Universidade Federal do Rio Grande do Norte

CCET: Centro de Ciências Exatas e da Terra

CT: Centro de Tecnologia

DCA: Departamento de Computação e Automação

Curso: Engenharia de Computação

Disciplina: Relatório de Graduação

Professores

Benjamín Callejas Bedregal

Ivan Saraiva Silva

Aluno

Frederiko Stenio L. N. de Araújo

PapílioXP - Uma Extensão do Algoritmo Criptográfico Papílio

NATAL

29-07-2003

FREDERIKO S. L. N. DE ARAÚJO

PAPÍLIOXP - UMA EXTENSÃO DO ALGORITMO CRIPTOGRÁFICO PAPÍLIO

NATAL

29-07-2003

FREDERIKO S. L. N. DE ARAÚJO

PAPÍLIOXP - UMA EXTENSÃO DO ALGORITMO CRIPTOGRÁFICO PAPÍLIO

Relatório de Graduação do curso de Engenharia de Computação da disciplina Relatório de Graduação, como parte dos requisitos para a obtenção do título de Engenheiro de Computação pela Universidade Federal do Rio Grande do Norte.

Orientador: Benjamín Callejas Bedregal
Co-Orientador: Ivan Saraiva Silva

NATAL
29-07-2003

FREDERIKO S. L. N. DE ARAÚJO

PAPÍLIOXP - UMA EXTENSÃO DO ALGORITMO CRIPTOGRÁFICO PAPÍLIO

Relatório de Graduação do curso de Engenharia de Computação da disciplina Relatório de Graduação, como parte dos requisitos para a obtenção do título de Engenheiro de Computação pela Universidade Federal do Rio Grande do Norte.

Aprovado em 4 de agosto de 2003

BANCA EXAMINADORA

_____

Prof. BENJAMÍN CALLEJAS BEDREGAL

Orientador

_____

Prof[a]. THAÍS VASCONCELOS BATISTA

_____

Msc. KARLA DARLENE NEPOMUCENO RAMOS

Universidade Federal do Rio Grande do Norte

SUMÁRIO

Índice de Figuras

Índice de Tabelas

Resumo do Relatório de Graduação do curso de Engenharia de Computação da disciplina Relatório de Graduação, como parte dos requisitos para a obtenção do título de Engenheiro de Computação pela Universidade Federal do Rio Grande do Norte.

FREDERIKO S. L. N. DE ARAÚJO

PAPÍLIOXP - UMA EXTENSÃO DO ALGORITMO CRIPTOGRÁFICO PAPÍLIO

Orientador: Benjamín Callejas Bedregal
Co-Orientador: Ivan Saraiva Silva

Desde as eras mais remotas, o homem sempre teve a necessidade de guardar sigilo em determinados contextos: segredo de estado, contrabando de informação, manobras militares, transações comerciais, etc. Nestas situações são utilizadas técnicas criptográficas como a esteganografia, utilização de códigos e a composição de cifras, afim de que a informação de caráter sigiloso fosse protegida. No século XXI o uso da criptografia é mais presente e difundido, devido ao uso extensivo que fazemos dos meios de comunicação: telefonia celular, comércio eletrônico, assinatura de serviços oferecidos via satelite, etc. No ano de 2002 um grupo de estudiosos do Departamento de Informática e Matemática Aplicada (DIMAp) da Universidade Federal do Rio Grande do Norte (UFRN), Natal, Brasil, propuseram um algoritmo Criptográfico simétrico em blocos que usa Rede de Feistel para codificação dos blocos. Este

algoritmo tem o nome de Papílio e é baseado nos princípios da decodificação Viterbi e na codificação Convolucional, processando blocos de comprimento 64 bits e trabalhando com chaves criptográficas de comprimento 128 bits. O presente trabalho propõe uma extensão ao algoritmo Papílio de forma a adicionar-lhe a capacidade de variar o modo como cifrar os textos dependendo do texto o qual se está cifrando e de maneira que atinja os melhores índices criptográficos possíveis, afim de que seja aumentada a dificuldade de se conseguir quebrar a mensagem codificada. Para validar esta proposta de extensão feita ao Papílio foi construído um grupo de ferramentas para a análise de textos criptografados dentre as quais se destaca um bench-mark chamado "mortalKombat", para se fazer o comparativo entre algoritmos criptográficos simétricos. Os índices usados para se medir a qualidade dos textos cifrados foram; avalanche, difusão e confusão. O algoritmo resultante das extensões feitas ao Papílio recebe o nome de PapílioXP (Papílio eXtenso). Foi realizado um estudo comparativo entre o PapílioXP e o padrão criptográfico da atualidade o Rijndael-AES (Advanced Encription Standard), o qual em nossas análises iniciais se mostra equivalente ao PapílioXP. Os testes foram executados para os 4 modos de operação: ECB, CBC, CFB, OFB na configuração de chave com comprimento de 128 bits e bloco com comprimento de 128 bits para o Rijndael. Para o PapílioXP a chave usada foi de comprimento 128 bits e bloco de comprimento 64 bits. Outro estudo comparativo é feito entre o PapílioXP e o Papílio original, agora chamado de Papílio Tradicional, para constatar se de fato houve ganho de desempenho no processo de cifra. Este estudo mostrou que a melhoria da qualidade foi sensível.

Capítulo 1 – Introdução

1.1. Introdução

O Papílio é um algoritmo criptográfico simétrico com chave de comprimento 128 bits que codifica blocos de 64 bits baseado em rede de Feistel que usa como função de substituição o Viterbi Modificado. A função Viterbi Modificado é usada para a codificação de blocos na rede de Feistel, cuja base é a Codificação Convolucional e a Decodificação Viterbi. Papílio é fruto do trabalho da dissertação de mestrado [RAM] defendida na UFRN. O presente trabalho é uma extensão dos estudos iniciados em [RAM], apresentando mudanças que conferem ao Papílio a capacidade de variar a forma com que codifica um texto em função dele mesmo e não apenas da chave criptográfica. O trabalho aborda no capítulo II algumas técnicas básicas de criptografia (cifra, esteganografia e códigos), capítulo III, fundamentos da teoria do Papílio (Rede de Feistel, códigos Convolucionais, Decodificação Viterbi), é definida a função de substituição usada na Rede de Feistel; o Viterbi Modificado, fala-se sobre modos de operação dos algoritmos Criptográficos, ECB, CBC, CFB e OFB e por fim é apresentado o algoritmo Papílio. No capítulo IV são apresentados os argumentos que justificam uma otimização para ganho de performance criptográfica no algoritmo Papílio. Em seguida são feitas análises que comprovam os efeitos da otimização. No capítulo V são apresentadas modificações para conferir ao Papílio maior versatilidade no processo de codificação. O algoritmo Papílio estendido recebe o nome de: PapílioXP (Papílio eXtenso). E por fim no capítulo VI são mostrados índices que medem a qualidade dos algoritmos criptográficos (avalanche, difusão e confusão) do PapílioXP e do Rijndael(AES) nos modos de operação ECB, CBC, CFB e OFB. Neste capítulo também foram exibidos os índices de qualidade dos algoritmos nos quatro modos de operação para comparação entre o PapílioXP e o algoritmo Papílio agora chamado de Papílio Tradicional.

1.2. Objetivo do trabalho

Este trabalho tem como objetivo conferir ao algoritmo Papílio a capacidade de variar a sua configuração, mudando o modo como codifica um bloco a cada camada da Rede de Feistel com o intuito de tornar a criptografia desempenhada por este mais forte, dificultando a quebra da mensagem, e otimizar a sua estrutura interna, afim de produzir uma criptografia mais eficiente.

