral, a caixa-S representa uma função não linear de um algoritmo de criptografia e é responsável pela segurança do mesmo.

Vários outros critérios podem ser considerados na concepção da função F. Heys e Tavares [Heys 95] analisaram e propuseram um critério relacionado às caixas-S definido por Avalanche Garantida. Para definir o critério avalanche os autores consideraram: 1. a mudança de um bit ou diferenças OU-exclusivo dentro da rede quando dois textos claros, P' e P", são selecionados como entrada, tal que *wt* ($\Delta P = P' \oplus P''$) = 1, onde $\Delta P = P' \oplus P''$ representa o OU-exclusivo bit-a-bit de P' e P" e *wt* (.) representa o peso Hamming do vetor especificado, isto é, a quantidade de 1's do vetor; 2. que o texto codificado resultante de P' e P" é representado por C' e C", respectivamente; e que o número de bits trocados no texto codificado é representado por *wt* (ΔC), onde $\Delta C = C' \oplus C''$. A seguir está a definição formal, dada pelos autores, do critério avalanche:

> *Definição*$_1$*:* Um codificador satisfaz ao *critério avalanche* se, para cada chave, em média metade dos bits do texto codificado muda quando um bit do texto claro é trocado. Isto é, $E(wt(\Delta C) \mid wt(\Delta P)=1) = N/2$.

Considerando que a definição dos autores [Heys 95] só contempla modificação de um bit no texto claro, e como o efeito avalanche existe tanto se modificando um bit no texto claro ou na chave, gerou-se uma nova definição:

> *Definição*$_2$*:* Um codificador satisfaz ao critério avalanche se, para cada chave, em média metade dos bits do texto codificada muda quando um bit do texto claro ou da chave (K) é trocado. Isto é, $E(wt(\Delta C) \mid wt(\Delta P) + wt(\Delta K) =1) = N/2$.

A figura-2.2 exibe a estrutura de um codificador Feistel. O codificador Feistel tem como entrada um bloco de 2*n* bits de texto claro e uma chave. O bloco de texto

claro é dividido em duas partes de *n* bits (E_0 e D_0). As duas partes sofrem um processo de *v* voltas e no final são concatenadas gerando um bloco codificado de 2n bits. Cada volta i tem como entrada E_{i-1} e D_{i-1}, obtidas na volta anterior, além da subchave Ki, obtida da chave inicial. Em geral, as subchaves Ki são diferentes entre si, bem como da chave inicial. Todas as voltas têm a mesma estrutura. No lado esquerdo do dado é executada uma substituição. A

substituição é realizada através de uma função F aplicada ao lado direito do dado (D_{i-1}), fazendo em seguida um OU-exclusivo da saída da função com o dado do lado esquerdo (E_{i-1}). A função tem a mesma estrutura para cada volta, porém é parametrizada pela subchave Ki. Após a substituição, uma permutação é executada. A permutação consiste da troca das duas metades dos dados. Esta estrutura é uma forma particular da rede de substituição-permutação proposta por Shannon [Stallings98].

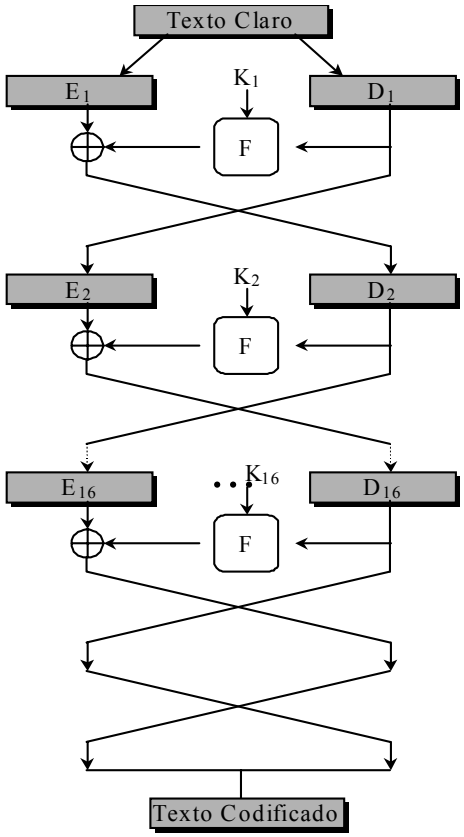

Figura-2.2: Estrutura do Codificador Feistel

A função F, aplicada ao lado direito do dado, constitui-se o coração do codificador Feistel, uma vez que ela fornece os elementos de confusão. A função F não precisa ser reversível. Independente de como seja constituída essa função, a estrutura do codificador permite que o mesmo algoritmo da codificação seja utilizado para

decriptação. Para o algoritmo de codificação ser utilizado para decodificação, é preciso que as subchaves geradas sejam usadas em ordem reversa, ou seja, a primeira volta da decodificação utiliza a última subchave de codificação, e assim por diante, até que o dado seja recuperado.

2.1.1. Principais Algoritmos Simétricos

DATA ENCRYPTION STANDARD (DES)

O DES foi desenvolvido pela IBM e adotado, em 1977, pelo *National Bureau of Standards* (NBS), atual *National Institute of Standards and Technology* (NIST), como o Padrão de Processamento de Informação Federal 46 (FIPS PUB 46). O DES utiliza uma chave de 56 bits para codificar blocos de texto claro de 64 bits. O algoritmo transforma, através de uma série de passos, uma entrada de 64 bits de texto claro em uma saída de 64 bits de texto codificado. A reversão do processo é realizada através dos mesmos passos e da mesma chave [Stallings98].

O tempo de vida útil do DES está chegando ao fim, em 1997 o NIST anunciou sua intenção em substituir o DES, e deu início ao processo de escolha do novo algoritmo de criptografia que será adotado como padrão [NIST-1]. O comprimento da chave do DES não se mostra adequada para os dias atuais. As principais características do DES são:

- tamanho do bloco: 64 bits;
- comprimento da chave: 56 bits;
- codificação do texto claro:

  O processamento do texto claro é realizado em três etapas. Inicialmente, o bloco de 64 bits passa por uma função de permutação inicial (IP), gerando uma saída permutada. Em seguida passa por um processo de 16 voltas que consiste de funções de permutação e substituição, envolvendo texto claro e chave. Por último, a saída do processo de 16 voltas passa através de uma permutação (IP^{-1}) que é o inverso da função de permutação inicial.
- quantidade de voltas de codificação: 16.

Como mencionado anteriormente, o tamanho da chave do DES torna-o inadequado para muitas aplicações. Entretanto, variantes tais como DES Triplo e DESX permitem a continuidade de uso do codificador.

INTERNATIONAL DATA ENCRYPTION (IDEA)

O IDEA é um codificador de blocos simétrico desenvolvido por Xuejia Lai e James Massey do *Swiss Federal Institute of Technology* [LAI 90][LAI 91], [LAI 92]. O algoritmo utiliza uma chave de 128 bits para codificar blocos de texto claro de 64 bits, e é de fácil implementação tanto em hardware como em software [Stallings 98].

As principais características do IDEA são:

- tamanho do bloco: 64 bits;
- comprimento da chave: 128 bits;
- codificação do texto claro:

  O processamento do texto claro envolve oito voltas e uma função de transformação final. O algoritmo divide a entrada em quatro sub-blocos de 16 bits. A cada volta, quatro sub-blocos de 16 bits são tomados como entrada, e produzem quatro sub-blocos de saída de 16 bits. A transformação final também produz quatro blocos de 16 bits, que são concatenados formando um texto codificado de 64 bits. Além dos sub-blocos, cada volta utiliza dezesseis subchaves, enquanto a transformação final utiliza quatro subchaves. Durante o processo de codificação 52 subchaves são utilizadas, e todas elas são geradas a partir da chave inicial de 128 bits. A cada volta três operações são utilizadas: 1. OU-exclusivo bit-a-bit; 2. Adição de inteiros módulo 2^{16}, com entradas e saídas tratadas como inteiros de 16 bits sem sinal; e 3. Multiplicação de inteiros módulo $2^{16} + 1$, com entradas e saídas tratadas como inteiros de 16 bits sem sinal, exceto o bloco constituído por zeros que é tratado como representação 2^{16}.
- quantidade de voltas de codificação: 8.

RC5

O algoritmo RC5 (Ron's Code) foi desenvolvido por Ron Rivest [Rivest 94, Rivest 95]. O processo de codificação depende de três parâmetros: tamanho da palavra, número de voltas e número de bytes da chave secreta. O RC5 codifica blocos de texto claro de 32, 64 ou 128 bits em blocos de texto codificado de mesmo tamanho. O comprimento da chave vai de 0 a 2040 bits. A codificação do RC5 é realizada através de três operações: 1. adição de palavras módulo 2^w, onde *w* representa o tamanho da palavra (16, 32 e 64 bits); 2. OU-exclusivo bit-a-bit; e 3. Deslocamento circular a esquerda da palavra por *y* bits.

O RC5 apresenta as seguintes características de concepção [Stallings 98]:

- utiliza operações computacionais primitivas presentes na maioria dos microprocessadores, o que o torna disponível tanto para hardware como para software;
- devido ao parâmetro tamanho da palavra, adapta-se a processadores com diferentes tamanhos de palavra;
- os parâmetros número de voltas e comprimento da chave permitem uma relação entre maior velocidade e maior segurança;
- necessita de pouca memória, podendo ser implementado em dispositivos de memória restrita;
- o deslocamento circular a esquerda torna os dados dependentes, tornando o algoritmo mais resistente a criptoanálise.

BLOWFISH

O blowfish é um codificador simétrico que foi desenvolvido por Bruce Schneier em 1993 [Schneier 93][Schneier 96]. O algoritmo consiste de duas partes: expansão de chave e criptografia dos dados. Blowfish usa uma chave cujo comprimento pode variar de 32 a 448 bits (1 a 14 palavras de 32 bits). Blowfish criptografa em 16 voltas blocos de texto claro de 64 bits em blocos codificados de 64 bits. Cada volta consiste de uma permutação dependente da chave e de uma substituição dependente da chave e do dado. Todas as operações são adições e OU-exclusivos sobre palavras de 32 bits. A única operação adicional é quatro matrizes indexadas de consulta de dados (Caixas-S).

Blowfish utiliza um grande número de subchaves. Tanto as subchaves como as caixas-S são pré-computadas antes de qualquer criptação/decriptação de dados. As subchaves são armazenadas em uma matriz P (P_1, p_2,..., P_{18}); existem quatro Caixas-S cada uma com 256 entradas de 32 bits ($S_{1,0}$,$S_{1,1}$,...,$S_{1,255}$; $S_{2,0}$,$S_{2,1}$,...,$S_{2,255}$; $S_{3,0}$,$S_{3,1}$,...,$S_{3,255}$; $S_{4,0}$,$S_{4,1}$,...,$S_{4,255}$.). Todo o processo de cálculo das subchaves e das caixas-S pode ser encontrado em [Schneier 96].

As principais características do algoritmo Blowfish são:

- pode ser executado em memórias menores que 5K;
- estrutura simples de fácil implementação;
- comprimento variável de chave, permitindo uma relação entre maior velocidade e maior segurança;
- as operações de adição e ou-exclusivo bit-a-bit não são comutativas, o que torna a criptoanálise mais difícil;
- Para geração da matriz de subchaves (matriz P) e das caixas-S são necessárias 521 execuções do algoritmo blowfish, portanto blowfish não pode ser utilizado em aplicações cujas chaves secretas mudam freqüentemente.

ADVANCED ENCRYPTION STANDARD – AES

Tendo em vista a fragilidade do DES e a baixa performance, bem como o custo elevado de implementação do algoritmo DES triplo, o National Institute of Standards Technology (NIST) identificou a necessidade de selecionar um novo padrão de criptografia que fosse mais rápido, barato e melhor, bem como, fornecesse maior segurança.

Em setembro de 1997 o NIST anunciou a seleção para o novo padrão de criptografia, o *Advanced Encryption Standard*, como o substituto do DES, que protegeu as comunicações governamentais dos Estados Unidos por mais de duas décadas [NIST 97]. As exigências do NIST foram que o novo algoritmo fosse um codificador de blocos simétrico e apresentasse as seguintes características:

- tamanho do bloco: 128 bits;
- comprimento da chave: 128, 196 ou 256 bits;

Vários algoritmos foram apresentados, e depois de muitas análises cinco algoritmos [Burwick 99], [Rivest 98], [Anderson 98], [Schneier 98], [Daemen 99]

foram finalistas, dentre eles o vencedor foi o algoritmo Rijndael [Daemen 01] que apresenta as seguintes características:

- tamanho do bloco: 128, 192 ou 256 bits;
- comprimento da chave: 128, 192 ou 256 bits;
- codificação do texto claro:

  A concepção do algoritmo Rijndael baseou-se no codificador de bloco Square [Daemen 95] em que cada volta regular envolve quatro passos ou camadas. A primeira camada envolve caixas-S orientadas por byte; a segunda camada envolve deslocamentos de linhas da matriz; na terceira camada as colunas da matriz são misturadas a partir de uma matriz de multiplicação, e na última camada a subchave é adicionada à matriz.

Na avaliação do NIST o algoritmo Rijndael foi escolhido por possuir as seguintes características [Flood 00]:

- utilizando uma chave de 128 bits, dispõe de uma velocidade média para criptação/decriptação de dados desenvolvendo a mesma performance em diferentes plataformas;
- execução rápida da chave (a execução da chave do Rijndael foi a mais rápida entre todos os algoritmos candidatos ao AES);
- é apropriado para ambientes com espaço reduzido de memória;
- boa eficiência quando implementado em hardware;
- enfim, a performance consistente durante o processo de avaliação influenciou a escolha do NIST para que o Rijndael fosse o selecionado para o AES.

Apesar do algoritmo Rijndael apresentar algumas vantagens em relação aos concorrentes, ele possui algumas desvantagens em relação a alguns algoritmos:

- as funções de criptação e decriptação não são semelhantes, logo necessitam de um espaço adicional quando as duas funções são implementadas.
- somente na criptação a computação das subchaves é realizada durante a operação (on-the-fly).
- a utilização de chaves maiores diminui a performance de criptação/decriptação.

2.1.2. Desvantagem dos Algoritmos Simétricos

Como já foi mencionada anteriormente, a segurança do algoritmo simétrico está na chave, pois um dos fundamentos da criptografia é que o inimigo conhece todos os detalhes do algoritmo, exceto a chave usada em uma dada codificação.

Para a troca de mensagens secretas entre duas partes, é necessário que as partes combinem uma chave secreta antes de iniciar o processo de troca. A chave combinada deve ser modificada com uma certa freqüência, a fim de evitar que um bisbilhoteiro a descubra e tenha acesso às mensagens codificadas. O principal desafio da criptografia simétrica é a geração, transmissão e armazenamento das chaves, ou seja, o gerenciamento de chaves.

Em geral, a criptografia convencional tem dificuldade em fornecer um gerenciamento de chave seguro, especialmente em sistemas abertos com um grande número de usuários.

Segundo Schneier os problemas dos algoritmos convencionais são [Schneier 96]:

- distribuição da chave: as chaves devem ser distribuídas de forma secreta, elas são tão valiosas quanto todas as mensagens que elas criptografam;
- revelação da chave: se a chave for revelada (roubada, adivinhada, extorquida, subornada, etc.), então terceiros podem decriptar todas as mensagens com a chave revelada, ou ainda fazer parte do sistema e produzir mensagens falsas enganando as outras partes;
- crescente número de chaves: o número de chaves cresce rapidamente, à medida que aumenta a quantidade de usuários de uma rede. Por exemplo, dois usuários precisam de apenas uma chave, mas uma rede com dez usuários precisa de 45 chaves para permitir que cada par de usuários se comunique com segurança. Em uma rede de cem usuários precisa-se de 4950 chaves diferentes. Enfim, uma rede de n usuários necessitará de $n(n-1)/2$ chaves.

2.2 Criptografia de Chave Pública

O conceito de criptografia por chave pública foi introduzido por Whitfield Diffie e Martin Hellman [Diffie 76] em 1976 com vistas a atacar os dois problemas mais difíceis associados à criptografia convencional: gerenciamento de chaves e assinatura digital (autenticação).

No sistema de criptografia por chave pública, também conhecida como assimétrico, cada usuário obtém um par de chaves, uma é chamada de chave pública, a outra é chamada de chave privada. A chave pública é publicada, enquanto a chave privada é mantida secreta. Através da chave pública uma mensagem pode ser codificada, mas a decodificação só poderá ser feita pelo detentor da chave privada.

A figura-2.3 ilustra o processo de criptografia de chave pública utilizado para codificação. A mensagem original ou texto claro é convertido em um texto codificado através de um processo de codificação que utiliza a chave pública do receptor da mensagem. Do lado do receptor, a mensagem é decodificada através de um algoritmo de decodificação e a chave secreta (privada) do receptor.