Capítulo 2 – Fundamentos

2.1. Criptografia

A palavra Criptografia provém do Grego Κρνπτός que significa oculto e γράφειν que significa escrita. Desde as civilizações antigas a criptografia tem sido associada a proteção de informações de caráter militar ou político. Sua história se divide em três períodos: até a década de 1940 era mais uma arte do que uma ciência, muitos dos algoritmos de encriptação não tinham muita fundamentação matemática e não possuíam uma formulação algébrica consistente. Esta época é conhecida como criptografia pré-científica. A partir de 1948 com o surgimento da Teoria da Comunicação por Shannon dar-se início a Criptografia Científica, que era mais elaborada e baseada fortemente em conceitos matemáticos. Em 1976 surge a criptografia de chave pública, atribuída ao trabalho de Diffie e Hellman.

O organograma na figura 2.1 mostra as três técnicas básicas que compõe a criptografia nos dias de hoje.

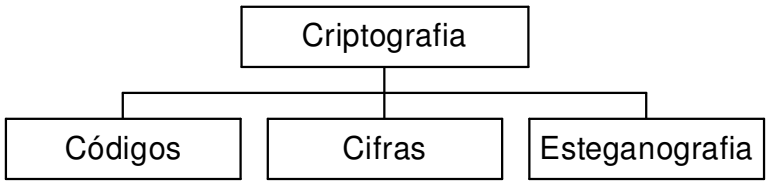

Figura 2.1: Conjunto de técnicas Básicas que compõem a Criptografia

2.2. Códigos

A criptografia feita por códigos é aquela onde o ocultamento do conteúdo da mensagem é baseado no desconhecimento do significado dos símbolos empregados para se escrever a mensagem criptografada. Um exemplo seria um recado gravado em uma secretária eletrônica em língua Tupi Guarani; pouquíssimas pessoas seriam capazes de entender este recado.

2.3. Esteganografia

No decorrer da história, o desenvolvimento da criptografia deu-se paralelamente ao da esteganografia. O objetivo da esteganografia é esconder a existência da mensagem, mas encontrando-se esta sempre no estado original. Um dos exemplos clássicos da esteganografia é atribuído a história de Histaiæus que queria convencer Aristágoras de Mileto a revoltar-se contra o rei Persa. Para passar a mensagem, aquele rapou a cabeça do mensageiro, tatuou a mensagem no couro cabeludo e esperou que o cabelo crescesse. O mensageiro pôde viajar sem problemas e, quando chegou ao destinatário, rapou a cabeça.

2.4. Cifras

Segundo o dicionário [GLO] a palavra cifra quer dizer: conjunto dos símbolos ou caracteres convencionais de uma escrita que não deve ser compreendida por todos. As cifras podem ser divididas em duas classes: cifras por transposição e cifras por substituição.

2.4.1. Transposição

Na transposição as letras originais do texto são preservadas, existindo apenas uma troca das suas posições. Por exemplo, a palavra RAPOSA pode ser codificada para PARASO. Porém, para mensagens muito curtas, este método torna-se pouco seguro, já que há um número limitado de formas para reordenar as poucas letras existentes. Uma das estratégias de transposição sistemática é a cifra "em grade" (veja a tabela 2.1), em que a mensagem é escrita com letras alternadas em linhas separadas.

MUDAM-SE OS TEMPOS	M D M E S E P S U A S O T M O	MDMESEPSUASOTMO

Tabela 2.1: Cifra em Grade

2.4.2. Substituição

Na substituição as letras do texto são trocadas por outras letras, números ou símbolos. Por exemplo, a palavra raposa passa a OVLMCV. Um dos primeiros exemplos da substituição acha-se no Kama-Sutra, recomendando às mulheres que aprendam escrita secreta, de forma a ocultarem as suas relações. Um dos métodos seria fazer pares de letras do alfabeto ao acaso e substituir cada letra da mensagem original pelo seu par. Cada cifra por substituição tem um algoritmo e uma chave, sendo o algoritmo o método de cifragem (transposição ou substituição). A chave define o alfabeto de cifra exato a ser usado para uma dada cifragem. Segundo o lingüista Auguste Kerckhoffs, a segurança deve depender da chave e não do algoritmo. Outro exemplo clássico é a cifra de César, onde o imperador se limitava a substituir cada letra da mensagem pela terceira à sua frente no alfabeto.

Capítulo 3 - Codificador Papílio

3.1. Rede de Feistel

A rede de Feistel é uma seqüência de operações sobre um texto com o objetivo de cifrar este. A rede processa o texto em blocos em um número de interações previamente definido, veja sua representação na figura 3.1, o bloco a ser codificado é dividido ao meio (D_0: direito, E_0: esquerdo) que irá gerar um bloco cifrado que obedece as seguintes equações.

$$E_i = D_{i-1}$$
$$D_i = E_{i-1} \text{ Xor } F(D_{i-1}, Sc_i)$$

Onde "i" representa o número da interação da rede de Feistel, a função "F" tem dois argumentos; o primeiro é um semibloco proveniente de uma interação anterior da rede, o outro é uma subchave criptográfica, que é um bloco gerado por uma função de expansão de chaves que usa como argumento a chave criptográfica.

A idéia da rede de Feistel está centrada na seguinte propriedade da função Xor.

$$(A \text{ Xor } B) \text{ Xor } A = (A \text{ Xor } A) \text{ Xor } B = 0 \text{ Xor } B = B$$
$$(A \text{ Xor } B) \text{ Xor } B = (B \text{ Xor } B) \text{ Xor } A = 0 \text{ Xor } A = A$$

Que em criptografia é utilizado como:

<Texto> Xor <Chave> = <criptograma>
<criptograma> Xor <Chave> = <Texto>

O que acarreta em:

<criptograma> Xor <?> = <?>

é um sistema sem solução -- codificador ON TIME PAD.

Cada transformação na rede de Feistel preserva um semibloco, e codifica o outro semibloco através do Xor deste e o resultado de uma função de substituição que usa como argumentos uma subchave gerada a partir de um chave secreta e o semibloco preservado. O processo de decodificação consiste em processar o texto da ultima camada para a primeira, usando as subchaves também de forma inversa. Algébricamente a decodificação pode ser descrita pelas seguintes equações.

$$D_i = E_{i+1}$$
$$E_i = D_{i+1} \text{ Xor } F(E_{i+1}, Sc_i)$$

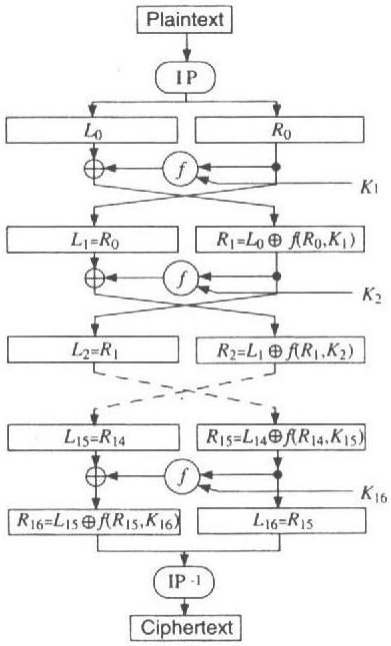

Figura 3.1: Rede de Feistel Codificando

3.2. Codificador Convolucional

Elias (1955), propôs um codificador da categoria FEC, (Foward Error Correct) que insere redundância na mensagem afim de que possa ser corrigida em seu destino. Este codificador é chamado de codificador Convolucional, pois o código gerado é resultado da soma de bits da mensagem original. Uma descrição do codificador Convolucional de uma arquitetura particular é o circuito lógico abaixo.

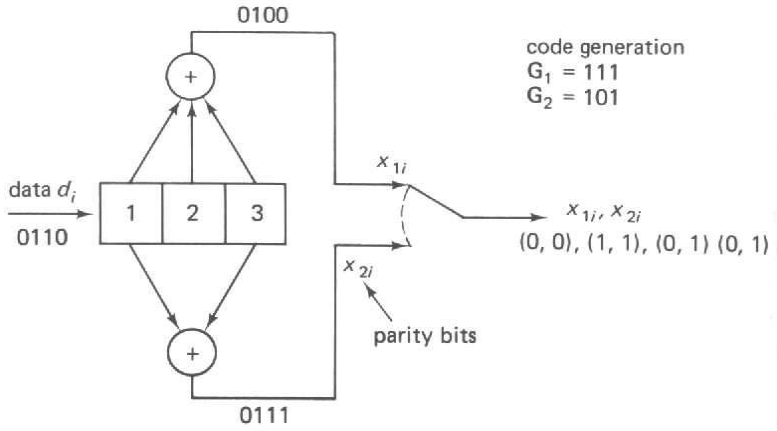

Figura 3.2: Codificador Convolucional

onde temos um registrador de deslocamentos e duas portas Xor de múltiplas entradas. Para esta arquitetura de Codificador convolucional as entradas das portas Xor podem ser ditadas pelos polinômios geradores G_1 e G_2 cujo significado é; se um coeficiente for Zero a célula do registrador cuja ordem corresponde ao grau do coeficiente não participará da entrada da porta Xor, mas se o coeficiente for Um a célula participará da entrada. Para que o processo de codificação inicie é preciso atribuir um estado ao registrador de deslocamentos (carregar uma palavra), depois disso o codificador opera sobre fluxo de bits que chegam continuamente na entrada. Sua saída é representada pela concatenação dos fluxos de X_1 e X_2.

O codificador também pode ser visto como uma máquina de estados (vide Figura 3.3). No caso da Figura 3.2, o codificador tem três células de memória (células 1, 2, 3), para efeito dos nossos cálculos foram consideradas apenas as células 2 e 3, (qualquer outra combinação seria possível) logo temos quatro estados possíveis: 0, 1, 2 e 3. Inicialmente, através do carregamento do clear do conteúdo as memórias 2 e 3, o codificador é colocado no estado 0. Se o primeiro bit de entrada, que é armazenada na célula 1, for 0, o próximo estado permanecerá em 0, porém se for

1 ocorrerá uma transição de estado, e o próximo estado será 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:
0	00	10
1	00	10
2	01	11
3	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 0 e na entrada tem-se um 0, as saídas X_2 e X_1 serão 0, logo o símbolo de saída do codificador será 00. Se o bit de entrada for 1, as saídas X_2 e X_1 serão 1 e o símbolo de saída 11. 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:
0	00	11
1	11	00
2	10	01
3	01	10

Tabela 3.2: Símbolos de Saída

A figura 3.3 exibe a representação da máquina de estados das tabelas 3.1 e 3.2. A tabela 3.3 contempla todos os dados do codificador exibido na figura 3.1.

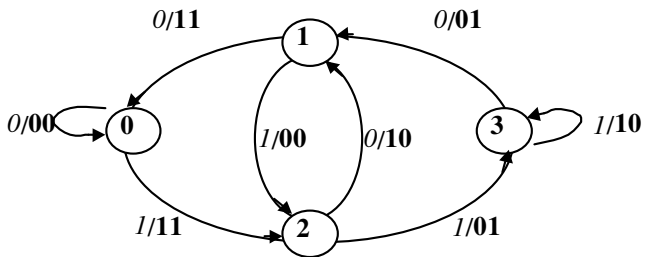

Figura 3.3: Máquina de Estados Finitos

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: Máquina de Estados do Codificador

Para o nosso codificador 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.

3.3. Decodificador Viterbi

Um dos mais populares algoritmos para decodificação de códigos Convolucionais é o algoritmo Viterbi [JAM]. Ele pertence a classe dos decodificadores chamados de; Decodificadores de Máxima Verossimilhança pois garante a máxima similaridade entre a cadeia codificada no emissor e a cadeia decodificada no receptor. Devido a isso é grande sua robustez contra ruídos incorporados à mensagem ao longo do seu trajeto da fonte emissora para o receptor.

Para um codificador Convolucional como mostrado na figura 3.2, podemos implementar um decodificador de máxima verossimilhança comparando-se as 2M seqüências de saída (onde M é o comprimento de um fluxo que sai de uma das portas do Codificador Convolucional) com todos os $4(2^M)$ caminhos possíveis que partem dos quatro nós (quantidade de símbolos que podem sair por vez concatenando-se as saídas do Codificador Convolucional) e selecionando-se a seqüência de código com a maior correlação cruzada. Este cálculo é extremamente caro em termos computacionais para um M grande e resulta em uma estrutura de decodificador bastante complexa.

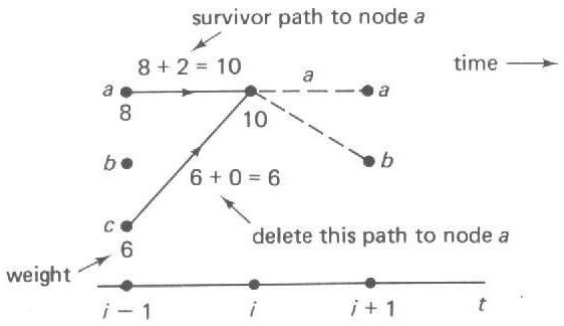

Figura 3.4: Computação dos Pesos

O cálculo da máxima verossimilhança feita por Viterbi considera que cada um dos 4 nós possui dois predecessor e não 4 e que somente o caminho que possuir a maior auto-correlação liga um nó ao outro. Por exemplo, os caminhos que chegam ao no "a" podem ter pesos 10 e 6 como mostrado na figura 3.4. Em cada nó, o peso do caminho sobrevivente determina o novo peso. Por exemplo, para t_i o caminho "aa" adiciona um peso 2 ao peso 8 do nó anterior. A adição de pesos pode ser computada pela correlação dos bits codificados recebidos r_{1i} e r_{2i} com o bit de paridade para cada transmissão p_{1i} e p_{2i} para produzir $w_i = p_{1i}r_{1i} + p_{2i}r_{2i}$.

A seguir dois exemplos de decodificação Viterbi; um para uma cadeia recebida sem corrupção e outro para uma cadeia corrompida.