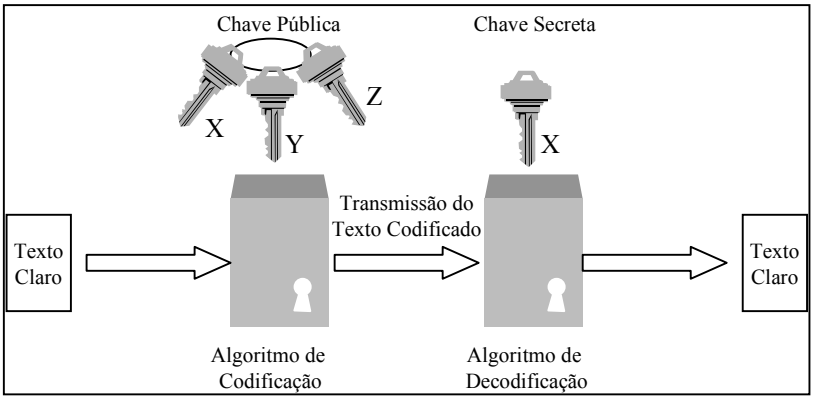

Figura-2.3: Processo de Criptografia de Chave Pública

O algoritmo de criptografia de chave pública além de oferecer confiabilidade, pode ser utilizado, também, para autenticação. A figura-2.4 exibe o processo de criptografia de chave pública utilizado para autenticação. Uma mensagem é codificada através do algoritmo de codificação e da chave secreta do emissor. Do lado do receptor, a mensagem é decodificada através do algoritmo de decodificação e da chave pública do emissor. Devido à mensagem ter sido criptografada usando a chave privada de X, somente X poderia ter codificado a mensagem, logo, a mensagem codificada serve como assinatura digital. Além de autenticar o emissor, este tipo de codificação autentica

também a integridade da mensagem, pois somente com a chave privada de X a mensagem poderia ser alterada.

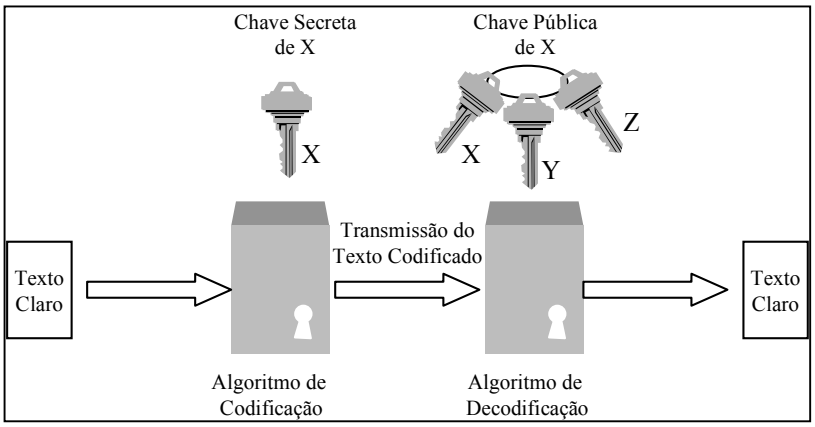

Figura-2.4: Autenticação

2.2.1. Principais Algoritmos De Chave Pública

RSA

O RSA é um sistema de criptografia de chave pública que foi desenvolvido por Ronald **R**ivest, Adi **S**hamir e Leonard **A**dleman em 1977 [Rivest 78]. O RSA executa tanto a codificação (confiabilidade) como assinatura digital (autenticação).

O processo de codificação do algoritmo RSA é realizado da seguinte maneira: adotam-se dois números primos grandes, p e q, e gera-se o produto dos mesmos $n = pq$; n é chamado de módulo. Escolhe-se um número e, menor que n e relativamente primo a (p–1). (q – 1), ou seja, o máximo divisor comum (mdc) entre e e (p – 1)(q – 1) é o fator 1, apenas. Encontra-se outro número d, tal que (ed – 1) seja divisível por (p – 1)(q – 1). Os valores e e d são chamados de expoentes público e privado, respectivamente. A chave pública é o par (n, e), e a chave privada é o par (n, d). Os fatores p e q devem ser destruídos ou mantidos com a chave privada. Atualmente, é difícil obter a chave privada d a partir da chave pública (n, e). A segurança do sistema RSA baseia-se na suposição que a fatoração de p e q é difícil.

O sistema RSA quando utilizado para codificação funciona da seguinte forma: o emissor para enviar uma mensagem M ao receptor, gera uma mensagem codificada C através da exponenciação $C = M^e \bmod n$, onde e e n constituem a chave pública do receptor. O emissor envia C para o receptor. Na decriptação o receptor também realiza

uma operação de exponeciação $M = C^d \bmod n$. A relação entre *e* e *d* garante que a mensagem M será recuperada corretamente pelo receptor. Somente o receptor pode decodificar a mensagem, pois somente ele conhece *d*.

O sistema RSA quando utilizado com assinatura digital executa o seguinte processo: o emissor quer enviar uma mensagem M ao receptor, de tal forma que o receptor tenha certeza que a mensagem foi enviada pelo emissor, e que a mesma não fora violada. O emissor cria uma assinatura digital *s*, através da fórmula s = Md mod n, onde *d* e *n* constituem a chave privada do emissor. O emissor envia M e *s* ao receptor. Para verificar a assinatura o receptor utiliza a chave pública do emissor, ou seja, a mensagem é recuperada através da fórmula M = se mod n, onde *e* e *n* constituem a chave pública do emissor.

O algoritmo de chave pública RSA executa a codificação e autenticação sem o compartilhamento da chave privada, cada usuário utiliza somente sua chave privada ou a chave pública de um outro usuário. Qualquer pessoa pode enviar uma mensagem criptografada ou verificar uma mensagem assinada, entretanto somente uma pessoa de posse da chave privada pode decriptar ou assinar uma mensagem.

Diffie-Hellman Key Exchange

Este algoritmo foi apresentado por Diffie-Hellman em 1976 e limita-se a troca de chaves entre dois usuários [Diffie 76], o algoritmo não serve para criptografar e decriptar mensagens. O objetivo do algoritmo Diffie-Hellman é permitir a dois usuários a troca de chave com segurança, as quais, em seguida, podem ser usadas para codificar mensagens.

A matemática do processo de troca de chaves é simples, entretanto está além do escopo desta dissertação demonstrá-la, em [Stallings 98] existe uma breve visão sobre esta matemática e em [Ore 76] e [Leveque 90] há um detalhamento maior. Inicialmente as partes escolhem números primos grandes, *p* e *q*, de maneira que *q* gere potências de todos os inteiros de 1 até $p - 1$, ou seja, *q* seja uma raiz primitiva mod *p*. Esses números não precisam ser secretos.

Supondo-se que dois usuários, A e B, desejam trocar uma chave, o protocolo de geração das chaves secretas é constituído dos seguintes passos:

- A escolhe aleatoriamente um número inteiro x, computa $X = q^x \bmod p$, e envia X para B;
- B escolhe aleatoriamente um número inteiro y, computa $Y = q^y \bmod p$, e envia Y para A;
- A gera a chave secreta k computando $k = Y^x \bmod p$;
- B gera a chave secreta k' computando $k' = X^y \bmod p$.

Ambas k e k'são iguais a $g^{xy} \bmod n$. Um bisbilhoteiro não pode computar este valor, pois somente os valores de p, q, X e Y são conhecidos. Os valores de x e y só poderiam ser recuperados computando-se o logaritmo discreto, que se constitui um problema matemático de certa maneira intratável (mais precisamente, NP-completo).

O algoritmo Diffie-Hellman depende da dificuldade da computação dos logaritmos discretos. Portanto, *p* deve ser um número grande e (*p*-1)/2 também deve ser primo. O valor de *g* pode ser qualquer, desde que gere um grande subgrupo de raiz primitiva mod *p* [Schneier 96].

Elliptic Curve Cryptography – ECC

Em 1985, os pesquisadores Neil Koblitz e Victor Miller usaram as curvas elípticas em um criptosistema de chave pública. Eles não inventaram um novo algoritmo criptográfico. Eles implementaram o algoritmo Diffie-Hellman usando curvas elípticas. A criptografia baseada em curvas elípticas vem ganhando destaque como alternativa a sistemas assimétricos mais convencionais. Atualmente, sistemas criptográficos de curvas elípticas são considerados o estado-da-arte em criptografia assimétrica [Barreto 99].

As curvas elípticas são construções matemáticas que podem ser definidas em qualquer campo (real, racional, complexo). Entretanto, na criptografia as curvas elípticas são definidas sobre campos finitos. Os matemáticos chamam de campo finito um número *n* que seja primo ou a potência de um primo grande [Schneier 96]. Uma curva elíptica é constituída de elementos (x, y) que satisfazem a equação $y^2 = x^3 + ax + b$, juntamente com um simples elemento denotado *O* chamado de ponto do infinito ou ponto zero, que pode ser visualizado como o ponto de cima e de baixo de toda linha vertical. A adição de dois números sobre uma curva elíptica é definida de acordo com

um conjunto simples de regras. A figura-2.5, por exemplo, exibe uma regra simples: o ponto p1 mais o ponto p2 é igual ao ponto –p3.

A curva elíptica permite a geração de outros grupos além do Z^*_p (grupo multiplicativo dos inteiros positivos módulo p) que torna mais difícil o problema do algoritmo discreto. A matemática é complexa e está além do escopo desta dissertação, entretanto em [Miller 86][Koblitz 87][Menezes 93] pode-se encontrar os principais conceitos envolvidos.

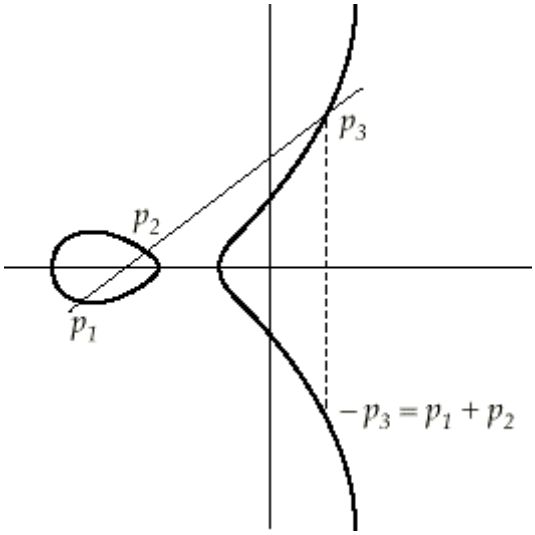

Figura-2.5: Exemplo de Curva Elíptica

A operação de adição em criptografia de curvas elípticas corresponde a operação de multiplicação modular do algoritmo RSA, enquanto que a adição múltipla corresponde a exponenciação modular. Para gerar um sistema criptográfico usando curvas elípticas precisa-se encontrar um problema difícil tal como fatorar o produto de números primos ou fazer o logaritmo discreto. Considerando a equação $Q = kP$, onde Q, $P \in E_p(a,b)$ e $k < p$, é relativamente fácil calcular Q dado k e P, porém é relativamente difícil determinar k dado Q e P.

A troca de chave proposta por Diffie-Hellman pode ser realizada através de curvas elípticas. O processo obedeceria aos seguintes passos [Stallings 98]:

- primeiro selecionaria um número primo $p \cong 2^{180}$ e os parâmetros da curva elíptica a e b para a equação $y^2 \cong x^3 + ax + b \pmod p$. Isto define o grupo elíptico dos pontos $E_p(a, b)$;
- em seguida selecionaria um ponto gerador $G = (x_1,y_1)$ em $Ep(a,b)$. Existe um critério importante para seleção do ponto gerador G: o valor de nG=O, onde O é um

simples elemento chamado o ponto zero, deve ser um número primo grande a partir do menor valor possível para n. Ep(a,b) e G são parâmetros do sistema criptográfico que todos os participantes conhecem.

Para a troca de chave entre os usuários A e B o processo é executado conforme abaixo:

- o usuário A seleciona um inteiro n_A menor que n, que será sua chave privada. A chave pública de A será gerada a partir de $P_A = n_A \times G$; a chave pública é um ponto em $E_p(a, b)$.
- o usuário B seleciona uma chave privada n_B e computa a chave pública P_B.
- o usuário A gera uma chave secreta $K = n_A \times P_B$ e, o usuário B gera uma chave secreta $K = n_B \times P_A$. As duas chaves secretas geradas pelos usuários A e B são iguais:

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

Para quebrar este sistema o atacante teria que ser capaz de computar k dado G e kG, o que é considerado muito difícil.

2.2.2. Desvantagens da Criptografia de Chave Pública

Uma desvantagem da criptografia de chave pública é a velocidade. A computação matemática usada para criptografar dados é intensiva, exigindo muito tempo de processamento, o que para algumas aplicações é inaceitável como é o caso do protocolo IPSec (protocolo que estabelece canais seguros de comunicação), que realiza, em tempo real, o processo de codificação e decodificação na porção de dados de cada pacote transmitido. Dependendo do algoritmo, o texto codificado pode ficar muito maior que o texto claro, tornando impraticável o uso corrente da criptografia de chave pública para criptografar mensagens longas.

Outra desvantagem do sistema de chave pública é o fato de uma mensagem criptografada só poder ser enviada a apenas um receptor. Considerando que, para criptografar uma mensagem, é necessário utilizar a chave pública do receptor, enviar uma mensagem para uma lista de receptores torna-se impraticável a utilização do método da criptografia de chave pública.

No sistema de criptografia de chave pública, assume-se que uma autoridade central mantém um diretório dinâmico contendo as chaves públicas de todos os participantes. Cada participante seguramente conhece uma chave pública para a autoridade central, e a autoridade por sua vez possui uma chave privada. Quando um dos participantes deseja obter a chave pública de um outro, faz uma solicitação à autoridade usando a chave pública da autoridade. A autoridade fornece a chave pública solicitada utilizando uma chave privada. Um ataque de sucesso à autoridade central permitirá um adversário controlar o diretório de chaves públicas, destruindo toda segurança do sistema, pois o mesmo poderia substituir uma chave de sua escolha para qualquer participante.

Em algumas situações o uso da criptografia de chave pública é desnecessário. Abaixo estão relacionadas situações em que o uso da criptografia de chave pública é desvantajosa:

- ambientes seguros onde a distribuição de chaves secretas pode acontecer, por exemplo reuniões privadas;
- ambientes onde uma única autoridade conhece e gerencia todas as chaves, por exemplo um sistema bancário fechado;
- ambientes de um único usuário, por exemplo, se um usuário desejar manter seus arquivos pessoais criptografados é mais coerente faze-lo através da criptografia convencional, ou seja, utilizando sua senha como chave secreta.

Em geral, a criptografia de chave pública é mais adequada em ambiente multi-usuário aberto.

2.3. Sistema Criptográfico Híbrido

A criptografia de chave pública não foi criada para substituir a criptografia convencional, mas para complementá-la, tornando-a mais segura [Diffie 76]. Para a criptografia a melhor solução é combinar os dois sistemas: chave pública e convencional, a fim de obter as vantagens da segurança da criptografia de chave pública e as vantagens de velocidade da criptografia convencional.

A estrutura denominada envelope digital apresenta a solução para os usuários, de um sistema de criptografia convencional, comunicarem-se com segurança, ou seja, sem

correrem o risco da chave ser interceptada durante a transmissão. Esta estrutura consiste de uma mensagem criptografada usando a criptografia convencional e de uma chave secreta criptografada.

A figura-2.6 mostra um exemplo da aplicação de um sistema criptográfico híbrido. Neste exemplo, o usuário A envia uma mensagem ao usuário B utilizando a criptografia convencional para codificar a mensagem e a criptografia de chave pública para transferir a chave de codificação da mensagem. Os passos são os seguintes:

- o emissor A escolhe uma chave secreta K e criptografa a mensagem com ela;
- em seguida criptografa a chave secreta K utilizando a chave pública do usuário B, K_{PB};
- o emissor A envia ao receptor B a mensagem e a chave criptografadas;
- o receptor B, para ler a mensagem, decripta a chave secreta K através da sua chave privada K_{KB} ;
- e em seguida decripta a mensagem com a chave secreta K.

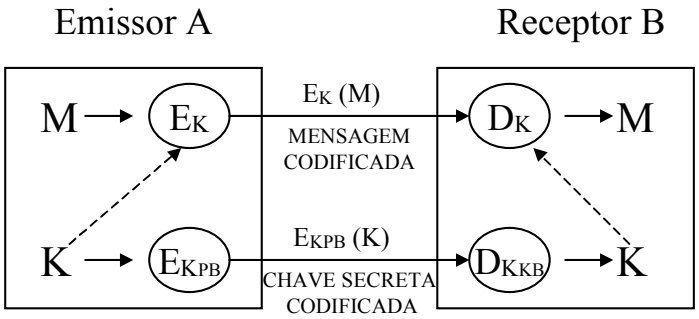

Figura-2.6: Sistema Criptográfico Híbrido

Os usuários desse sistema podem usar a chave secreta K para criptografar somente uma mensagem ou podem usá-la para uma comunicação estendida. Uma boa característica desta técnica é que a chave secreta pode ser trocada com freqüência. A troca de chave em curto espaço de tempo é o ideal para a segurança de um sistema, pois torna mais difícil um invasor descobrir a chave.

Sistema híbrido, como envelope digital, resolve o problema de gerenciamento de chave verificado na criptografia convencional, bem como aumenta a performance (comparando-se com a utilização da criptografia de chave pública para codificar diretamente a mensagem) sem sacrificar a segurança.