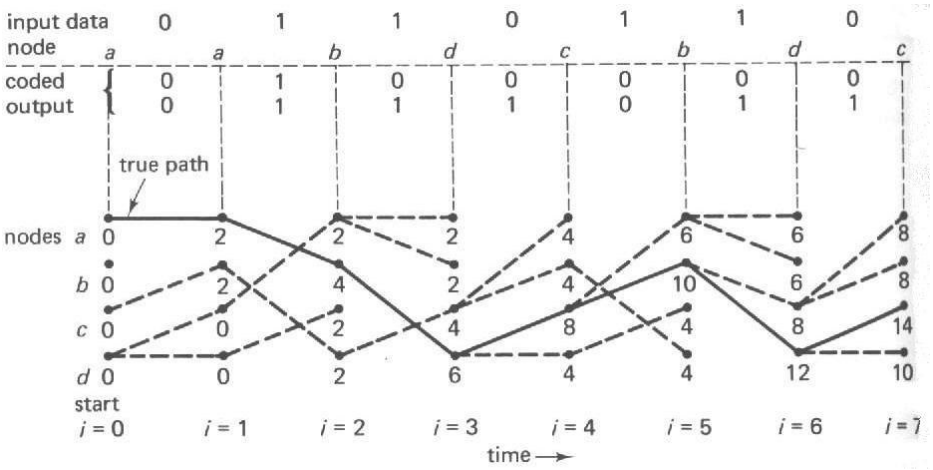

Figura 3.5: Decodificação Viterbi feita para uma cadeia sem erro

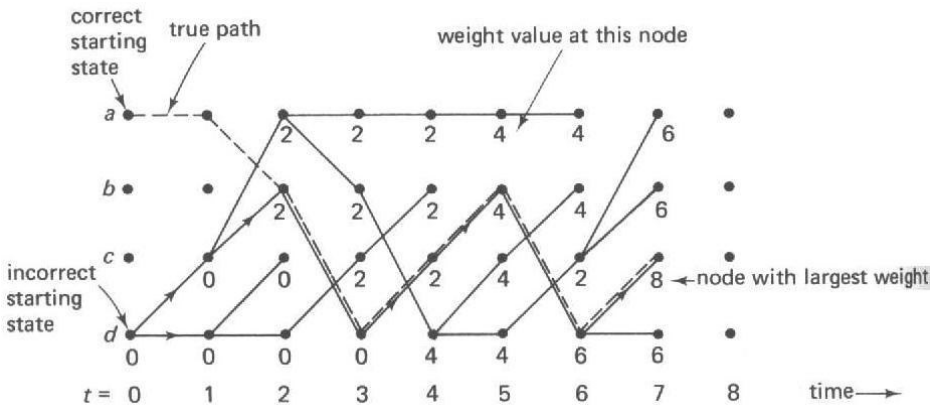

Figura 3.6: Decodificação Viterbi feita para uma cadeia com erro

A tabela 4.1 exibe a uma máquina de estados finita que representa todas as transições possíveis em função da entrada do decodificador Viterbi. Note que as saídas geradas pelo codificador Convolucional são as entradas do algoritmo Viterbi (veja a tabela).

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: Máquinas de Estado do Decodificador

3.4. Viterbi Modificado

A explicação que segue foi uma adaptação de [RAM]

O algoritmo desenvolvido por Andrew J. Viterbi foi proposto como solução para a decodificação convolucional. 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 3.3). A quantidade de transições de cada estado (2^n) é determinada pela quantidade de bits da entrada n. 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 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 portas Xor, determinam-se os polinômios geradores do par de bits da saída. Por exemplo, o polinômio gerador da saída X_1, da figura 3.2, é 111, e o da saída X_0 é 101. 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. Para o codificador figura 3.2 ,o valor de m, a quantidade de memórias consideradas para o cálculo dos estados de codificação é igual a 2, as células 2 e 3, logo a tabela terá quatro (2^2) estados possíveis. A tabela 3.1 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 3.2. 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 anterior, 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 (S0) mais uma saída (S1). A saída S0 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 decodificador Viterbi associado ao Codificador Convolucional da figura 3.2 cada par de bits de entrada (rótulo de entrada) gera um bit de S0 (rótulo de saída). A saída

S1 informa se cada rótulo de saída de S0 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 S0 é obtido de acordo com o algoritmo Viterbi a saída S1 gera o bit 0, do contrário a saída S1 gera o bit 1.

A tabela 3.4 exibe a máquina de estados do Decodificador Viterbi para o codificador da figura 3.2. 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 (S1) a informação 0. Por exemplo, no caso da tabela 3.5, 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 S1, enquanto a saída S0 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 S0, apenas o primeiro dos $\lceil s/n \rceil$ rótulos de tamanho n (o fluxo S1, 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.

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 3.2,o processo descrito acima pode ser simplificado usando-se a tabela 3.5.

ESTADO ATUAL	ENTRADA	S0	S1	PRÓXIMO ESTADO
0	00	0	0	0
0	01	0	1	2
0	10	1	1	1
0	11	1	0	2
1	00	1	0	2
1	01	0	1	2
1	10	1	1	1
1	11	0	0	0
2	00	0	1	0
2	01	1	0	3
2	10	0	0	1
2	11	1	1	3
3	00	0	1	0
3	01	0	0	1
3	10	1	0	3
3	11	1	1	3

Tabela 3.5: Máquina de Estados - Viterbi Modificado

3.5. Expansão de Chaves

O algoritmo Papílio utiliza uma chave de 128 bits, a qual gera 16 subchaves. As subchaves criadas são armazenadas temporariamente em um vetor até que o processo de codificação do bloco seja finalizado. Para geração das subchaves (vide figura 3.7), a tabela 3.5 de codificação é utilizada. O esquema de geração é o seguinte: as quatro primeiras subchaves, rotuladas SC1, SC2, SC3 e SC4 são geradas aplicando-se a codificação Viterbi Modificado na chave inicial de 128 bits, a qual gera dois fluxos de 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.

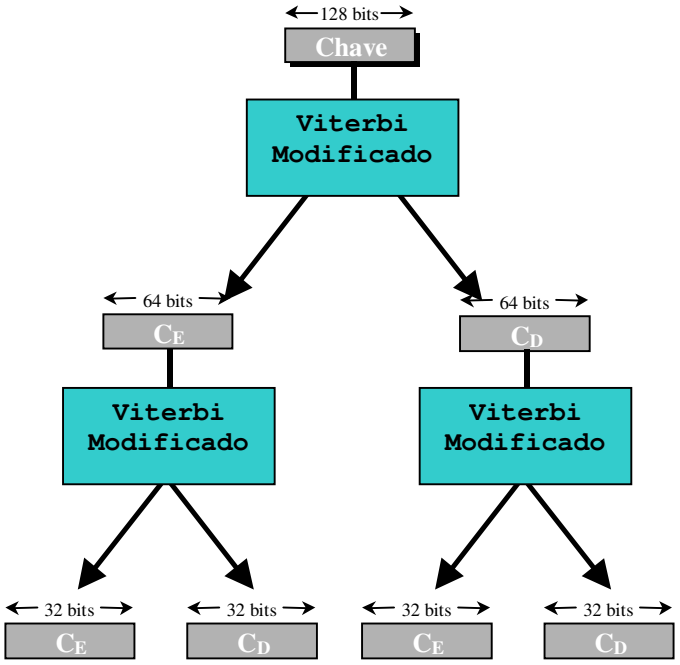

Figura 3.7: Expação de Chaves

3.6. Modos de Operação

3.6.1. ECB - Eletronic Code Book Mode

No modo ECB o texto claro é dividido em blocos de igual tamanho, que são codificados independentemente usando-se a mesma chave criptográfica. A característica de independência entre os blocos codificados permite ao codificador em modo ECB por exemplo codificar uma base de dados que é organizada em arquivos consultados de maneira aleatória. A independência entre os blocos também garante ao modo ECB robustez contra erros pois caso um bloco apresente algum erro, só ele será prejudicado, e não toda a mensagem. A figura 3.8 ilustra os esquemas do codificador e do decodificador no modo ECB.

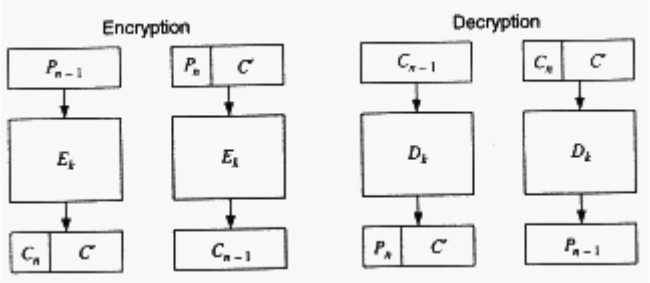

Figura 3.8: Esquemas do modo ECB

3.6.2. CBC - Cipher Mode Chaing Mode

O modo CBC incorpora um mecanismo de retroalimentação ao cifrador de blocos, o que significa que o bloco anterior influencia na construção do bloco atual. Isso é conseguido fazendo-se uma operação XOR entre o bloco da mensagem que se quer codificar e o último bloco cifrado obtido. Em seguida o bloco operado é codificado usando-se um codificador em blocos. Para evitar o caso em que duas mensagens iguais que codificadas com a mesma chave obtenham cifras iguais é usado um vetor de inicialização que pode ser um bloco aleatório colocado no início da mensagem afim de influenciar na construção de todos os blocos da mensagem. O vetor de inicialização tem também a função de emular o bloco cifrado anterior que será usado na construção do primeiro bloco. A figura 3.9 ilustra os esquemas do codificador e do decodificador no modo CBC.

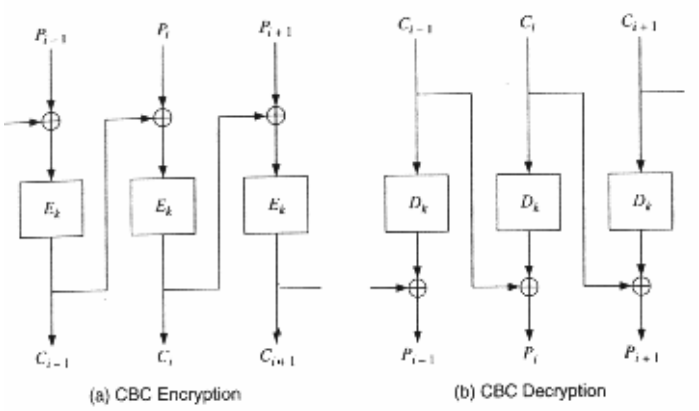

Figura 3.9: Esquema do modo CBC

3.6.3. CFB - Cipher Feedback Mode

O modo CBC apenas codifica ou decodifica corretamente se receber ou transmitir um bloco completo de informação. Isto pode se converter em um sério inconveniente se o protocolo de comunicação usado variar o tamanho do bloco de informação com o qual trabalha. A proposta do modo CFB é trabalhar com uma quantidade de dados inferior ao tamanho do bloco.

O esquema usado para explicar o modo CFB pode ser visualizado na figura 3.10. Ele consiste em uma fila de comprimento de um bloco que é a entrada de um cifrador no modo ECB. Esta fila por sua vez é alimentada pelo produto do Ou-Exclusivo entre o semibloco de texto que se deseja codificar e a saída do cifrador. Para o decodificador o esquema é similar, mas com a diferença de que os semiblocos que alimentam a fila são retirados do texto de entrada. Como no modo CBC aqui também faz-se necessário pelas mesmas razões um vetor de inicialização.

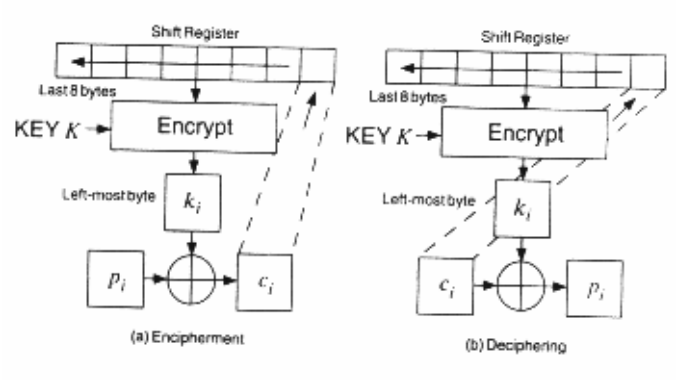

Figura 3.10: Esquema do modo CFB

3.6.4. OFB – Output Feedback Mode

O modo CFB foi projetado para resolver o problema do desperdício de banda do canal, mas possui uma fraqueza: se um bloco transmitido apresentar algum tipo de erro na transmissão, toda a mensagem decriptada no destino estará perdida. Em um ambiente com baixa taxa de erro como uma rede de computadores essa limitação do modo CFB não é crítica. Porém quando se trata de comunicações móveis, aonde o tempo de envio de um pacote é grande, cerca de 0,25s e o sinal está sujeito a interferências elétro magnéticas das mais diversas como; flutuações na rede elétrica, uma outra transmissão que esteja operando na mesma faixa de freqüência, falha em algum dos dispositivos de transmissão enfim; a probabilidade de um pacote está corrompido não é um fator que possa ser desprezado, o que torna bastante desejável que a corrupção de um pacote no ato da transmissão não prejudique a decodificação dos demais blocos da mensagem. Para suprir essa deficiência do modo CFB a idéia inicial que se tem é a da remoção do encadeamento, mas esta abordagem seria ao custo de uma mesma mensagem encriptada com a mesma chave sempre resultar em um mesmo criptograma. A solução encontrada foi manter a realimentação, mas de uma forma controlada a qual não sofresse influência alguma do meio externo. Isso é implementado com uma pequena modificação na estrutura do modo CFB, onde o bloco de entrada da fila que alimenta o cifrador de blocos é a sua própria saída.

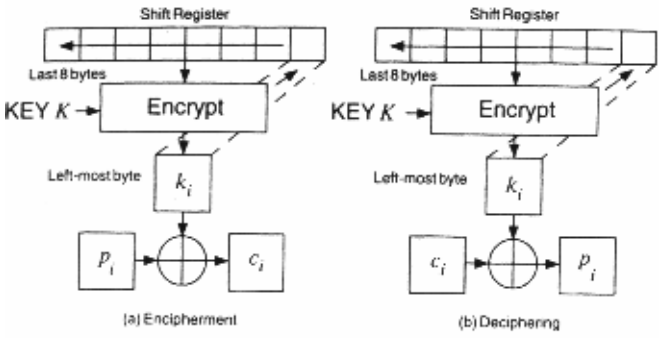

Figura 3.11: Esquema do modo OFB

3.7. Algoritmo Papílio

Para a encriptação de um texto o algoritmo Papílio se vale de uma rede de Feistel que usa como função de substituição o Viterbi Modificado. A entrada de Papílio consiste em dois argumentos: o texto que se deseja cifrar, e a chave criptográfica. A primeira ação de Papílio é submeter a chave a um processo de expansão de chave, que é a geração a partir da chave ,128 bits, de 16 blocos de 32, visto na seção 3.5. Em seguida o texto é dividido em blocos de 64 bits e cada um deles é processado pela Rede de Feistel.