2.4. Principais Tipos de Ataques

O objetivo principal da criptografia é manter secreto o texto original, bem como a chave. O processo de tentativa para descobrir o texto original ou a chave é conhecido como criptoanálise. As estratégias empregadas pelo criptoanalista, indivíduo que executa a criptoanálise, dependem do tipo de algoritmo de criptografia e das informações disponíveis.

Basicamente são cinco os tipos de ataque sobre mensagens criptografadas (operação de criptoanálise) [Stallings 98][Schneier 96]. Em cada um deles, assume-se que o criptoanalista conhece o algoritmo utilizado:

- Ataque ao texto codificado

  O criptoanalista tem textos codificados de várias mensagens criptografadas pelo mesmo algoritmo. O trabalho do criptoanalista é recuperar o texto original do maior número de mensagens possível, ou melhor ainda, deduzir a chave (ou chaves) usada na codificação das mensagens. Recuperando a chave, o criptoanalista poderá no futuro usá-la para decriptar outras mensagens. Este tipo de ataque é mais fácil de se defender, pois o criptoanalista não dispõe de muitas informações para trabalhar.

- Ataque ao texto original conhecido

  O criptoanalista não só tem acesso aos textos criptografados de várias mensagens, mas também aos textos originais dessas mensagens. Neste método, o trabalho do criptoanalista é deduzir a chave (ou chaves) usada(s) para criptografar as mensagens, ou um algoritmo para decriptar novas mensagens codificadas com a mesma chave.

- Ataque ao texto original escolhido

  O criptoanalista além de ter acesso a várias mensagens originais e codificadas, pode escolher o par de textos original e criptografado. Este tipo de ataque é mais poderoso que um ataque ao texto original conhecido, porque o criptoanalista pode escolher blocos de texto original para criptografar, podendo conseguir mais informações sobre a chave. O trabalho do criptoanalista é deduzir a chave usada na codificação das mensagens.

- Ataque adaptativo ao texto original escolhido

  É um caso especial de ataque ao texto original escolhido. O criptoanalista pode escolher não somente o texto original que está criptografado, mas pode modificar a escolha baseada nos resultados de codificações anteriores. Em um ataque ao texto original escolhido, o criptoanalista pode estar apto a escolher um bloco grande de texto original para ser criptografado; em um ataque adaptativo ao texto original escolhido, o criptoanalista pode escolher um bloco menor de texto original e então escolher outro baseado nos resultados do primeiro, etc.

- Ataque ao texto codificado escolhido

  O criptoanalista pode escolher textos criptografados diferentes a serem decriptados e ter acesso ao texto original decodificado. Este ataque é mais aplicável aos sistemas de chave pública. Um ataque ao texto codificado escolhido também pode ser usado em algoritmos simétricos, mas devido a simetria desse sistema, é equivalente ao ataque a um texto original escolhido.

Algumas vezes o algoritmo de criptografia não é conhecido e um possível ataque nessa circunstância é chamado de força bruta. Este método procura descobrir o texto original através de tentativas de todas as chaves possíveis de codificação. Se o espaço de chave (faixa de valores possíveis) for muito grande, este método torna-se impraticável.

Os computadores estão se tornando cada vez mais rápidos e baratos, e devido a esse avanço da tecnologia, algoritmos considerados seguros no passado estão se tornando obsoletos. O comprimento das chaves dos codificadores de blocos tem aumentado consideravelmente tornando os ataques por força bruta impraticáveis. Nos últimos anos os métodos de ataque mais poderosos e promissores são as criptoanálises diferencial [Biham 93] e linear [Matsui 93].

A segurança de sistemas criptográficos está na dificuldade em quebrá-los. Se o tempo e o dinheiro necessários para quebrar um algoritmo for maior que o valor da informação criptografada, então ele pode ser considerado seguro [Stallings 98]. Alguns algoritmos só podem ser quebrados em um período de tempo superior a idade do universo e em um computador maior que toda matéria do universo. Esses algoritmos são teoricamente quebráveis, mas inquebráveis na prática. Um algoritmo inquebrável na prática é seguro.

2.5 Modos de Operação dos Codificadores de Bloco

A fim de capacitar o algoritmo DES para todas as possíveis aplicações de criptografia, foram definidos quatro modos de operação [FIPS 80]. Esses modos de operação podem ser aplicados a qualquer codificador de bloco simétrico.

2.5.1 Modo Livro de Código Eletrônico (Eletronic Codebook Mode - ECB)

É o modo mais simples de ciframento de bloco, e recebe esta denominação em virtude de um bloco de texto claro, dada uma chave, sempre resultar em um único bloco de texto codificado, independente de sua localização. Teoricamente, é possível criar um livro código de textos claros e seus correspondentes textos codificados. Entretanto, se os blocos forem de 64 bits existirão 2^{64} entradas para o livro de código, isto para cada chave, o qual torna muito elevado a computação e o espaço de armazenamento.

O modo ECB é o mais fácil de trabalhar. Cada bloco de texto claro é codificado independentemente, tornando-se útil para arquivos codificados em que se deseja acessar aleatoriamente, por exemplo, um banco de dados. Este modo também é ideal para pequenas quantidades de dados, tal como codificação de chave.

Para mensagens maiores o modo ECB não se mostra seguro, pois no caso de uma mensagem bem estruturada é possível explorar as regularidades da mensagem e compor um livro código. Para minimizar este problema, deve-se mudar a chave freqüentemente. Entretanto, a solução é utilizar o modo de encadeamento, o qual – a seguir - será apresentado.

2.5.2 Modo Codificador de Encadeamento de Bloco (Cipher Block Chaining Mode - CBC)

O modo CBC sugere um mecanismo de realimentação para um codificador de bloco, pois neste modo a entrada para o algoritmo de criptografia é o OU-exclusivo do

bloco de texto claro atual e o bloco codificado anteriormente; a mesma chave é usada para cada bloco.

Para produzir o primeiro bloco de texto codificado, faz-se um OU-exclusivo do bloco de texto claro com um vetor de inicialização. O vetor de inicialização evita que blocos de texto claro de mensagens idênticas gerem blocos iguais de texto codificado. Deste modo, é impossível um intruso tentar repetir um bloco, bem como se torna mais difícil a geração de um livro código. O vetor de inicialização deve ser conhecido por ambos, o emissor e o receptor. Todo o processo de codificação e decodificação desse modo de operação está especificado no apêndice C do FIPS PUB 81 [FIPS 80].

O modo CBC é apropriado para criptografar mensagens mais longas que 64 bits.

2.5.3 Modo Codificador Realimentado (Cipher-Feedback Mode - CFB)

Conforme mencionado na seção 2.1, os codificadores podem ser de bloco ou de fluxo. Através dos modos CFB e Output Feedback Mode (OFB) é possível converter um codificador de bloco em codificador de fluxo. Um codificador de fluxo elimina a necessidade de preenchimento de uma mensagem para obter o tamanho de bloco ideal, bem como, pode operar em tempo real. Dessa maneira, se um fluxo de caracteres está sendo transmitido, cada caractere pode ser criptografado e transmitido imediatamente usando um codificador de fluxo orientado por caractere.

No codificador de fluxo é desejável que o texto codificado seja do mesmo comprimento que o texto claro. Assim, se caracteres de 8 bits estão sendo transmitidos, cada caractere deve ser criptografado com 8 bits.

Esse modo de operação utiliza para entrada da função de criptografia um registrador de deslocamento de 64 bits, caso o codificador de bloco seja de 64 bits. A parte que realimenta o registrador de deslocamento, até que todo texto claro seja codificado, é o resultado do Ou-exclusivo realizado entre os 8 bits mais significativos da saída da função de criptografia e a unidade do texto claro, ou seja, é a unidade codificada. Em [Stallings 98], [Schneier 96] o processo de codificação/decodificação pode ser visto em detalhes. Uma aplicação típica do modo de operação CFB é a autenticação.

2.5.4 Modo de Saída com Realimentação (Output Feedback Mode - OFB)

O modo OFB é semelhante ao modo CFB. A diferença entre eles consiste da parte que realimenta o registrador de deslocamento da entrada da função de criptografia. No modo OFB a saída da função de criptografia é a parte que realimenta o registrador de deslocamento, enquanto que no modo CFB o registrador de deslocamento é realimentado pela unidade codificada.

O modo OFB é às vezes chamado de realimentação interna, porque o mecanismo de realimentação é independente dos fluxos de texto claro e texto codificado [Schneier 96]. Uma vantagem desse modo de operação é que erros de bits na transmissão não se propagam. Por exemplo, se ocorre um erro de bit na parte codificada C, somente o valor recuperado da parte do texto claro P é afetada; as unidades de texto claro subseqüentes não são modificadas.

Devido a vantagem, acima citada, uma aplicação típica desse modo de operação é a transmissão orientada por fluxo feita em canal ruidoso, por exemplo, comunicação por satélite.

Capítulo 3

Técnicas de Correção Antecipada de Erro

Estudos preliminares revelaram que a decodificação Viterbi juntamente com a codificação convolucional, que juntas constituem uma técnica de correção antecipada de erro, apresentam peculiaridades, que ajustadas, auxiliam a criptografia convencional. Portanto, as sub-seções que seguem apresentam as caracterísiticas das técnicas de codificação convolucional e decodificação Viterbi, as quais constituem a base do sistema de criptografia proposto.

3.1 Codificação Convolucional

A transmissão de dados por difusão utiliza técnica de correção antecipada de erro (Forward Erro Correction–FEC) a fim de garantir a transmissão correta dos dados. A utilização dessa técnica é conhecida como codificação de canal. Os principais métodos de codificação de canal são: codificação em blocos e codificação convolucional [Fleming 99]. A codificação em bloco opera com mensagens de até centenas de bytes, enquanto que a convolucional com dados seriais, um ou poucos bits de cada vez. O processo de codificação é acompanhado do processo de decodificação. No caso da codificação convolucional, as duas técnicas que realizam a decodificação são a decodificação Viterbi (DV) e a seqüencial. A decodificação Viterbi é bastante utilizada. As técnicas Codificação Convolucional e Decodificação Viterbi, juntas, constituem uma técnica de correção antecipada de erro (FEC).

A codificação convolucional produz uma seqüência de bits codificados na saída em função de bits não codificados na entrada.

O codificador convolucional basicamente é descrito através de dois parâmetros: restrição de comprimento (R) e a taxa de codificação (n/s). O parâmetro Q indica em

quantas saídas futuras do decodificador, a entrada atual irá influenciar. A taxa de codificação é a razão entre a quantidade de bit que entra (*n*) e a quantidade de bit que sai (*s*) em um dado ciclo do codificador .

Intrinsecamente relacionado ao parâmetro *Q* está o parâmetro *m*, o qual indica durante quantos ciclos um bit de entrada será conservado no decodificador. Em outras palavras, o parâmetro *m* indica a profundidade do pipiline do codificador, e pode ser entendido como o comprimento da memória do codificador.

A codificação do fluxo de dados é feita por dois componentes básicos do codificador: registrador de deslocamento e módulos somadores. Por toda dissertação será utilizado um exemplo simples e tradicional de codificador convolucional, o qual está exibido na figura-3.1. O codificador da figura 3.1 tem as seguintes características: Taxa de codificação (n/s) = 1/2, Q=3, m=2.

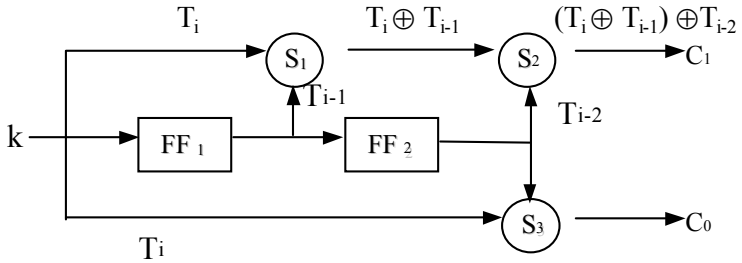

Figura-3.1: Diagrama Esquemático de um Codificador (1/2,3,2)

O codificador acima opera os bits de entrada a uma taxa de *n* bits, enquanto que a taxa na saída é de *s=2n* símbolos. O bit de entrada é estável durante o ciclo do codificador. No ciclo T_i o flip-flop FF_2 guardará a entrada do ciclo $T_{i-2,}$ o flip-flop FF_1 guardará a entrada do ciclo T_{i-1} e a entrada do codificador receberá a entrada atual. A saída C_1 (mais significativa) receberá o resultado de $(T_i \oplus T_{i-1}) \oplus T_{i-2}$. A saída C_0 receberá o resultado de $T_i \oplus T_{i-2}$.

O codificador também pode ser visto como uma máquina de estados (vide figura-3.2). No caso da figura-3.1, o codificador tem dois bits de memória (FF_1 e FF_2), logo terá quatro estados possíveis: 00_2, 01_2, 10_2 e 11_2. Inicialmente, através da iniciação (clear) do conteúdo dos flip-flops FF_1 e FF_2, o codificador é colocado no estado 00. Se o primeiro bit de entrada for 0, o próximo estado permanecerá em 00_2, porém se for 1 ocorrerá uma transição de estado, e o próximo estado será 10_2. A Tabela-3.1 exibe a transição dos estados em função dos bits de entrada.

Estado Atual	Próximo estado, se	
	Entrada = 0:	Entrada = 1:
00	00	10
01	00	10
10	01	11
11	01	11

Tabela-3.1: Transição de Estados

Conforme a taxa de codificação (1/2), para cada bit de entrada existirá 2 bits na saída. A tabela-3.2 exibe o estado atual da memória do codificador e os símbolos de saída em função das entradas. Por exemplo, se o estado atual do codificador é 00_2 e na entrada tem-se um 0, as saídas C_1 e C_0 serão 0, logo o símbolo de saída do codificador será 00_2. Se o bit de entrada for 1, as saídas C_1 e C_0 serão 1 e o símbolo de saída 11_2. A tabela-3.2 exibe os símbolos de saída do codificador de acordo com as entradas.

Estado atual	Símbolos de saída, se	
	Entrada = 0:	Entrada = 1:
00	00	11
01	11	00
10	10	01
11	01	10

Tabela-3.2: Símbolos de Saída

A figura-3.2 exibe a representação da máquina de estados das tabelas 3.1 e 3.2. A tabela 3.3 contempla todos os dados do codificador (1/2,3,2).

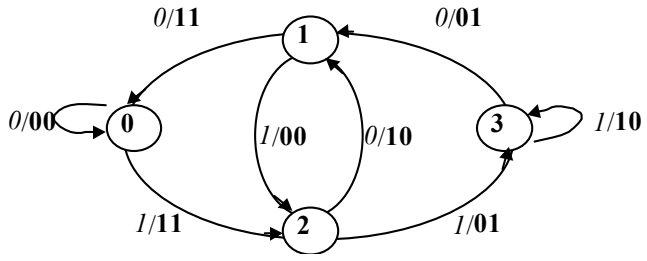

Figura-3.2: Máquina de Estados Finitos

Estado Atual	Entrada	Saída	Próximo Estado
0	0	00	0
	1	11	2
1	0	11	0
	1	00	2
2	0	10	1
	1	01	3
3	0	01	1
	1	10	3

Tabela-3.3: Dados do Codificador (1/2,3,2)

Nesse tipo de codificação (1/2,3,2) o fluxo de bit de entrada 0 1 0 1 1 1 terá como saída os seguintes símbolos: 00 11 10 00 01 10. A profundidade dessa codificação é *Q*=3, isto é, cada bit de entrada influenciará em três pares sucessivos de símbolos da saída. Portanto, para que o último bit de entrada influencie em três saídas sucessivas, são necessários mais dois símbolos de saída. O processo de geração de mais dois símbolos é chamado *flushing*. O *flushing* consiste em manter a entrada do codificador com 0 durante dois ciclos de modo a descarregar os flips-flops. Desse modo o fluxo de bit na saída do codificador convolucional será 00 11 10 00 01 10 01 11.

3.2 Decodificação Viterbi

O algoritmo Viterbi foi proposto, inicialmente, por Andrew J. Viterbi, como uma solução para a decodificação de códigos convolucionais [Viterbi 67]. Como já foi mencionado na seção anterior, o codificador convolucional pode utilizar duas técnicas de decodificação : Viterbi ou seqüencial. A técnica mais utilizada é a Viterbi, uma vez que ela tem um tempo fixo de decodificação, fato que favorece a implementação do decodificador em hardware [Fleming 99].

O algoritmo Viterbi além de ser bastante utilizado na área de comunicações, é utilizado também em outras áreas, tais como: rastreamento de alvo [Demirbas 81], reconhecimento de caracteres em palavras manuscritas e impressas [Ryan 93], segmentação de região e detecção de bordas de imagem [Pitas 89], [Pitas 87]. Este trabalho propõe mais uma utilização do algoritmo Viterbi, desta vez no campo da criptografia. Esta seção objetiva apresentar o que é relevante no processo de decodificação Viterbi para o entendimento do algoritmo proposto de criptografia.

A decodificação Viterbi trabalha segundo os parâmetros do codificador convolucional. Com a finalidade de reconstruir os bits de entrada do codificador, a

decodificação retira a redundância adicionada ao fluxo de bits, através de um processo probabilístico. O processo de decodificação Viterbi compara as probabilidades de ocorrência de um conjunto de transições possíveis de estado, e decide qual dessas transições tem a maior probabilidade de ocorrência [Ryan 93].