Procedimento Papílio (texto, cifra, chave)

```
Início
  caracter bloco[BLOCO_PAPILIO];
caracter subChave[ROUND_PAPILIO][HALF_BLOCK];
  inteiro i;

ExpansãoChave (chave);
  Enquanto (i < tamanho(texto)) faca
  inicio
    copia (bloco, texto[i], BLOCO_PAPILIO);
    Feistel (bloco, subChave);
    cifra := cifra + bloco;
    i := i + BLOCO_PAPILIO;
  fim
fim
```

Tabela 3.6 Algoritmo Papílio

O processo de decriptação é bastante similar, diferindo apenas em dois aspectos: o texto de entrada é um texto anteriormente cifrado por Papílio e a Rede de Feistel processa da última camada para a primeira para recuperar o texto original. Papílio foi implementado em linguagem C nos quatro modos de operação: ECB, CBC, CFB e OFB. Para a computação da função de substituição Viterbi Modificado é usada a representação em forma de máquina finita de estados, que é construída por uma rotina que implementa o Viterbi Modificado.

Capítulo 4 - Otimização Papílio

4.1. Características desejáveis em um algoritmo criptográfico

Um algoritmo criptográfico deve oferecer a maior dificuldade possível à descoberta por indivíduo não autorizado do conteúdo da mensagem cifrada. Os algoritmos da atualidade se baseiam em três critérios para atingir tal objetivo, são eles: avalanche, difusão e confusão.

4.1.1. Avalanche

A avalanche diz respeito a influência que um bit da chave ou do texto claro tem no resultado do processo de cifra. É considerado bom o algoritmo onde havendo modificação de um bit na chave ou no texto claro, influencie no estado de 50% dos bits do texto cifrado[RAM]. Essa característica torna difícil ao criptoanalista deduzir a chave ou o texto claro a partir do texto cifrado, pois se ele erra em pelo menos um bit o texto cifrado que ele obtém é diferente do cifrado do original em 50% dos bits.

4.1.2. Difusão

A difusão é a destruição da estatística dos símbolos presentes no texto cifrado. Ela coíbe o ataque estatístico, que se baseia na idéia de que símbolos do texto cifrado tem freqüência de aparição similar aos símbolos do texto claro. Caso o texto claro seja proveniente de uma linguagem natural, basta que o criptoanalista disponha de uma tabela da freqüência natural de aparição destes caracteres para fazer a correspondência um para um dos símbolos do texto cifrado e do texto claro.

4.1.3. Confusão

A confusão é o quão obscura a relação entre o texto claro e o texto cifrado. Quando um bit da chave ou do texto claro é alterado o algoritmo que satisfaz a confusão deve produzir um texto cifrado com símbolos diferentes de quando não lhe é alterado nenhum bit na chave ou no texto claro. A confusão é similar a avalanche, só que ela se preocupa com os símbolos gerados e não com o estado dos bits.

4.2. Motivação para otimização

O algoritmo Papílio utiliza como função de substituição o Viterbi Modificado, que pode ser visto como uma máquina de estados que é construída pelo algoritmo Viterbi Modificado o qual tem como entrada um par de polinômios em notação binária. Os polinômios são então fator determinante no processo de cifra influenciando diretamente nos valores dos índices de avalanche, difusão e confusão. Para tornar o algoritmo Papílio mais eficiente fomos levados a selecionar um conjunto de polinômios que confiram a Papílio bons índices criptográficos contribuindo diretamente para tornar a criptografia produzida por ele mais forte.

4.3. Procedimentos utilizados para otimização

4.3.1. Medição da Avalanche, Difusão e Confusão

Inicialmente foram coletados índices de avalanche, difusão e confusão para 64 polinômios em notação binária. Para os testes de Confusão e Difusão foram usados 3136 caracteres de textos em língua Inglesa, mais precisamente um dos livros do projeto Gutenberg [GUT] onde foram desprezados os primeiros 400 caracteres para a eliminação do cabeçalho que contém a descrição de cada um. As chaves usadas nestes teste são geradas de modo aleatório, como também os blocos de texto usados no teste da avalanche.

Para se medir a avalanche que é o percentual de bits alterados na saída do codificador quando se altera um bit do texto de entrada, fez-se a média das distâncias de Hamming entre o bloco de texto encriptado a partir de um bloco de texto claro e o bloco provenientes da encriptação do bloco de texto claro alterado em um bit em todas as possibilidades. Para se medir a difusão, ou destruição da informação estatística dos caracteres do texto encriptado, foi calculado o desvio padrão das freqüências do conjunto de caracteres do texto cifrado. Para se medir a confusão que é a influência que um bit da chave ou do texto claro tem na formação do criptograma foi usada a média das distâncias Euclidianas entre o texto encriptado do texto claro com uma chave e todos os textos resultantes da encriptação do texto claro e da chave alterada em um bit.

Para a realização destes testes foi implementado em linguagem C um programa chamado "metrix", cuja entrada é um arquivo que contém a descrição dos textos que serão utilizados no teste (quantidade de textos, seu tamanho e seu path), e também a quantidade de avalanches que será feita por cada polinômio.

Como saída o programa retorna três arquivos correspondentes a cada índice (avalanche, difusão e confusão), cada linha destes arquivos tem um par de números; polinômio e índice obtido. Para se permitir uma melhor leitura dos dados os resultados obtidos das medidas foram normalizados de forma a ficarem entre zero e um. Após a normalização o resultado dos três arquivos foi plotado em um único gráfico onde o eixo horizontal corresponde aos polinômios e o vertical ao valor normalizado do índice obtido. Vale ainda destacar que a curva relativa ao índice difusão foi exagerada por um fator de 10x para que fosse possível a sua visualização. Nas figuras 4.1-a

e 4.1-b são exibidos os resultados dos índices avalanche, difusão e confusão para para dois textos diferentes (tor0 e tor1 respectivamente) no modo ECB.

4.3.1.1. Análise dos resultados (metrix)

Podemos observar que o índice avalanche, representado pela curva vermelha, para a grande maioria dos polinômios atinge índice superior aos 45%, o que é bom pois está próximo do valor teórico de 50%. Constatamos ainda que a difusão não ultrapassou os 4% (3% para a figura 4.1-a e 3,7% para a figura 4.1-b) de desvio padrão o que demonstra que não há relação estatística entre o texto claro e o criptograma. Para avalanche a grande maioria dos polinômios supera os 35% de dissimilaridade entre o texto cifrado com uma chave e o mesmo texto cifrado com a chave alterada em um bit. Os dois gráficos usam textos diferentes e chaves diferente, mas como se pode ver os resultados são bastante similares para o conjunto de polinômios investigados. Como última observação constatamos que o comportamento dos polinômios correspondentes a metade direita de ambos os gráficos exibe um resultado inferior comparada a metade esquerda.