Para a decodificação Viterbi estimar a seqüência de entrada do codificador convolucional, utiliza-se um diagrama denominado treliça (figura-3.3), que representa todas as transições possíveis em função da entrada do codificador. A treliça da figura-3.3 representa um codificador convolucional de Q=3, taxa 1/2 e m=2 para uma mensagem de 15 bits. A tabela 3.4 exibe os estados, entradas, saídas e próximos estados da referida treliça.

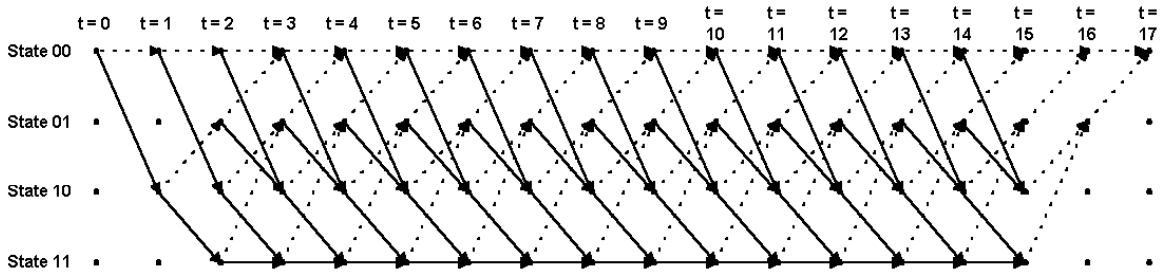

Figura-3.3: Treliça

Estado Atual	Entrada	Saída	Próximo Estado
0	00	0	0
	11	1	2
1	00	1	2
	11	0	0
2	01	1	3
	10	0	1
3	01	0	1
	10	1	3

Tabela-3.4: Dados da Treliça do Decodificador

Neste exemplo o codificador pode possuir até 4 estados, uma vez que o parâmetro *m* é igual a dois. A transição entre o estado atual e o próximo estado representa uma possível mudança de estado no codificador. Na treliça o resultado da decodificação (Saída) depende da entrada e do estado atual. Por exemplo, em t_0 o estado atual é 0. Se o par de bit de entrada for 00, o próximo estado do codificador, em t_1, será 0 e o resultado da decodificação será 0. Por outro lado, se o par de bit de entrada for 11, o próximo estado será 2 e o resultado da decodificação será 1. A figura-3.4 esboça o exemplo acima.

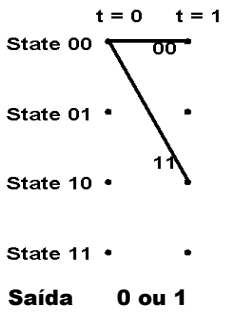

Figura-3.4: Transição de Estado em t_0

A decodificação Viterbi precisa basicamente de três unidades funcionais para encontrar, através de um processo probalístico, o fluxo de dados de entrada do codificador convolucional. As três unidades são: *Branch Metric*, *Add-Compare-Select* e *Traceback.*

O decodificador Viterbi, em um dado estado, rotula todas as transições possíveis com o valor da saída da codificação convolucional. Quando *n* bits alimentam a entrada do decodificador, ele compara o valor da entrada com o valor dos rótulos, e calcula uma medida de distância (*metric*). A distância pode ser linear e não-linear (logarítmica). Geralmente, a distância linear é mais aplicada, pois favorece à implementação de hardware mais simples, além de utilizar uma área menor. A distância linear é calculada através da distância de Hamming, que consiste simplesmente em contar quantos bits diferentes existem entre os símbolos recebidos e os símbolos que rotulam as trasições.

Branch Metric

A unidade *branch metric* tem por função calcular, a cada instante, a distância entre o valor do código recebido e os valores dos rótulos das transições. O resultado desse cálculo é armazenado e associado ao seu respectivo estado. Tomando como exemplo a figura-3.4, no estado 00_2, tem-se duas transições possíveis (estados futuros 00 e 10), rotuladas 00_2 e 11_2, respectivamente. Caso o símbolo 00_2 seja recebido na entrada do decodificador, tem-se – através da distância de Hamming – os resultados 0 e 2. O resultado 0 será associado ao estado 00_2 em t_1, enquanto que o resultado 2 será associado ao estado 10_2, também em t_1.

Na técnica de Correção Antecipada de Erro (FEC) o resultado do cálculo da distância representa a quantidade de erro existente em um dado estado, em um determinado instante de tempo. Esses erros são acumulados e chamados de medida de erro acumulado. Portanto, em t_1 (figura-3.5) os estados 00 e 10 teriam uma medida de erro 0 e 2, respectivamente.

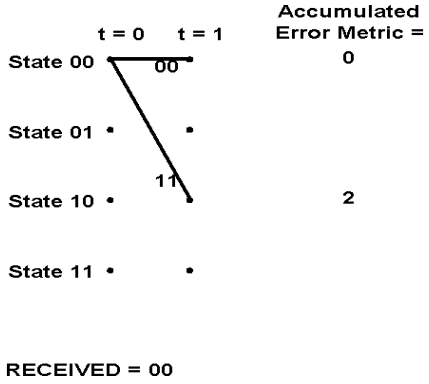

Figura-3.5: Transição de Estado em t_0 com Erro Acumulado

Add-Compare-Select

Os próximos passos da decodificação virterbi são feitos pela *unidade Add-Compare-Select* (ACS) a qual é tida como a mais importante de toda decodificação [Giulietti 98].

A unidade ACS, como o próprio nome indica, desempenha três funções: adição, comparação e seleção. A função de adição consiste em somar o valor da distância (*branch metric*) atual ao valor acumulado anteriormente em cada estado. Da figura-3.3, vê-se que o início da treliça está no estado 00_2. A partir desse estado tem-se duas transições possíveis, para o estado 00_2 ou para o estado 10_2. Supondo, como no exemplo dado anteriormente, que as distâncias em t_1 para estes estados são 0 e 2, respectivamente. A unidade ACS adicionará o valor da métrica inicial (zero) a estes valores de distância. Após a operação de adição a unidade ACS executa as outras funções: comparação e seleção. A operação *compare* verifica qual o estado que contém o menor valor acumulado de erro, enquanto que a função *select* seleciona o estado com menor valor, e armazena as informações em uma tabela, conhecida como *history of states*.

A medida que a treliça avança no tempo sua complexidade aumenta. Observando a figura-3.3 pode-se ver que até o tempo t_2, para cada estado, as transições só têm um estado predecessor. Entretanto, a partir de t_3 cada estado passa a ter dois estados predecessores. No caso de existir mais de um estado predecessor, a unidade ACS, ao executar o cálculo do erro acumulado, leva em consideração todas as transições que geraram o estado atual. Caso os resultados sejam iguais, a escolha será feita segundo um critério a ser adotado pelo desenvolvedor. A figura-3.6 exibe a treliça até t_3, onde pode ser verificada a situação acima.

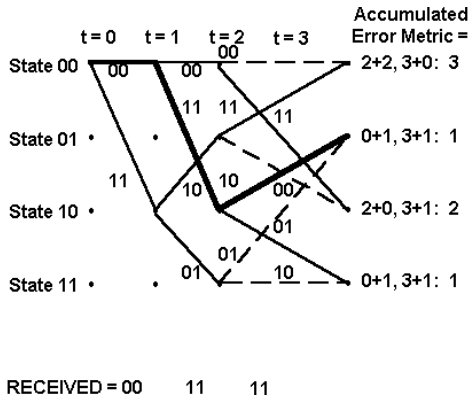

Figura-3.6: Transição de Estado em t_3

Traceback

Na decodificação Viterbi, existem duas técnicas de organização de memória para armazenar a seqüência de estados com menor valor de erro acumulado, da qual a seqüência do fluxo de bit original será recuperada – são elas: *register exchange* e *traceback* [Feygin 96a].

A técnica register exchange, do ponto de vista de concepção, é a mais simples, porém tem a desvantagem de cada bit decodificado pecisar ser lido e reescrito na memória. A técnica *traceback* utiliza ponteiros para interpretação dos símbolos, não necessitando, deste modo, mover dados na memória. Além disso, a técnica traceback consome menos área e largura de banda [Feygin 96a], [Feygin 96b]. Nesta dissertação, a técnica estudada para recuperar o fluxo de bit original foi a *traceback*.

A unidade traceback recria a seqüência de bits que estavam na entrada do codificador convolucional, através de três passos:

1. Seleciona, em cada instante de tempo, o estado de menor medida de erro acumulado e salva o número do estado. A tabela-3.5 exibe, a cada instante, o erro acumulado em cada estado, para o fluxo de dados inicial 0 1 0 1 1 1, dado como exemplo na seção do codificador convolucional. A tabela-3.6 apresenta o resultado da seleção do número do estado relacionado a menor medida de erro.

t =	0	1	2	3	4	5	6	7	8
Estado 0	0	0	2	3	3	3	3	4	1
Estado 1			3	1	2	2	3	1	4
Estado 2		2	0	2	1	3	3	4	3
Estado 3			3	1	2	1	1	3	4

Tabela-3.5: Medida de Erro Acumulado

t =	0	1	2	3	4	5	6	7	8
	0	0	2	1	2	3	3	1	0

Tabela-3.6: Número do Estado com Menor Erro Acumulado

2. A partir do estado atual seleciona-se o estado predecessor e salva-o. Este passo é executado iterativamente até atingir o início da treliça, e é conhecido como *traceback*.

3. Por último, usando a tabela-3.7 que exibe as transições de estados causadas pelas entradas, pode-se recriar a mensagem original. Os dois símbolos originados pelo *flushing* são descartados.

Estado Atual	Dado próximo estado, a entrada foi =			
	$00_2 = 0$	$01_2 = 1$	$10_2 = 2$	$11_2 = 3$
$00_2 = 0$	0	x	1	x
$01_2 = 1$	0	x	1	x
$10_2 = 2$	x	0	x	1
$11_2 = 3$	x	0	x	1

Tabela-3.7: Transição de Estado Causada pelas Entradas.

t =	1	2	3	4	5	6
	0	1	0	1	1	1

Mensagem Original

Em [Viterbi 79] encontra-se uma completa descrição de todo processo de decodificação.

Capítulo 4

Algoritmos: Viterbi Modificado e PAPÍLIO

O algoritmo de criptografia convencional, PAPÍLIO, proposto nesta dissertação, usa o algoritmo Viterbi Modificado como função F do codificador Feistel, a fim de atingir os princípios de difusão e confusão do codificador Feistel. Além da função F, Viterbi Modificado também é utilizado para geração das subchaves. As seções seguintes apresentam: a motivação para utilização do algoritmo Viterbi em criptografia, o algoritmo de geração das tabelas e o algoritmo de criptografia PAPÍLIO.

4.1. Motivação para utilização do Algoritmo Viterbi em Criptografia

A partir de uma série de análises, testes, e definições contidas em [Viterbi 79] constatou-se que:

- Uma seqüência de 5 bits na entrada do codificador (n/s=1/2, K=3, m=2) geraria o diagrama de árvore da figura 4.1. Na figura 4.1 observa-se ambas as seqüências de entrada e saída do codificador. As entradas são indicadas pelo caminho seguido no diagrama de árvore, enquanto que as saídas são indicadas pelos símbolos ao longo das ramificações da árvore. Uma entrada 0 (zero) especifica a ramificação superior de uma bifurcação, enquanto que uma entrada 1 (um) especifica a ramificação inferior. Através da figura 4.1 identifica-se facilmente que o fluxo de entrada 01011 geraria a codificação 0011100001. Assim, no diagrama da figura 4.1 pode-se identificar todas as seqüências de saída correspondentes a todas as 32 seqüências possíveis para os cinco primeiros bits de entrada.

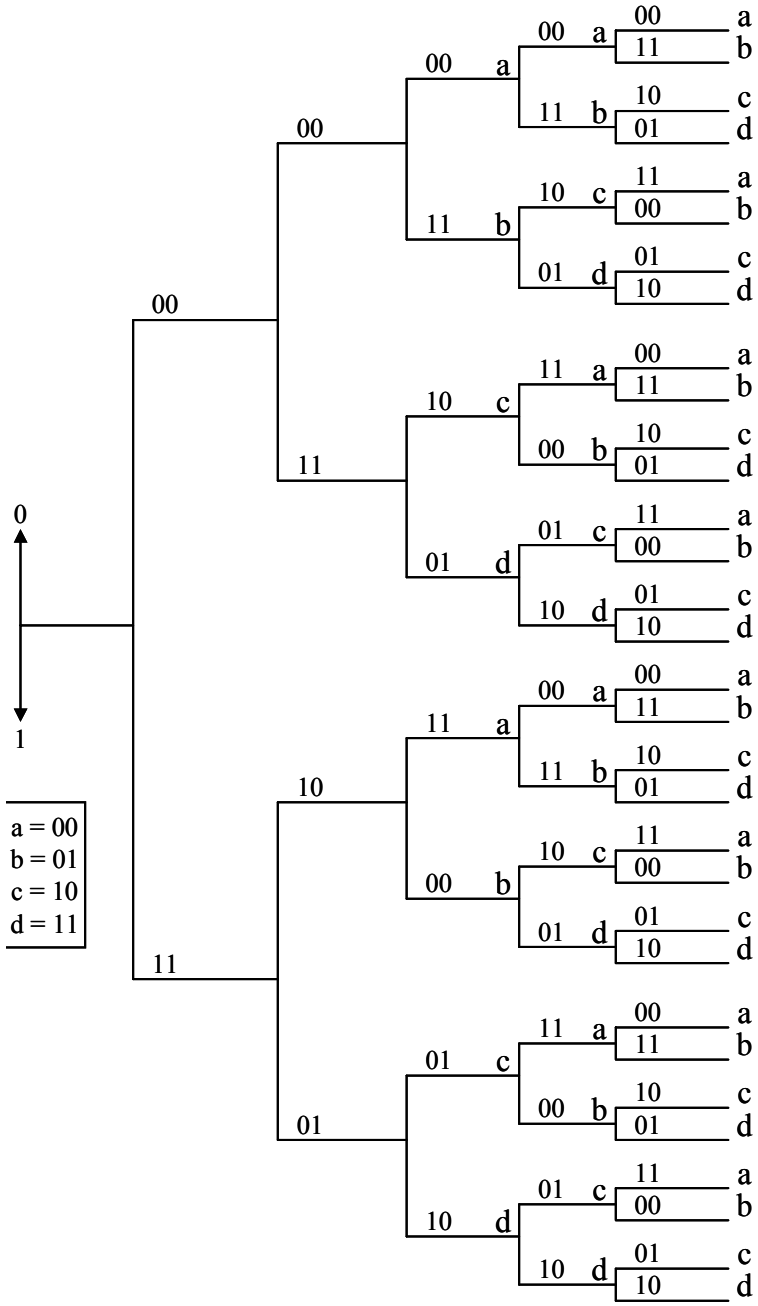

Figura-4.1: Diagrama de árvore do codificador (1/2, 3, 2)

- um fluxo de 5 bits na entrada do codificador convolucional, por exemplo, gerará uma seqüência de 10 bits na saída. Dez bits podem ser seqüenciados em 1024 formas distintas. Entretanto, como foi observado no item anterior, para toda seqüência de entrada de 5 bits haverá somente 32 seqüências possíveis de saída. Logo, existirão 992 seqüências que apresentarão erro ao serem decodificadas pelo algoritmo Viterbi.

- a quantidade de transições saindo e chegando em cada estado do codificador convolucional depende da quantidade de bits (n) da entrada, ou seja, de 2^n. Para um codificador com taxa de codificação de 1/2, Q=3 e m=2, cada estado teria 2^1 transições de saída e 2^1 transições de chegada (vide figura 3.2).

A característica que a Decodificação Viterbi (DV) possui de retirar redundância inserida pela codificação convolucional, por outro lado, poderia ser vista como uma codificação, pois cada par de bits da entrada resulta em apenas um bit. Essa codificação do ponto de vista de armazenamento e transmissão seria perfeita, pois – ao mesmo tempo – codificaria e comprimiria o dado. Entretanto, para que essa codificação fosse possível, seria necessário que o algoritmo Viterbi tratasse, por exemplo, para o caso de seqüências de 5 bits, as 1024 seqüências possíveis de 10 bits, ou seja, seria necessário que cada estado pudesse tratar qualquer par de bits de entrada.

Do lado da decodificação Viterbi, as constatações feitas com a ajuda da figura 4.1 podem ser observadas através da tabela 3.4. Os estados atuais da decodificação Viterbi possuem um espaço de bits de entrada limitado. Por exemplo, os estados 0 e 1, admitem as entradas 00 e 11, enquanto que os estados 2 e 3 admitem as entradas 01 e 10. Portanto, se o estado atual for 0 ou 1 e as entradas forem 01 ou 10 ocorrerá erro (rótulo não tratável). O erro também ocorrerá quando o estado atual for 2 ou 3 e as entradas forem 00 ou 11.

O fato do estado atual não permitir todo tipo de entrada (00,01,10,11) faz com que o algoritmo Viterbi – na sua forma original – não sirva para codificação, pois a cada instante em que o estado atual fosse incompatível com o par de bits da entrada, o processo seria interrompido, deixando de gerar um bit na saída do codificador.

Para adequar o algoritmo Viterbi para criptografia, gerou-se mais um fluxo de saída além daquele de codificação já existente. A forma como o novo fluxo foi gerado está descrito na seção 4.2. É importante salientar que só foi possível tratar qualquer par de bits de entrada, independente do estado atual, após a criação desse novo fluxo. Na realidade um par de bits passou a gerar um novo par de bits, e esse processo de geração deu origem às tabelas que substituem as caixas-S dos algoritmos simétricos da seção 2.1.1.

As tabelas geradas quando usadas com a função F do codificador Feistel alcançam as propriedades tão desejadas em qualquer algoritmo de criptografia: difusão,

confusão e efeito avalanche. Estas propriedades serão comentadas e analisadas no capítulo 5.

4.2. Algoritmo Viterbi Modificado (VM)

O algoritmo desenvolvido por Andrew J. Viterbi foi proposto como solução para a decodificação convolucional. A codificação convolucional como foi vista na seção 3.1 possui uma taxa de codificação variada (1/2, 2/3, 3/4, etc.). Os símbolos de saída da codificação convolucional representam a entrada para o algoritmo Viterbi. Portanto, o algoritmo Viterbi só decodifica símbolos gerados pela codificação convolucional.

Na codificação convolucional a transição de estado é rotulada com os símbolos de saída (vide figura 3.2). Conforme constatada na seção 4.1, a quantidade de transições de cada estado (2^n) é determinada pela quantidade de bits da entrada (n), logo para n = 1 haverá 2^1 transições de chegada e 2^1 transições de saída em cada estado; para n = 2 haverão 2^2 transições de chegada e 2^2 transições de saída em cada estado, e assim por diante. Assim, o algoritmo Viterbi só tratará a quantidade de bits gerada na saída do codificador convolucional. Além disto, cada estado só tratará um subconjunto dos rótulos de transição, provocando, desta forma, uma limitação no espaço de decodificação.

O objetivo do algoritmo Viterbi Modificado é ampliar o espaço de decodificação, a fim de possibilitar a geração de tabelas que serão utilizadas na criptografia de dados. A ampliação do espaço de decodificação consiste do estado atual tratar qualquer fluxo de bits de entrada.

O algoritmo Viterbi Modificado, inicialmente, adota os princípios da Codificação Convolucional (CC) e da Decodificação Viterbi (DV). A partir da definição dos componentes básicos do codificador: registradores de deslocamento e módulos somadores, determinam-se os polinômios geradores do par de bits da saída. Por exemplo, o polinômio gerador da saída C_1, da figura 3.1, é $T_i \oplus T_{i-1} \oplus T_{i-2}$, e o da saída C_0 é $T_i \oplus T_{i-2}$.

Com os componentes básicos e os polinômios definidos, a tabela da codificação convolucional é gerada. A quantidade de estados que comporão a tabela dependerá do parâmetro *m* do codificador. A fim de facilitar o entendimento o exemplo do codificador

convolucional (n/s=1/2,Q=3, m=2) será utilizado. Na figura 3.1 *m* é igual a 2, logo, a tabela terá quatro (2^2) estados possíveis. A tabela-3.3 exibe os valores de saída e de próximo estado, caso as entradas sejam 0 ou 1, para todos os estados possíveis do codificador da figura 3.1.

O erro no algoritmo Viterbi surge toda vez que o rótulo de entrada não pode ser tratado pelo estado atual. A execução do algoritmo Viterbi Modificado inicia-se no estado 0 e executa o algoritmo Viterbi, descrito na seção 3.2, até que um rótulo de entrada não seja tratável no estado atual. A fim de tratar qualquer rótulo de entrada, independentemente do estado atual, gerou-se além da saída já existente (S_0) mais uma saída (S_1). A saída S_0 representa o resultado do algoritmo Viterbi propriamente dito, ou seja, corresponde a um conjunto de rótulos de saída, um rótulo de saída para cada rótulo de entrada. Para o exemplo de um algoritmo Viterbi (n/s=1/2,Q=3, m=2) cada par de bits de entrada (rótulo de entrada) gera um bit de S_0 (rótulo de saída). A saída S_1 informa se cada rótulo de saída de S_0 foi obtido de acordo com o algoritmo Viterbi, ou se foi obtida de forma especial (Viterbi Modificado). Em outras palavras, informa quando um rótulo de entrada não era tratável pelo estado atual do polinômio gerador do algoritmo Viterbi. Quando um rótulo de saída de S_0 é obtido de acordo com o algoritmo Viterbi a saída S_1 gera o bit 0, do contrário a saída S_1 gera o bit 1.

A tabela-3.4 exibe os dados da treliça da decodificação Viterbi para o codificador da figura-3.1. Para que seja possível executar uma codificação a partir dos dados da tabela-3.4 é necessário que cada estado atual possa tratar todos os rótulos de entrada (00,01,10,11), caso contrário a codificação não se processará. Por exemplo, se o estado atual for 0 e a entrada for 01, não tem como codificar este par de bits. No estado zero, segundo a tabela 3.4, somente os pares 00 e 11 poderão ser codificados para 0 e 1, respectivamente.

A partir da tabela gerada pela codificação convolucional, identifica-se os rótulos que poderão ser tratados por cada estado, acrescentando ao novo fluxo de saída (S_1) a informação 0. Por exemplo, no caso da tabela-4.2, os pares 00 e 11 são tratados pelos estados 0 e 1, neste caso, os fluxos de saída dos respectivos pares receberão a informação 0 na saída S_1, enquanto a saída S_0 receberá o resultado do algoritmo Viterbi.

No caso dos rótulos de entrada que não podem ser tratados pelo estado atual, a proposta apresentada nesta dissertação é que o rótulo não tratável passe pelo processo de codificação convolucional considerando isoladamente cada bit formador do rótulo.

Para o processo de codificação convolucional usa-se como estado inicial o estado atual. Nota-se que a codificação convolucional vai gerar $\lceil s/n \rceil$ de rótulos adicionais, sendo n/s a taxa de codificação convolucional. Com este procedimento os rótulos gerados podem ser tratados pelo algoritmo Viterbi. A aplicação do algoritmo Viterbi geraria um rótulo de tamanho n para cada um dos rótulos adicionais gerados. Entretanto o que interessa para a codificação é a geração de um único rótulo de tamanho n. A solução adotada consiste em considerar, para compor o fluxo S_0, apenas o primeiro dos $\lceil s/n \rceil$ rótulos de tamanho n (o fluxo S_1, nestes casos, e só nestes casos recebe o valor 1). Para a continuação do processo de codificação usando o algoritmo Viterbi adota-se como estado atual o estado final do processo de codificação convolucional, até que um novo rótulo não tratável seja encontrado ou terminar a codificação.

Como exemplo de execução do algoritmo Viterbi Modificado, considerando uma taxa de 1/2, tem-se:

Seja o fluxo de bits 0011000001, nota-se que os dois primeiros rótulos de dois bits podem ser tratados com o algoritmo Viterbi iniciando-se no estado 0 (zero). O terceiro rótulo, entretanto, não pode ser tratado, pois, o estado atual, no momento de sua codificação é o estado 2 (dois), ver tabela 3.4. O processo de codificação convolucional do rótulo não tratável resulta nos rótulos adicionais 10 e 11 e no estado final 0 (zero). O quarto rótulo é tratável, usando-se o algoritmo Viterbi, pelo estado final do processo de codificação convolucional, resultando, após a aplicação do algoritmo de viterbi no estado atual 0 (zero). Mais uma vez nota-se, neste exemplo, que o quinto rótulo não é tratável por este estado. Repete-se então o processo de codificação convolucional, seguido de decodificação Viterbi, considerando-se apenas o primeiro rótulo (de tamanho 1) da saída da decodificação Viterbi.

Tem-se assim:

Fluxo inicial:	00 11 <u>00</u> 00 <u>01</u>	Os rótulos não tratáveis estão sublinhados
Fluxo de saída S_0	0 1 x x x	incerto após o segundo rótulo devido a intratabilidade
Codificação convolucional	10 11	Estado final da Codificação convolucional é o estado 0 (zero)
Decodificação Viterbi	- - 0 0	Usa-se como estado inicial o estado atual.
Fluxo de saída S_0	0 1 0 0 x	Considera-se apenas o primeiro rótulo de tamanho 1 bit. Continua-se a aplicar o algoritmo Viterbi usando-se como estado inicial o estado final da convolução (estado zero)
Codificação convolucional	00 11	Estado final da Codificação convolucional é o estado 2 (dois)
Decodificação Viterbi	- - - - - 0 1	Usa-se como estado inicial o estado atual.
Fluxo de saída S_0	0 1 0 0 0	Considera-se apenas o primeiro rótulo de tamanho 1 bit. Continua-se a aplicar o algoritmo Viterbi usando-se como estado inicial o estado final da convolução (estado dois)
Fluxo de saída S_1	0 0 1 0 1	Igual a 1 (um) apenas onde os rótulos não são tratáveis

Tabela-4.1: Exemplo de Execução do Algoritmo Viterbi Modificado

Os fluxos de bit S_0 e S_1 são independentes. Ao final da codificação os dois fluxos são concatenados, de forma a gerar um fluxo de bit de mesmo tamanho do original.

Para o caso de um algoritmo Viterbi Modificado, baseado no algoritmo Viterbi com os polinômios da figura 3.1 e os parâmetros (n/s=1/2,Q=3, m=2), o processo descrito acima pode ser simplificado usando-se a tabela-4.2.

Estado Atual	Entrada	S_0	S_1	Próximo Estado
0	00	0	0	0
	01	0	1	2
	10	1	1	1
	11	1	0	2
1	00	1	0	2
	01	0	1	2
	10	1	1	1
	11	0	0	0
2	00	0	1	0
	01	1	0	3
	10	0	0	1
	11	1	1	3
3	00	0	1	0
	01	0	0	1
	10	1	0	3
	11	1	1	3

Tabela-4.2: Resultado do Algoritmo Desenvolvido

4.3. PAPÍLIO: Algoritmo de Criptografia Proposto

O algoritmo criptográfico proposto é do tipo convencional. O processo para cifrar o texto claro é realizado em blocos. Para cada bloco de entrada é produzido um bloco de saída. A seguir são apresentadas as características do algoritmo proposto.

4.3.1. Características do Algoritmo Proposto

Tamanho do Bloco

Com o objetivo de evitar a análise estatística por parte de um intruso, bem como evitar complexidade de implementação, o tamanho do bloco para codificação foi definido em 64-bits. Este tamanho é reconhecido como suficientemente forte [Stallings 98] e é utilizado na maioria dos sistemas criptográficos, tais como: DES, Blowfish, IDEA, etc.

Tamanho da Chave

A segurança de um algoritmo convencional depende do segredo da chave, e não do algoritmo. Os tamanhos das chaves criptográficas são medidas em bit e a dificuldade de tentar todas as chaves possíveis cresce exponencialmente com o número de bits usados. Adicionando-se um bit à chave, dobra-se o número de chaves possíveis; adicionando 10 bits aumenta aos milhares (2^{10}) o número de chaves possíveis. Diferentes algoritmos simétricos possuem diferentes tamanhos de chave. Quanto maior o tamanho da chave, maior a segurança, mas a velocidade de codificação/decodificação diminui.

Assume-se que é impraticável decriptar uma mensagem somente com o conhecimento dos algoritmos de codificação/decodificação e do texto codificado. Com o avanço da tecnologia de hardware, o tamanho de 64 bits mostra-se inadequado. No relatório [Blaze 96] apresentado por um grupo de criptógrafos em janeiro de 1996, consta que para proteger uma informação para os próximos 20 anos, em face aos avanços do poder computacional, um sistema de formação recente deve ter uma chave de no mínimo 90 bits de comprimento. Atualmente 128 bits tornou-se o tamanho comum, e 128-bits é o tamanho da chave utilizada neste algoritmo.

Número de Voltas

Visando dificultar a criptoanálise[1], adotou-se a quantidade de 16 voltas, pois mesmo que a tabela de codificação não ofereça uma segurança adequada, uma maior quantidade de voltas dificulta a execução da criptoanálise. O codificador Feistel utiliza 16 voltas, pois a segurança oferecida por apenas uma volta é inadequada.

Geração de Subchaves

Este algoritmo utiliza uma chave de 128-bits, a qual gera 16 subchaves. As subchaves criadas são armazenadas temporariamente em uma matriz até que o processo de codificação do bloco seja finalizado. Para geração das subchaves (vide figura 4.2), as tabelas de codificação da seção 4.1 são utilizadas. O esquema de geração é o seguinte: as quatro primeiras subchaves, rotuladas SC_1, SC_2, SC_3 e SC_4 são geradas aplicando-se a codificação Viterbi Modificado na chave inicial de 128-bits, a qual gera dois fluxos de

[1] Ciência que recupera o texto original de uma mensagem sem o conhecimento da chave.

64-bits. Os dois fluxos de 64-bits passam pelo mesmo processo anterior e geram quatro fluxos de 32-bits que correspondem às quatro primeiras subchaves. Para gerar as quatro subchaves seguintes, concatenam-se os quatro fluxos gerando um só de 128-bits e aplicando-se todo o processo novamente até que gere mais quatro subchaves. O esquema se repete até que, dezesseis subchaves sejam criadas.

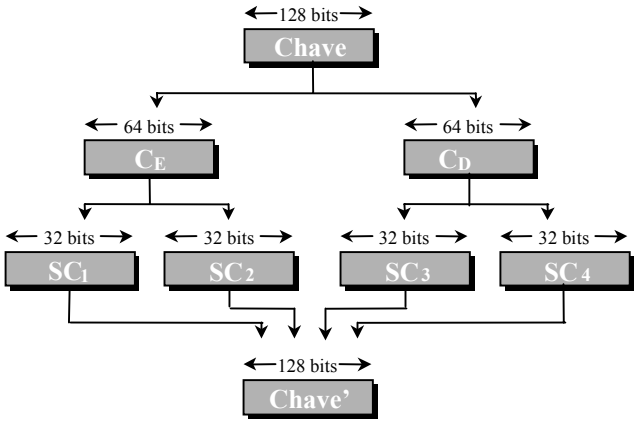

Figura-4.2: Estrutura das Subchaves

Função Utilizada

Em geral, os sistemas de criptografia utilizam diferentes funções para conseguirem uma boa difusão e confusão. No algoritmo proposto, buscam-se esses elementos através da função de substituição que utiliza as tabelas geradas a partir do algoritmo Viterbi Modificado. A figura-4.3 mostra a estrutura do codificador.

Como mostrado na figura-4.3, o algoritmo proposto deriva da estrutura clássica do codificador Feistel. Uma volta V do codificador criptografa através da divisão do bloco de entrada (texto claro) de 64 bits em duas partes: lado esquerdo E e lado direito D. O bloco do lado direito D_1 é transformado pela função F juntamente com a subchave da volta, e depois é executado um ou-exclusivo bit-a-bit com o bloco do lado esquerdo E_1. Em seguida os dois blocos são trocados de lado e o processo continua até a volta dezesseis. Após dezesseis voltas, executa-se mais uma troca fazendo com que os blocos retornem aos seus lados de origem. Conseqüentemente, para uma volta i, $1 \leq i \leq V$ do codificador, E_i e D_i representam os blocos da esquerda e direita, respectivamente, e SC_i representa a subchave em bits aplicada a função F. O algoritmo proposto pode ser visualizado como a seguir:

$E_{i+1} = D_i$

$D_{i+1} = E_i \oplus F(D_i, SC_i)$

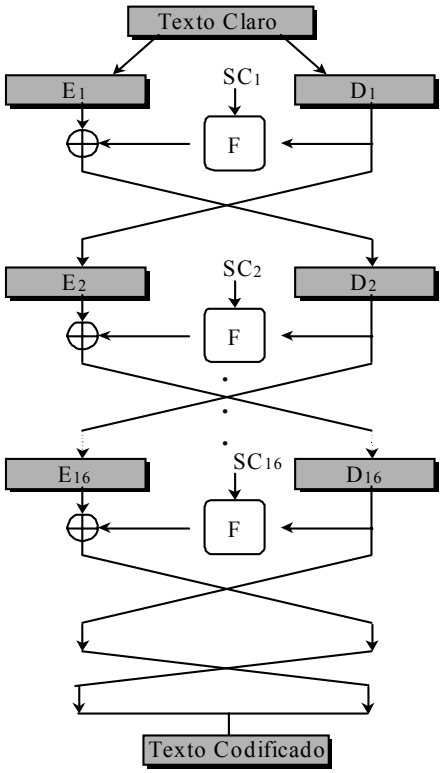

Figura-4.3: Estrutura do Codificador

A função F, detalhada na figura-4.4, executa inicialmente um OU-exclusivo (XOR) bit a bit da subchave SC_1 de 32 bits com o bloco do lado direito D_1 de 32 bits. O resultado do ou-exclusivo passa através de uma função de substituição que produz duas saídas de 16 bits (S_0 e S_1). Os dois fluxos de saída, S_0 e S_1, são concatenados gerando um bloco de 32 bits. Executa-se um OU-exclusivo do bloco de saída da função de substituição com o bloco E_1.