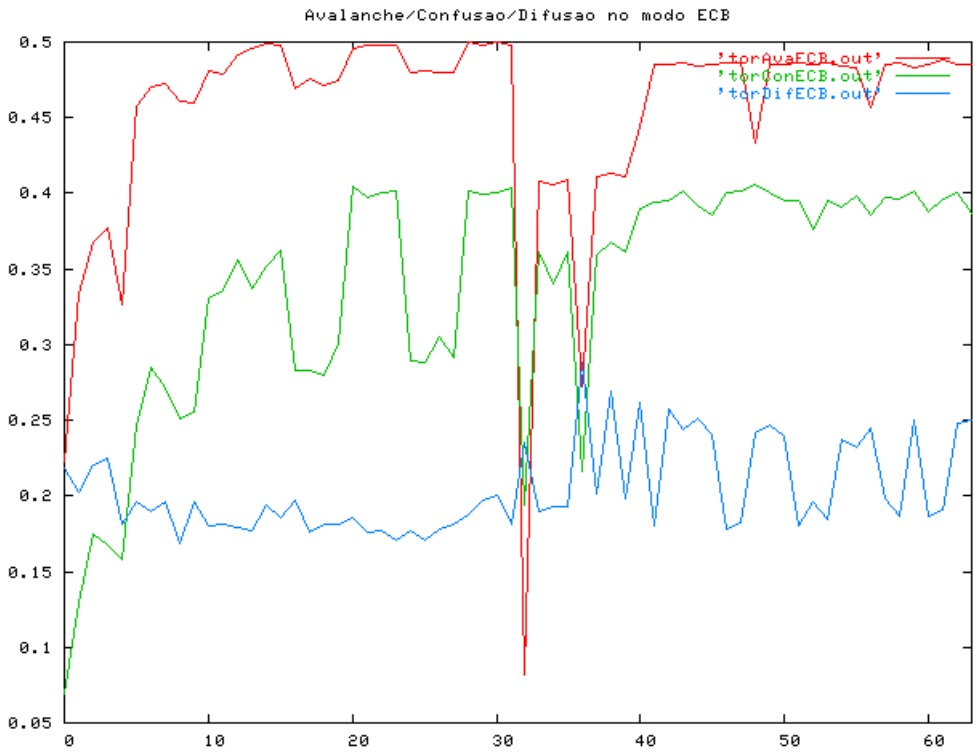

Figura 4.1-a: Avalanche, Difusão e Confusão no modo ECB

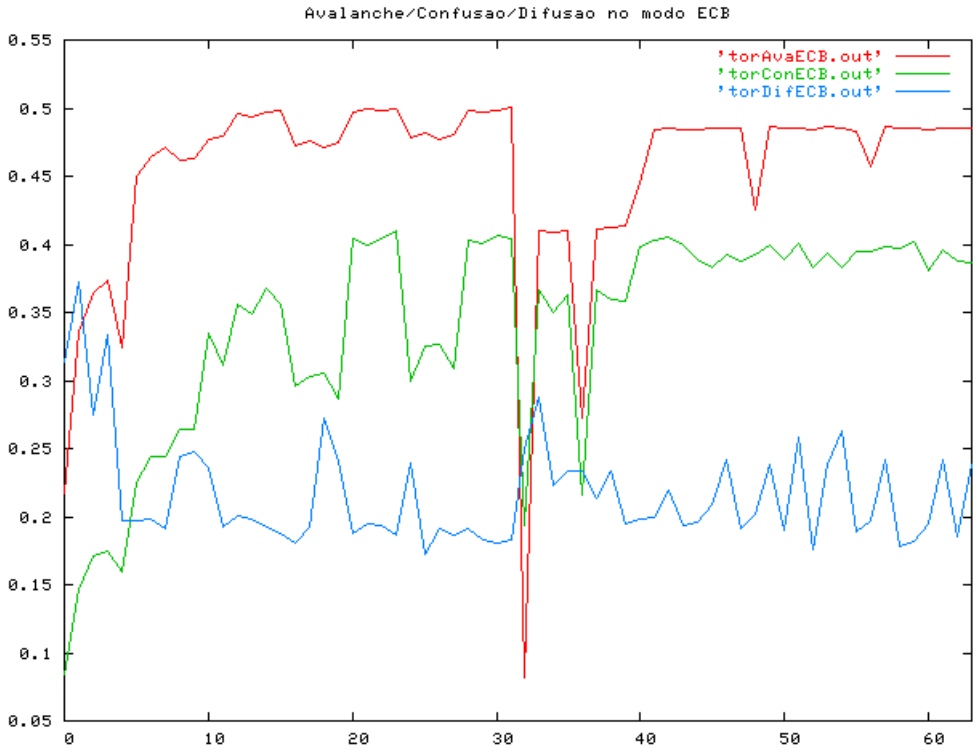

Figura 4.1-b: Avalanche, Difusão e Confusão no modo ECB

4.3.2. Similaridade entre os Índices

Com base nos resultados obtidos nesta primeira análise julgou-se por bem avaliar o comportamento de Papílio para constatar se os índice obtidos podem ser sempre, ou em grande maioria das vezes, conseguidos independente do texto e da chave criptográfica, sendo ele função apenas dos polinômios. Para que isso fosse confirmado, foi desenvolvido um programa chamado "similar" em linguagem C cuja entrada é um arquivo contendo informações sobre: a quantidade de arquivos a serem processados, o tamanho deles, seus caminhos, a quantidade de avalanches à serem feitas, e a quantidade de interações do programa. Para cada teste foram processados 10 arquivos [GUT], que foram uniformemente distribuídos para o conjunto de polinômios. O programa nada mais faz do que medir o desvio padrão dos índices avalanche e difusão para 100 interações de todos os 64 polinômios. As chaves são geradas de maneira aleatória como também os blocos do teste avalanche. Optou-se por não medir a similaridade do índice Confusão atingidos pêlos polinômios em virtude da execução do teste para Confusão ter custo computacional elevado e pelo fato verificado nas curvas de índice feitas pelo programa "metrix" ,figuras 12-a e 12-b, que o efeito Confusão está correlacionado com o efeito avalanche, o que significa que a economia que estamos fazendo não representa prejuízo algum na qualidade da análise similaridade dos índices atingidos pêlos polinômios geradores de máquinas de estado Papílio. O programa "similar" gera dois arquivos de saída, um referente aos desvios para o índice difusão e o outro como os desvios referente aos desvios do índice avalanche. O programa foi executado cinco vezes e as curvas foram plotadas todas em um mesmo gráfico para conferir se há convergência de valores. Para cada execução do programa foi utilizado um conjunto de dez textos que não foram usados nos outros testes.