O papel da função F é tornar o texto codificado dependente da chave e do texto claro, bem como fazer com que cada bit do texto claro e da chave influencie em muitos bits do texto codificado.

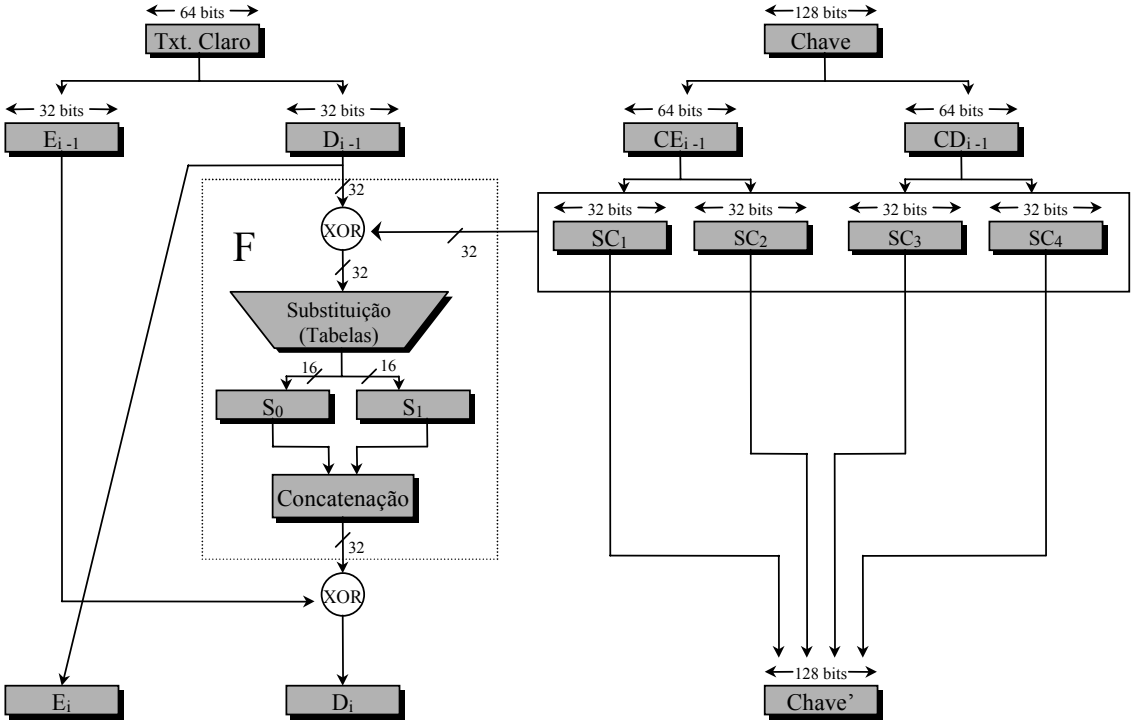

Figura-4.4: Uma volta do algoritmo PAPÍLIO

O processo de decriptação do algoritmo proposto (figura-4.5), é essencialmente o mesmo do processo de criptografia. Inicialmente, toma-se como entrada o texto codificado e utilizam-se as subchaves em ordem reversa, ou seja, SC_{16} na primeira volta, SC_{15} na segunda volta, e assim por diante até SC_1 ser utilizada na última volta. Esta forma de execução evita a implementação de um algoritmo para criptografia e um outro para decriptação.

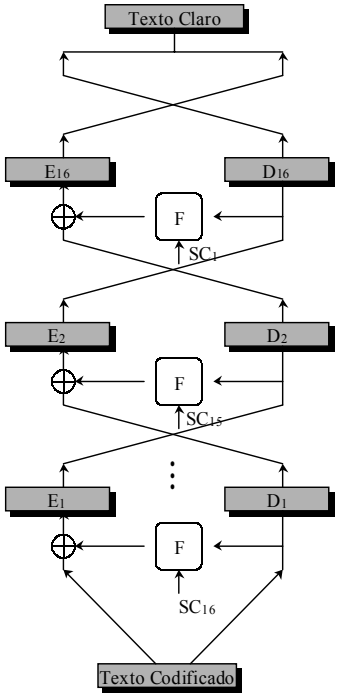

Figura-4.5: Estrutura do Decodificador

Abaixo está a prova da validade do processo de decriptação do codificador.

I. A saída da primeira volta do processo de decriptação é igual à entrada, trocada, da décima sexta volta do processo de codificação.

II. Observando o processo de codificação nota-se que o lado esquerdo da última volta é igual ao lado direito da penúltima volta, ou seja, $E_{16} = D_{15}$,onde: E_{16} representa o lado Esquerdo do processo de Codificação da 16ª volta, e D_{15} representa o lado Direito do processo de Codificação da 15ª volta.

III. Observa-se, também, que o lado Direito do processo de Codificação da última volta (D_{16}) é igual ao lado esquerdo da 15ª OU-Exclusivo o resultado da função F. A função F (figura-4.3) é o resultado do algoritmo Viterbi Modificado (tabelas da seção-4.1) aplicado, subchave SC16 OU-exclusivo DC15. Portanto, $DC_{16} = EC_{15} \oplus VM(DC_{15},K_{16})$.

No lado da decriptação tem-se:

IV. $ED_1 = DD_0 = EC_{16} = DC_{15}$, onde ED e DD significam lados esquerdo e direito do processo de decodificação, respectivamente. Enquanto, EC e DC significam lados esquerdo e direito do processo de codificação.

V. $DD_1 = ED_0 \oplus VM(DD_0, SC_{16})$

$= DC_{16} \oplus VM(DC_{15}, SC_{16})$ { o $ED_0 = DC_{16}$}

$= [EC15 \oplus VM(DC_{15},SC_{16})] \oplus VM(DC_{15}, SC_{16})$, como o Ou-Exclusivo tem as seguintes propriedades:

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$

$$D \oplus D = 0$$

$$E \oplus 0 = E, \text{ então}$$

VI. $ED_1 = DC_{15}$ e $DD_1 = EC_{15}$, portanto a saída da primeira volta do processo de decodificação é $EC_{15} \,||\, DC_{15}$, que é a entrada trocada da 16ª volta do processo de criptografia.

VII. Generalizando, temos: $EC_i = DC_{i-1}$ e $DC_i = EC_{i-1} \oplus VM(DC_{i-1},K_i)$, ou

VIII. $DC_{i-1} = EC_i$, e $EC_{i-1} = DC_i \oplus VM(DC_{i-1},K_i) = DC_i \oplus VM(EC_i,K_i)$.

IX. Por fim, a saída da última volta do processo de decriptação é $DC_0 \parallel EC_0$. Trocando de lado as duas partes, recupera-se o texto claro.

4.2.6. Implementação

A implementação do PAPÍLIO foi feita na linguagem Delphi, versão 5.0. A figura-4.6 mostra a interface da implementação.

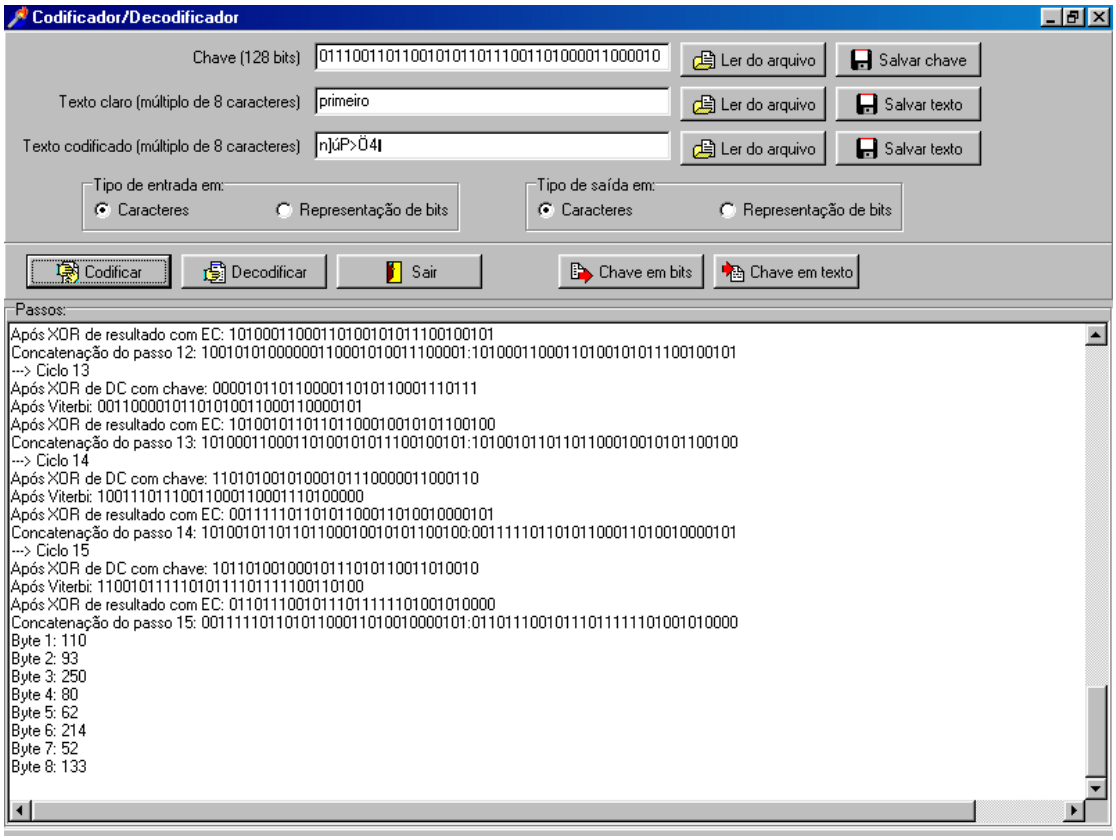

Figura-4.6: Interface do PAPÍLIO

- CHAVE

  A entrada da chave pode ser através de caracteres, porém os mesmos devem ser convertidos em bits. Como a chave só poderá ter 128 bits, então só poderão ser digitados 16 caracteres. O programa critica uma quantidade de bits inferior ou superior a 128. Para efeito de testes esta chave poderá ser salva.

- TEXTO CLARO

  O texto claro é dividido em blocos de 64 bits, ou seja, oito caracteres. A entrada do texto claro pode ser digitada na caixa de entrada ou lido a partir de um

arquivo texto. A entrada pode ser representada por caracteres ou fluxo de bits. O tipo de entrada deve ser selecionado: caractere ou em bits.

- TEXTO CODIFICADO

A saída será codificada. O bloco de saída é do mesmo tamanho do bloco de entrada, ou seja, 64 bits. Dependendo do tipo selecionado, o fluxo de saída poderá ser representado em caracteres ou bits.

- PASSOS

Os passos de geração de chave, bem como de codificação/decodificação, são exibidos na janela intitulada “passos”.

- PECULIARIDADES DA IMPLEMENTAÇÃO

As tabelas de conjuntos de caracteres podem representar até 256 caracteres (0 a 255). Em geral, os caracteres de 0 a 31 são reservados para caracteres de controle, os caracteres de 32 a 126 estão contidos no teclado, e os caracteres de 127 a 255 são caracteres estendidos. O caractere 0 não tem nenhuma representação, significa nulo. Portanto, durante a codificação ao surgir um byte 00000000, este byte, que não tem representação, omite um caractere. Porém o fluxo de bit da saída é constituído por 64 bits que será transmitido/armazenado, e conseqüentemente decodificado sem problema.

O sistema de criptografia proposto gera codificação para todo bloco de 8 caracteres (64-bits). No caso do bloco não possuir 8 caracteres, atribui-se espaços automaticamente de forma a codificar até mesmo um caractere.

Capítulo 5

Resultados Obtidos e Resistência a Ataques

5.1. Resultados Obtidos

É importante para um algoritmo de criptografia, conforme visto no capítulo 3 – seção 3.3 – que uma pequena mudança no texto claro ou na chave de codificação provoque uma mudança significante no texto codificado (Efeito Avalanche). Neste capítulo pretende-se mostrar, através de resultados de testes práticos, que o algoritmo proposto apresenta características desejáveis a qualquer algoritmo de criptografia. A seção 5.1.1 apresenta os resultados obtidos do algoritmo de codificação proposto quanto ao Efeito Avalanche.

Outros aspectos inerentes aos algoritmos de criptografia são os princípios de *difusão* e *confusão*. Como já foram mencionados no capítulo 3, esses princípios foram introduzidos por Claude Shannon [Shannon 49]. O interesse de Shannon pelos métodos de difusão e confusão era impedir a criptoanálise baseada em análise estatística. Assumindo-se que um intruso conheça as características estatísticas do texto claro, ou seja, a freqüência de distribuição de várias letras em uma mensagem escrita em um determinado idioma, ou ainda, palavras ou frases que provavelmente aparecem na mensagem. Se essas estatísticas de alguma forma refletem no texto codificado, o criptoanalista poderá descobrir a chave de criptografia, ou parte da chave, ou no mínimo um conjunto de chaves prováveis que contenha a chave exata. Segundo Robshaw [Robshaw 95], a difusão e a confusão constituem a pedra fundamental da concepção de codificadores de blocos modernos, uma vez que estes princípios capturam a essência dos atributos desejados de um codificador de blocos.

Na difusão, a estrutura estatística do texto claro é dissipada em estatística de longo alcance do texto codificado. Em outras palavras, a difusão tende a igualar a freqüência dos códigos do texto codificado, de forma a impedir que se analisando a estatística dos códigos descubra-se a letra no texto claro. A seção 5.1.2 exibe os resultados obtidos do algoritmo proposto com relação ao método de difusão.

O método da confusão busca tornar o mais complexo possível a relação entre a estatística do texto codificado e o valor da chave de criptografia, a fim de evitar a descoberta da chave. Admitindo-se que um intruso obtenha a estatística do texto codificado, a forma pela qual a chave foi utilizada para codificar o referido texto é tão complexa que dificulta a dedução da mesma. A seção 5.1.3 apresenta os resultados do algoritmo proposto em relação ao princípio de confusão.

5.1.1. Efeito Avalanche

Após a realização de vários testes, verificou-se que as tabelas geradas com o algoritmo Viterbi Modificado apresentam a propriedade desejada em qualquer algoritmo de criptografia: o efeito avalanche. A tabela-5.1 apresenta os resultados obtidos utilizando-se um texto claro de 8 caracteres, bloco de 64 bits, e uma chave de 128-bits. O texto claro teve um bit alterado, por vez, em todas as 64 posições, enquanto a chave permaneceu a mesma. A coluna BYTE da tabela-5.1 exibe os oito bytes do bloco de 64 bits; o caractere representado em bits é apresentado na coluna TEXTO CLARO, enquanto que a coluna BIT MODIFICADO DO BYTE exibe a posição do bit modificado e o caractere que passou a ser representado com a mudança do bit. A coluna BITS MODIFICADOS (BM) mostra o resultado do efeito avalanche. A média dos bits modificados por byte é apresentada na coluna MÉDIA BM P/BYTE.

Texto claro:	primeiro	(64 bits)
Texto codificado[2]:	ó2□Y¿□cº	(64bits)
Chave:	senha para teste	(128 bits)

[2] Os símbolos □ são distintos. Neste caso a representação em bit do primeiro é 00000001 e do segundo é 10011101

Byte	Texto Claro	Bit modificado do Byte	Bits Modificados (BM)	Média BM p/byte
1	Letra p (01110000)	ð (**1**1110000)	29	30,5
		0 (0**0**110000)	33	
		P (01**0**10000)	37	
		` (011**0**0000)	34	
		x (0111**1**000)	35	
		t (01110**1**00)	34	
		r (0111001**0**)	24	
		q (0111000**1**)	28	
2	Letra r (01110010)	ò (**1**1110010)	31	32
		2 (0**0**110010)	32	
		R (01**0**10010)	28	
		b (011**0**0010)	27	
		z (0111**1**010)	34	
		v (01110**1**10)	33	
		p (0111000**0**)	37	
		s (0111001**1**)	34	
3	Letra i (01101001)	é (**1**1101001)	29	33
		) (0**0**101001)	36	
		| (01**0**01001)	38	
		y (011**1**1001)	30	
		a (0110**0**001)	30	
		m (01101**1**01)	29	
		k (0110101**1**)	33	
		h (0110100**0**)	28	
4	Letra m (01101101)	í (**1**1101101)	31	29,5
		- (0**0**101101)	29	
		M(01**0**01101)	35	
		}(0111**1**101)	28	
		e (0110**0**101)	33	
		i (0110100**1**)	37	
		o (0110111**1**)	22	
		l (01101100)	33	
5	Letra e (01100101)	å (**1**1100101)	36	31,5
		% (0**0**100101)	31	
		E (01**0**00101)	33	
		u (011**1**0101)	30	
		m (0110**1**101)	40	
		a (0110**0**001)	31	
		g (0110011**1**)	23	
		d (0110010**0**)	35	
6	Letra i (01101001)	é (**1**1101001)	32	32
		) (0**0**101001)	36	
		I (01**0**01001)	35	
		y (011**1**1001)	28	
		a (0110**0**001)	39	
		m (01101**1**01)	32	
		k (0110101**1**)	31	
		h (0110100**0**)	25	

7	Letra r (01110010)	ò (**1**1110010)	36	33
		2 (0**0**110010)	30	
		R (01**0**10010)	31	
		b (011**0**0010)	32	
		z (0111**1**010)	30	
		v (01110**1**10)	31	
		p (0111000)	32	
		s (01110011)	31	
8	Letra o (01101111)	ï (**1**1101111)	24	29
		/ (0**0**101111)	29	
		O (01**0**01111)	21	
		□ (011**1**1111)	37	
		g (0110**0**111)	34	
		k (01101**0**11)	31	
		m (0110110**1**)	33	
		n (0110111**0**)	30	
Média Geral				31,56

Tabela-5.1: Resultados da codificação com mudança em um bit do texto claro e chave constante

Os resultados da tabela 5.1 mostram que se alterando apenas um bit em qualquer posição do bloco de 64 bits o efeito avalanche é alcançado. A substituição de um bit em certas posições poderá gerar um efeito avalanche maior ou menor. Por exemplo, a troca de um bit em uma dada posição do byte do caractere ***m*** provocou um efeito avalanche de 40 (quarenta) bits, enquanto que a troca de um bit em uma dada posição do byte do caractere ***o*** provocou um efeito avalanche de apenas 21 (vinte e um) bits. Conforme a definição de Heys e Tavares [Heys 95] o critério avalanche é satisfeito quando em média metade dos bits do texto codificado mudam quando um bit do texto claro é trocado. Portanto, verifica-se na tabela 5.1 que o algoritmo proposto apresenta um bom efeito avalanche, uma vez que em média metade dos bits do texto codificado mudaram quando um bit do texto claro foi alterado.

A propriedade do efeito avalanche também ocorreu quando um bit da chave foi alterado e o texto claro permaneceu constante. A tabela 5.2 apresenta os resultados do teste cuja chave foi alterada em 64 (sessenta e quatro) posições aleatórias (uma posição por vez), e o texto claro permaneceu inalterado. A coluna ALTERAÇÃO da tabela-5.2 ordena os testes realizados; a coluna POSIÇÃO DO BIT MODIFICADO NA CHAVE indica a posição do bit que foi alterado na chave, enquanto que a coluna BITS MODIFICADOS mostra o resultado do efeito avalanche.

Texto claro:	primeiro	(64 bits)
Texto codificado:	ó2□Y¿□cº	(64bits)
Chave:	senha para teste	(128 bits)

Alteração	Posição do Bit Modificado na Chave	Bits Modificados
1	3	35
2	4	37
3	5	36
4	6	34
5	8	32
6	9	32
7	12	30
8	13	33
9	14	24
10	16	36
11	17	33
12	19	26
13	20	32
14	24	34
15	25	36
16	29	32
17	33	33
18	35	33
19	36	32
20	37	31
21	38	33
22	39	34
23	41	33
24	42	25
25	43	32
26	44	26
27	48	28
28	53	32
29	56	39
30	59	29
31	60	27
32	62	37
33	63	32
34	64	35
35	65	36
36	67	30
37	70	30
38	71	31
39	72	26
40	77	35
41	79	30
42	80	35
43	82	30
44	83	35
45	84	33
46	86	24
47	87	26
48	88	35
49	89	33
50	90	38
51	92	28
52	94	30
53	95	32
54	104	35
55	105	33
56	107	31

57	110	40
58	111	34
59	112	39
60	113	34
61	116	32
62	123	31
63	125	30
64	126	33
Média Geral		32,22

Tabela-5.2: Resultados da codificação com mudança em um bit da chave e texto claro constante

Os resultados da tabela 5.2 constatam a segunda definição constante no capítulo 2, "um codificador satisfaz ao critério avalanche se, para cada chave, em média metade dos bits do texto codificado mudam quando um bit do texto claro ou da chave (K) é trocado", uma vez que mudanças aleatórias realizadas na chave, em apenas um bit por vez, geraram uma mudança no texto codificado, em média, de 32,22 bits.

O efeito avalanche do algoritmo de criptografia proposto foi analisado usando diversos testes com diversos textos claros e diferentes chaves. A média geral obtida nos testes girou em torno de 32 bits modificados a cada troca de bit, seja do texto claro ou da chave.

5.1.2. Difusão

Em geral, a maneira mais simples de se obter difusão é através da permutação [Schneier 96]. Segundo Stallings [Stallings 98], a permutação – por si só – não modifica a estatística do texto claro em relação às letras individuais ou a blocos permutados. A difusão do algoritmo proposto é atingida através da estrutura da função F (figura 5.1) que é constituída da operação OU-exclusivo, do algoritmo Viterbi Modificado e da concatenação dos fluxos gerados pelo algoritmo Viterbi Modificado. A estrutura F tem como entrada dois valores de 32 bits, sendo um derivado do texto claro e outro da chave, e produz uma saída de 32 bits. Essa estrutura é repetida dezesseis vezes no algoritmo, a fim de fornecer uma difusão efetiva.

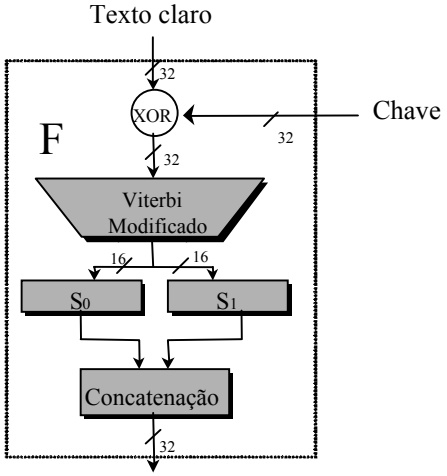

Figura 5.1: Estrutura da Função F

Como foi abordado no início desta seção, na difusão, cada bit do texto claro deve influenciar em vários bits do texto codificado. O espalhamento de um simples bit sobre muitos bits do texto codificado oculta a estrutura estatística do texto claro. A fim de constatar a difusão do algoritmo de criptografia proposto, foram realizados vários testes. Abaixo está um dos textos utilizados para teste [Stallings 98]:

> *"it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in moscow"*

O texto em referência após criptografado apresentou-se conforme segue [3]:

q:C'Ï¸□yM÷ôO—Ý§□□›âÓ…†Tzû□…ütÜ3AW□rŠIöÞ□õÔzÉ
¼ÇVcG□7· □:øMCÑAüyµ□Sò□ÈÏR®□?ŽŒšý$ u[□¢'°¼Œ+l*f*
"□¼qPq^□D□¸□m×)□»*ÖTÆš%:^©¤‰□*8,□šçQ9k6],„uÄ□9
^7¡í†²!¶²

Antes de analisar a difusão no texto em referência, é importante comentar sobre dois atributos particularmente importantes do princípio de difusão: freqüência das letras em um idioma e a redundância [Robshaw 95]. O conjunto de todos os caracteres do texto claro é conhecido como o alfabeto, assim como o conjunto de letras de uma linguagem natural também é conhecido como alfabeto. A freqüência pela qual as

[3] Duas ocorrências do símbolo □ não representam necessariamente o mesmo byte.

letras/caracteres ocorrem em uma grande coleção de texto claro é distinta, não é uniforme. Por exemplo, na língua inglesa a letra “e” é a letra mais comum de ocorrer (vide tabela 5.3). Uma análise da distribuição dos caracteres em um texto codificado pode revelar informações tanto do texto claro, quanto da chave, caso o codificador não seja suficientemente bom para destruir a informação da freqüência das letras.

Letra	Freqüência
A	8.167
B	1.492
C	2.782
D	4.253
E	12.702
F	2.228
G	2.015
H	6.094
I	6.996
J	0.153
K	0.772
L	4.025
M	2.406
N	6.749
O	7.507
P	1.929
Q	0.095
R	5.987
S	6.327
T	9.056
U	2.758
V	0.978
W	2.360
X	0.150
Y	1.974
Z	0.074

Tabela 5.3: Freqüência relativa na língua inglesa [Beker 82].

Através da redundância de uma língua é possível compreender uma sentença, sem que seja necessário ver o texto completo. Por exemplo, uma sentença escrita na língua inglesa pode omitir a letra “u”, que vem imediatamente após a letra “q”, sem prejudicar a compreensão da sentença. O par de letras “q” e “u” no idioma inglês é um exemplo de redundância, e um bom codificador deve garantir que o conhecimento da redundância do texto claro pouco influenciará na decriptação da mensagem.

Analisando os textos claro e codificado, em referência, observa-se que a variedade de caracteres do texto codificado é superior a variedade de caracteres do texto claro. Pode-se constatar, por exemplo, que “e“ é a letra de maior ocorrência (16) no texto claro. Entretanto, no texto codificado esta letra não ocorreu. Cada uma das dezesseis ocorrências da letra “e” do texto claro foi substituída por (dezesseis)

caracteres distintos (§,â,†,W,r,M,?,š,ý,¢,,,×,□,%,8,ç), destruindo completamente a informação da freqüência da letra.

No texto codificado não se observa a repetição de dois caracteres juntos, o que poderia representar uma redundância. No texto claro ocorre o digrama “th” três vezes (that, with, the), entretanto no texto codificado esse digrama é representado por …ü, Œ+, -* o que mais uma vez constata a difusão do algoritmo proposto.

A tabela 5.4 exibe a freqüência dos caracteres no texto claro, e a freqüência dos mesmos caracteres no texto codificado, por motivo de simplicidade a freqüência dos demais caracteres originados no texto codificado foram omitidos, porém a tabela 5.4 completa está inserida nos anexos deste trabalho.

Caracteres	Freqüência	
	Texto Claro	Texto Codificado
a	10	-
b	2	-
c	7	1
d	5	-
e	16	-
f	2	-
g	1	-
h	4	-
i	10	-
j	-	-
k	-	1
l	5	1
m	3	1
n	6	-
o	8	-
p	2	-
q	-	3
r	6	1
s	9	-
t	14	1
u	1	2
v	4	-
w	3	-
x	-	-
y	2	2
z	-	2
espaço	24	2

Tabela 5.4: Freqüência dos Caracteres e Códigos de um Texto Claro e Codificado.

Observa-se na tabela 5.4 que na coluna texto claro existem vários caracteres com freqüência superior a 8, enquanto que na coluna texto codificado a maior freqüência é 3. Uma vez dissipada a estatística de freqüência dos caracteres do texto claro no texto

codificado, pode-se constatar que o algoritmo PAPÍLIO dispõe da propriedade de difusão.

5.1.3. Confusão

O princípio da confusão busca tornar o mais complexo possível a relação entre a chave e o texto codificado. A confusão do algoritmo proposto é realizada através das tabelas geradas a partir do algoritmo Viterbi Modificado, as quais são utilizadas tanto na geração das subchaves, como na codificação dos dados.

Assim como o princípio da difusão sugere que cada bit do texto claro deve influenciar vários bits do texto codificado, a influência dos bits da chave deve espalhar-se sobre os bits do texto codificado. A tabela 5.5 exibe os resultados da codificação do texto da seção anterior, cuja chave foi alterada em um bit. A tabela 5.5 com a freqüência de todos os caracteres originados no texto codificado está inserida nos anexos deste trabalho.

Caracteres	Freqüência	
	Texto Claro	Texto Codificado
a	10	-
b	2	-
c	7	-
d	5	1
e	16	-
f	2	-
g	1	2
h	4	-
i	10	-
j	-	-
k		-
l	5	1
m	3	-
n	6	-
o	8	-
p	2	1
q	-	-
r	6	-
s	9	2
t	14	1
u	1	-
v	4	-
w	3	-
x	-	-
y	2	-
z	-	-
espaço	24	-

Tabela 5.5: Freqüência dos Caracteres dos Textos Claro e Codificado Referente à Alteração na chave.

Comparando-se as tabelas 5.4 e 5.5, observa-se que as freqüências dos caracteres das colunas TEXTO CODIFICADO, exibem valores diferenciados, entretanto ambas mantêm a dissipação da estatística de freqüência dos caracteres do texto claro. Observa-se na tabela 5.5 completa, localizada nos anexos, que, se mantendo o texto claro constante e alterando-se a chave em apenas um bit, novos códigos foram gerados em relação à tabela 5.4 completa, e que, entretanto, as freqüências dos códigos da coluna TEXTO CODIFICADO mantiveram-se relativamente constante. O caractere ***u***, por exemplo, ocorre no texto claro uma vez, porém na tabela 5.4 coluna TEXTO CODIFICADO o mesmo ocorre duas vezes, enquanto que na tabela 5.5 o mesmo não ocorre .

5.2. Resistência a Ataques

Os ataques criptoanalíticos, como mencionado no capítulo – 2, seção 2.4, são classificados de acordo com o tipo de informação que está disponível para o criptoanalista. Assume-se que o criptoanalista tem total conhecimento sobre o algoritmo que está sendo utilizado. Desta forma a segurança do sistema está vinculada totalmente a chave do sistema, a qual deve ser mantida em segredo. Quando o criptoanalista conhece os algoritmos de criptação e decriptação, pode tentar descobrir a chave através de dois métodos: tentar exaustivamente possíveis candidatos para ser a chave, ou, quando o criptoanalista conhece a natureza do texto claro poderá explorar as regularidades da linguagem, através de uma análise estatística. A seguir serão vistas estas duas possibilidades em relação ao PAPÍLIO.

O ataque mais comum a um codificador de bloco é aquele que testa exaustivamente cada possível candidato para ser a chave k . A dificuldade de um ataque pela força bruta é exponencial com relação ao tamanho da chave: incluindo-se um bit na chave, o ataque pela força bruta terá o dobro de dificuldade; incluindo-se dois bits o ataque será quatro vezes mais difícil [Schneier 00]. Se a chave tem um comprimento de c bits, então haverá 2^c possíveis chaves, embora se espere encontrar a chave com 2^{c-1} tentativas. Entretanto, o tamanho da chave torna-se irrelevante quando o algoritmo criptográfico apresenta outros pontos fracos, que permitem a dedução da chave.

Com base na tabela 5.6 [Stallings 98] o algoritmo PAPÍLIO tem uma quantidade de alternativas de chaves na ordem de 2^{128} ($3{,}4 \times 10^{38}$). Considerando que uma máquina leve 1μs (microsegundo) para executar uma decriptação, seriam necessários 5.4×10^{24} anos para testar a metade (2^{127}) do espaço de chave possível. Através dos multicomputadores fortemente paralelos pode-se conseguir taxas de processamento muito maiores. Considerando um sistema que pode processar um milhão de chaves por microsegundo (μs), o tempo necessário para descobrir a chave do algoritmo proposto seria na ordem de $5{,}4 \times 10^{18}$ anos, que é quase o dobro da idade do universo (10^{10} anos) [López 99].

Tamanho da Chave (bits)	Quantidade de chaves alternativas	Tempo necessário de 1 codificação/μs	Tempo necessário de 10^6 codificação/μs
32	$2^{32} = 4{,}3 \times 10^9$	2^{31}μs = 35,8 minutos	2,15 ms
56	$2^{56} = 7{,}2 \times 10^{16}$	2^{55}μs = 1142 anos	10,01 horas
128	$2^{128} = 3{,}4 \times 10^{38}$	2^{127}μs = $5{,}4 \times 10^{24}$ anos	$5{,}4 \times 10^{18}$ anos

Tabela 5.6: Tempo médio necessário para busca exaustiva da chave

Conforme comentado anteriormente, um outro método que caracteriza o uso da força bruta é o fato do criptoanalista conhecer a natureza do texto claro e tentar explorar as regularidades da linguagem. Este tipo de método sugere que se determine a freqüência dos caracteres do texto codificado, a fim de compara-las com a freqüência relativa das letras de uma dada linguagem.

O criptoanalista poderá relacionar os códigos de menor freqüência no texto codificado com as letras cuja freqüência relativa na linguagem seja baixa, ou o contrário. O criptoanalista poderá, ainda, analisar seqüências repetidas de caracteres do texto codificado e tentar deduzir o texto claro equivalente.

Uma análise estatística, conforme descrito acima, não obtém sucesso quando o algoritmo PAPÍLIO é utilizado. Devido às propriedades de difusão, confusão e efeito avalanche que o algoritmo PAPÍLIO dispõe, torna-se muito difícil recuperar o texto claro mediante aplicação de uma análise estatística, uma vez que o referido algoritmo dissipa toda a freqüência das letras do texto claro (vide tabelas 5.4 e 5.5).

Capítulo 6

Conclusão e Trabalhos Futuros

Esta dissertação apresentou uma proposta de algoritmo de criptografia denominado PAPÍLIO, cujo processo de codificação foi baseado no algoritmo Viterbi. PAPÍLIO é um codificador de bloco (criptografa dados em blocos de 8 bytes) simétrico (usa a mesma chave secreta para ambos criptação e decriptação) que usa uma chave de 128 bits e foi concebido para codificar qualquer tipo de dado através dos modos de operação apresentados na seção 2.5 desta dissertação. A importância desta proposta de dissertação não se concentrou apenas nas características do algoritmo PAPÍLIO, o qual foi desenvolvido a partir da estrutura do Codificador Feistel, mas principalmente da utilização do algoritmo Viterbi como parte fundamental de um algoritmo de criptografia.

Com a proposta e utilização do algoritmo Viterbi Modificado apresentada nesta dissertação, mostrou-se que é possível desenvolver algoritmos cujas tabelas de codificação não necessitam ser ocultadas. Como se sabe um bom princípio em criptografia é considerar que nada é secreto senão a chave [Stallings 98],[Schneier 00]. O algoritmo PAPÍLIO aqui proposto apresenta esta característica pois, como foi mostrado no capítulo 4 o conhecimento das tabelas em nada ajuda o processo de decriptação.

Mostrou-se também que a utilização do algoritmo Viterbi Modificado para a criptografia é vantajosa quanto à simplicidade de geração das tabelas de codificação. Apesar desta simplicidade foi possível mostrar que tais tabelas garantem propriedades tais como, efeito avalanche, difusão e confusão, desejáveis em qualquer algoritmo de criptografia.

A simplicidade do algoritmo Viterbi Modificado para a geração das tabelas de codificação permite também a modificação do Algoritmo PAPÍLIO para a utilização de chaves de mais de 128 bits ou de blocos de mais de 64 bits.

Finalmente, uma das maiores contribuições da proposta de algoritmo de criptografia aqui apresentada consiste na utilização de uma modificação simples (Viterbi Modificado) de um algoritmo usado na correção antecipada de erro (Viterbi). Tal característica permite a utilização de recursos software ou hardware comuns para os processos de criptografia e codificação para transmissão com correção de erro. Como se sabe em criptografia, a alteração de um único bit do texto codificado pode impedir a decriptação correta da informação. Com a utilização de PAPÍLIO é, portanto, possível, em um ambiente com ruídos, usar menos recursos software ou hardware para os processos de criptografia e codificação para transmissão.

O algoritmo PAPÍLIO apresenta-se atualmente no modo livro de código eletrônico (Eletronic Codebook Mode – ECB), entretanto, como proposta de trabalhos futuros pretende-se adequá-lo aos modos: codificador de encadeamento de bloco (Cipher Block Chaining Mode – CBC), codificador realimentado (Cipher Feedback Mode – CFB) e de saída com realimentação (Output Feedback Mode – OFB).

A fim de dotar o algoritmo PAPÍLIO de maior flexibilidade pretende-se ampliar a faixa de bits da chave, bem como tornar variáveis os tamanhos dos blocos de texto claro e texto codificado.

Geralmente, as inovações na área de criptografia são propostas por trabalhos acadêmicos. Esta dissertação teve como objetivo maior demonstrar a potencialidade do uso do algoritmo Viterbi Modificado como função F do codificador Feistel. Evidentemente que o algoritmo proposto (PAPÍLIO) ainda pode ser aprimorado pois, até que um algoritmo seja considerado realmente seguro, é necessário que o mesmo seja examinado por muitos especialistas no decorrer de alguns anos [Schneier 00]. PAPÍLIO surge assim como uma nova proposta de algoritmo cujas potencialidades e eventuais defeitos, não foram completamente explorados nesta dissertação. O primeiro passo foi dado, entretanto, ainda existe um grande caminho a ser percorrido.

Bibliografia

[Anderson 98] Anderson, R., Biham, E., Knudsen, L. “Serpent: A Proposal for the Advanced Encryption Standard”. June 1998. URL: http://www.cl.cam.ac.uk/~rj14/serpent.html (03.10.2000).

[Barreto 99] Barreto, Paulo S.L.M. Curvas Elípticas e Criptografia: Conceitos e Algoritmos. Junho 1999.

[Biham 93] Biham, E., Shamir, A. Differential Cryptanalysis of the Data Encryption Standard. New-York: Springer-Verlag, 1993.

[Blaze 96] Blaze, M., Diffie, W., Rivest, R. L., Schneier, B., Shimomura, T., Thompson, E., and Wiener, M.; “Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security”, A Report by na Ad Hoc Group of Cryptographers and Computer Scientists, January 1996.

[Beker 82] Beker, H.; Piper, F.; Cipher Systems. Van Nostrand, London, 1982.

[Burwick 99] Burwick, C., Coppersmith, D., et al. “MARS – a candidate cipher for AES”, august 1999. URL: http://www.research.ibm.com/ security/mars.html (03.10.2000)

[Daemen 95] Daemen, J., Knudsen, L. R., Rijmen, V. The Block Cipher Square Algoritm. Fast Software Encryption. Springer-Verlag, 1995.

[Daemen 99] Daemen, J., Rijmen, V. “AES Proposal: Rijndael”, version 2. March 1999. URL: http://www.esat.kuleuven.ac.be/~rijmen/ rijndael (03.10.2000)

[Daemen 01] Daemen, J., Rijmen, V. “Rijndael, the advanced encryption standard”. Dr. Dobb’s Journal, Vol.26, Nr. 3, March 2001, pp. 137 – 139.

[Demirbas 81] Demirbas, K. Target Tracking in the Presence of Interference, Ph.D. dissertation, University of California, Los Angeles, 1981.

[Diffie 76] Diffie W., Hellman, M.E.; New directions in cryptography, IEEE Transactions on Information Theory 22 (1976), 644-654.

[Feistel 73] Feistel, H.; "Cryptography and Computer Privacy". Scientific American, May 1973.

[Feygin 96a] Feygin, G.; Glenn; Survivor Sequence Memory Management in Viterbi Decoders. Department of Electrical Engineering, University of Toronto, Canada, July, 1996.

[Feygin 96b] Feygin, G.; Gulak, P.G.; Generalized Cascade Viterbi Decoder. Department of Electrical Engineering, University of Toronto, Canada, July, 1996.

[FIPS 80] Federal Information Processing Standards Publication 81. "DES Modes of Operation". December 1980. URL: http://www.itl.nist.gov/fipspubs/fip81.htm (09.09.2001)

[Fleming 99] Fleming, Chip. "A Tutorial on Convolutional Coding with Viterbi Decoding". Spectrum Applications, July, 1999.

[Flood 00] Flood, Kathleen. "SANS Reading Room: The Advanced Encryption Standard: A Review of the Finalists". Outubro 2000. URL:http://www.sans.org/infosecFAQ/encryption/encryption.htm (09.09.2001)

[Forney 74] Forney, G. David, Jr. Convolutional Codes II. Maximum-Likehood Decoding, Information and Control, July 1974; 25 (3): pp. 177-179

[Giulietti 98] Giulietti, A.; Strum, M.; Aardoom, E.; Gyselinckx, B.; A 250 kb/s, k=7, 3-bit soft decision programable code rate customized Viterbi decoder. IV Workshop Iberchip, Mar del Plata, Argentina, Março, 1998.

[Heys 95] Heys, H., and Tavares, S. ; “Avalanche Characteristcs of Substitution-Permutation Encryption Networks”. IEEE Transactions on Computers, September 1995.

[Koblitz 87] Koblitz, N. Elliptic Curve Cryptosystems. Mathematics of Computation, v. 48, n. 177, 1987, pp.203-209.

[Lai 90] Lai, X., Massey, J.; A proposal for a New Block Encryption Standard. In Proceedings of the EUROCRYPT 90 Conference, pp. 389-404,1990

[Lai 91] Lai, X., Massey, J.; Markov Ciphers and Differential Cryptanalysis”. Proceedings, EUROCRYPT 91, 1991; published by Springer-Verlag.

[Lai 92] Lai, X. On the Design and Security of Block Ciphers. Konstanz, Germany: Hartung-Gorre, 1992.

[Leveque 90] Leveque, W. Elementary Theory of Numbers. New York: Dover 1990.

[Lasertogo 97] Caixão-de-Defunto, Espia-Só, Papílio – Petrobrás.Disponível em: http://www.petrobras.com.br/portugue/acompanh/salaesta/posters/borbolet/borb14.htm. Acesso em: 17.01.2002

[López 99] López, M. J. Lucena. Criptografía y Seguridad en Computadores. 2ª ed., septiembre 1999.

[Matsui 93] Matsui, M. Linear Cryptanalysis Method for DES Cipher. Proceedings, EUROCRYPT’93, 1993; published by Springer-Verlag.

[Miller 86] Miller, V. S. Use of Elliptic Curves in Cryptography. Advances in Cryptology – CRYPTO’85 Proceedings, Springer-Verlag 1986, pp. 417-426.

[Menezes 93] Menezes, A. Eliptic Curve Public Key Criptosystems, Kluwer Academic Publishers, 1993

[Menezes 96] Menezes, A., Oorschot, P. van, Vanstone S.; Handbook of Applied Cryptography. CRC Press, 1996.

[NBS 77] National Bureau of Standards, NBS FIPS PUB 46. Data Encryption Standard. National Bureau of Standards, U. S. Department of Commerce, January 1977

[NIST 97] National Institute of Standards and Technology, U.S. Department of Commerce. “Annoucing Request for Candidate Algorithm Nominations for the Advanced Encryption Standard (AES)”. Federal Register, Volume 62, Number 177. 12 september 1997. URL:http://csrc.nist.gov/encryption/aes/pre-round1/aes_9709.htm (09.09.2001)

[Omura 69] Omura, J. K. On the Viterbi Decoding Algorithm, IEEE transactions on information Theory, January 1969; IT – 15 (1): pp. 177-179

[Ore 76] Ore, O. Number Theory and its History. New York: Dover 1976.

[Pitas 87] Pitas, I. A Viterbi algorithm for image edge detection. Proc. of the eleventh International Conference GRETSI on Digital Signal Processing, Nice, France, 1987.

[Pitas 89] Pitas, I. A Viterbi algorithm for region segmentation and edge detection. Proc. CAIP89, Leipzig, pp. 129-133, 1989.

[Rivest 78] Rivest, R.; Shamir, A.; and Adleman, L. “A Method for Obtaining Digital Signatures and Public Key Cryptosystems.” Communications of the ACM, February 1978.

[Rivest 94] Rivest, R.; The RC5 Encryption Algorithm. Proceedings, Second International Workshop on Fast Software Encryption, December 1994; published by Springer-Verlag.

[Rivest 95] Rivest, R.; The RC5 Encryption Algorithm. Dr. Dobb’s Journal, January 1995.

[Rivest 98] Rivest, R., Robshaw, M., et al. “The RC6[tm] Block Cipher”. june 1998. URL: http://www.rsasecurity.com/rsalabs/aes/ index.html (03.10.2000)

[Ryan 93] Ryan, M.S.; Nudd, G. R.; The Viterbi Algorithm. Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England, February, 1993.

[Robshaw 95] Robshaw, M.; Block Ciphers. RSA Laboratories Technical Report TR-601, August, 1995.

[Schneier 93] Decription of a New Variable-Lenght Key, 64-bit Block Cipher (Blowfish). Cambridge Security Workshop Proceedings, Springer Verlag, pages 191- 204, december 1993.

[Schneier 96] Schneier, B. Applied Cryptography. New York: Wiley, 1996.

[Schneier 98] Schneier, B., Kelsey, J., et al. "Twofish: A 128-Bit Block Cipher". June 1998. URL: http//:www.counterpane.com/twofish-paper.html (03.10.2000)

[Schneier 01] Schneier, Bruce. Segurança.com: segredos e mentiras sobre a proteção na vida digital; tradução de Daniel Vieira. Rio de Janeiro: Campus, 2001.

[Shannon 49] Shannon, C. E.; "Communication Theory of Secrecy Systems". Bell Systems Technical Journal, Nº 4, 1949.

[Stallings 98] Stallings, William; Cryptography and Network Security: Principles and Practice. 2ª Ed., Prentice Hall,1998.

[Viterbi 67] Viterbi, A. J., Error Bounds for Convolutional Codes and na Asymptotically Optimum Decoding Algorithm, IEEE Transactions on Information Theory, April 1967; IT – 13 (2): pp. 260-269.

[Viterbi 79] Viterbi, Andrew J., Omura, Jim. K.; Principles of digital communication and coding. McGraw-Hill, Inc. 1979.

Anexos

Tabela 5.4 – Freqüência dos caracteres e códigos de um texto claro e codificado.

Ordem	Caracteres	Freqüência	
		Texto Claro	Texto Codificado
1	a	10	-
2	b	2	-
3	c	7	1
4	d	5	-
5	e	16	-
6	f	2	-
7	g	1	-
8	h	4	-
9	i	10	-
10	j	-	-
11	k	-	1
12	l	5	1
13	m	3	1
14	n	6	-
15	o	8	-
16	p	2	-
17	q	-	3
18	r	6	1
19	s	9	-
20	t	14	1
21	u	1	2
22	v	4	-
23	w	3	-
24	x	-	-
25	y	2	2
26	z	-	2
27	espaço	24	2
28	A	-	2
29	C	-	2
30	D	-	1
31	G	-	1
32	I	-	1
33	M	-	2
34	O	-	1
35	P	-	1
36	Q	-	1
37	R	-	1
38	S	-	1
39	T	-	2
40	V	-	1
41	W	-	1
42	:	-	3
43	’	-	2
44	Ï	-	2
45	,	-	2
46	÷	-	1
47	ô	-	1
48	Ô		1

49	—	-	1
50	Ý	-	1
51	ý	-	1
52	§	-	1
53	□(10001111)	-	1
54	□(10010000)	-	1
55	›	-	1
56	â	-	1
57	Ó	-	1
58	…	-	1
59	†	-	2
60	□(00000000)	-	1
61	û	-	1
62	□(10011101)	-	1
63	ü	-	2
64	Ü	-	1
65	3	-	1
66	Š	-	4
67	ö	-	1
68	Ö	-	1
69	Þ	-	1
70	õ	-	1
71	É	-	1
72	¼	-	3
73	ç	-	1
74	Ç	-	1
75	7	-	2
76	·	-	1
77	qpag	-	2
78	ø	-	1
79	Ñ	-	1
80	µ	-	1
81	enter	-	1
82	□(00100000)	-	13
83	ò	-	1
84	È	-	1
85	®	-	1
86	?	-	1
87	Ž	-	1
88	Œ	-	2
89	$	-	1
90	[	-	1
91	¢	-	1
92	°	-	1
93	+	-	1
94	*ƒ*	-	1
95	“	-	1
96	^	-	1
97	×	-	1
98	)	-	1
99	»	-	1
100	*	-	2
101	Æ	-	1
102	%	-	1
103	ˆ	-	2
104	©	-	1
105	¤	-	1
106	‰	-	1

107	-	-	1
108	8	-	1
109	¸	-	1
110	9	-	2
111	6	-	1
112	]	-	1
113	„	-	1
114	Ä	-	1
115	¡	-	1
116	í	-	1
117	²	-	2
118	!	-	1
119	¶	-	1
Total		144	144

Tabela 5.5: Freqüência dos caracteres dos textos claro e codificado referente à alteração na chave.

Ordem	Caracteres	Freqüência	
		Texto Claro	Texto Codificado
1	a	10	-
2	b	2	-
3	c	7	-
4	d	5	1
5	e	16	-
6	f	2	-
7	g	1	2
8	h	4	-
9	i	10	-
10	j	-	-
11	k		-
12	l	5	1
13	m	3	-
14	n	6	-
15	o	8	-
16	p	2	1
17	q	-	-
18	r	6	-
19	s	9	2
20	t	14	1
21	u	1	-
22	v	4	-
23	w	3	-
24	x	-	-
25	y	2	-
26	z	-	-
27	espaço	24	-
28	A	-	2
29	C	-	2
30	E	-	1
31	F	-	1
32	I	-	1
33	N	-	1
34	R	-	1
35	T	-	1
36	V	-	1
37	X	-	1

38	z	-	2
39	espaço	-	4
40	’	-	4
41	Ï	-	1
42	¸	-	1
43	ô	-	3
44	□(10010000)	-	1
45	›	-	1
46	Û	-	1
47	□(10011101)	-	1
48	Ü	-	1
49	3	-	2
50	ö	-	1
51	Þ	-	1
52	õ	-	1
53	Õ	-	1
54	é	-	2
55	¼	-	1
56	7	-	2
57	·	-	1
58	qpag	-	2
59	ø (Ø)	-	3
60	Ñ	-	2
61	µ	-	1
62	□(00100000)	-	12
63	ò	-	1
64	?	-	1
65	ž	-	2
66	$	-	1
67	¢	-	1
68	+	-	1
69	×	-	1
70	)	-	2
71	*	-	3
72	¤	-	1
73	Ž	-	1
74	-	-	3
75	¸	-	1
76	6	-	1
77	]	-	2
78	Ä	-	2
79	í	-	1
80	!	-	1
81	¶	-	1
82	™	-	1
83	¹	-	3
84	‹	-	1
85	2	-	1
86	¾	-	1
87	{	-	4
88	À	-	2
89	Ú	-	2
90	¥	-	1
91	º	-	1
92	´	-	2
93	ª	-	1
94	î	-	2
95	ê	-	2

96	¨	-	1
97	¯	-	1
98	Ã	-	1
99	ÿ	-	1
100	£	-	1
101	Ð	-	1
102	å	-	1
103	”	-	1
104	□(10001101)	-	1
105	□(01111111)	-	1
106	Ë	-	1
107	&	-	1
108	Á	-	1
109	□(10000001)	-	1
110	¿	-	1
111	#	-	1
112	ù	-	3
113	¸	-	1
114	¯	-	1
Total	-	144	144