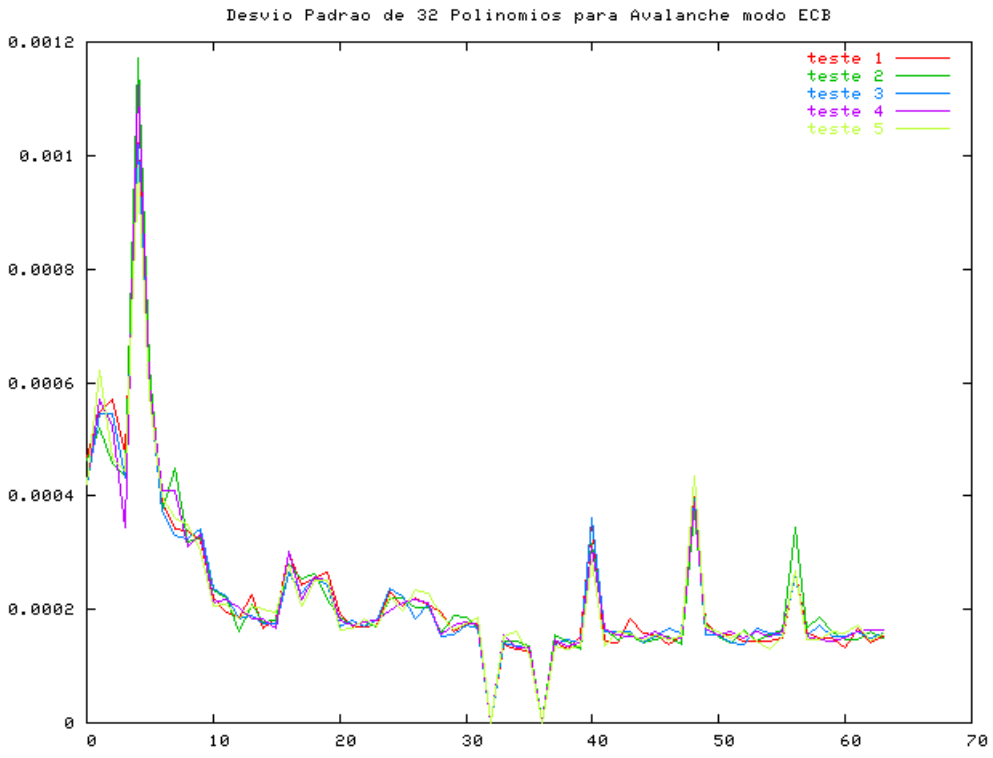

Figura 4.2: Desvio Padrão de 64 Polinômios do Efeito Avalanche no Modo ECB

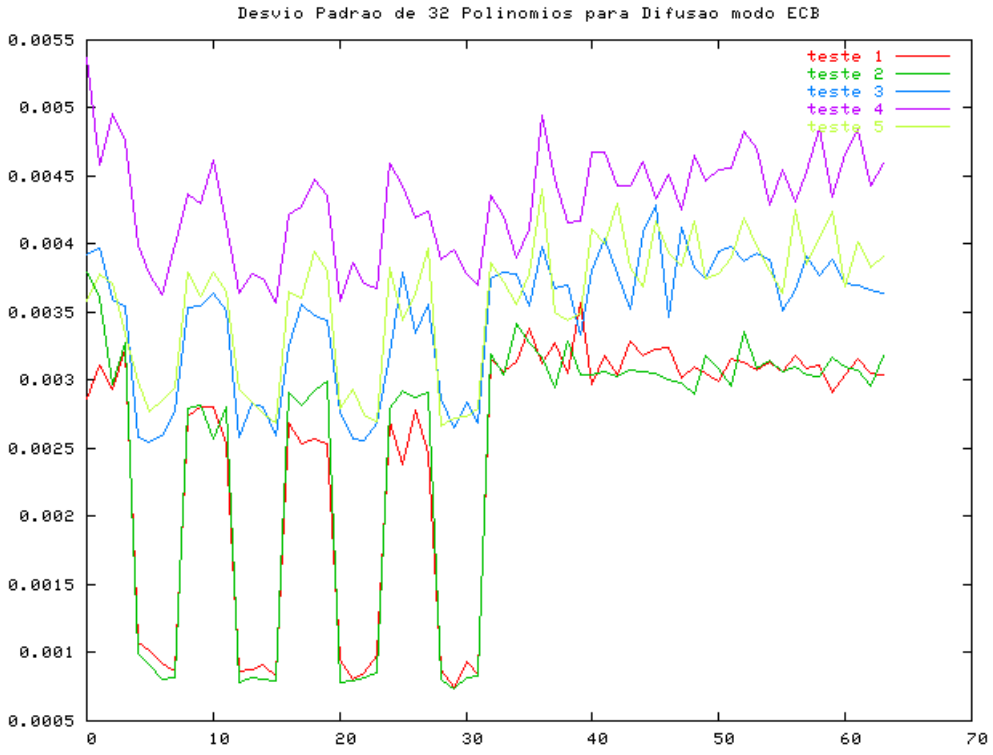

Figura 4.3: Desvio Padrão de 64 Polinômios do Efeito Difusão no Modo ECB

4.3.2.1. Análise dos resultados (similar)

O resultado que vemos nos gráficos nos dá as seguintes certezas: o comportamento das máquinas de Papílio é sempre o mesmo, pois as curvas provenientes de chaves diferentes e textos diferentes apresentam desvios padrões muito baixos. No caso da Avalanche isso é muito forte, por ela ser medida usando-se blocos aleatórios podemos afirmar que as curvas apresentadas na figura 4.2 exibem o comportamento geral do Papílio para a avalanche. Para as curvas exibidas na figura 4.3 os textos influenciam sobre o índice difusão conseguido, mas a variação é considerada pequena, pois valores de picos das curvas são muito pequenos, inferiores a 1%, o que implica que em termos de desempenho as máquinas de estado Papílio se comportam de maneira similar. Logo caso ela seja uma boa máquina, isto é, confere a encriptação Papílio índices Avalanche, Difusão e Confusão bons, ela tem alta probabilidade de sempre dar ao algoritmo um bom desempenho.

4.3.3. Seleção de Polinômios

A certeza do comportamento regular das máquinas de Papílio nos impulsioram a fazer mais uma investigação: descobrir um subconjunto de Polinômios que confira a encriptação Papílio bons índices criptográficos. Foi então escrito um programa chamado “torneio” em linguagem C para fazer a seleção de tais polinômios. Foi usado como critério de otimização o índice Avalanche, por ser barato em termos computacionais e por haver correlação direta entre uma boa Avalanche e bom índice Confusão. Julgou-se por bem desprezar o resultado da difusão na escolha dos polinômios bons, pelo fato dos gráficos apresentados pelo “metrix” no que se refere a difusão terem sido consideradas boas em geral para todos os Polinômios, próximas do limite teórico de zero, mesmo a parte irregular tinha um erro inferior aos 4%, o que o bom senso em engenharia diz que é um erro aceitável está na casa dos 3%. Para a seleção dos Polinômios foram considerados apenas os 32 primeiros pois de antemão sabemos que seu comportamento sempre é melhor do que os seus 32 subsequentes, como mostrado pelas curvas provenientes de “metrix”. O “torneio”, implementou a estratégia de fornecer ao Papílio uma máquina gerada pelo polinômio avaliado, e fazer uma série de 50 avalanches usando blocos e chaves geradas de forma aleatória. Em cada interação do programa marca ponto Polinômio que chegar mais perto do valor ideal que é de uma Avalanche de 50%. Para se evitar que caíssemos em um ótimo local, um polinômio era considerado elegível se marcasse 20 pontos, então ele era retirado do torneio, e o programa continuava a processar sobre os demais, até que fossem selecionados oito polinômios. Posteriormente o programa foi executado com uma entrada mais difícil, agora os Polinômios tinham que marcar 50 pontos para serem considerados eleitos, e o resultado se repetiu. Os polinômios vencedores do torneio se apresentam na tabela 4.1.

O POLINÔMIOS ELEITOS NO TORNEIO							
14	21	22	23	28	29	30	31

Tabela 4.1 Polinômios Eleitos

Capítulo 5 – PapílioXP

5.1. PapílioXP - Algoritmo Papílio eXtenso

Com os resultados obtidos nas seções anteriores apresentamos aqui uma proposta de extensão para o Algoritmo Papílio visando uma criptografia mais forte comparável a dos melhores algoritmos existentes na atualidade, aumentando consideravelmente as formas com as quais ele encripta um texto.

Na abordagem Tradicional do Papílio, trabalha apenas com uma máquina de estados e a usa em todo processo. A máquina usada para o Papílio Tradicional era a gerada pelo polinômio 61. A proposta para PapílioXP é que se use um conjunto de máquinas de Papílio, mais precisamente 8 máquinas, as geradas a partir dos polinômios eleitos pelo programa "torneio". Essas máquinas são escolhidas a cada rodada da rede de Feistel, usando-se uma função de dispersão dupla. A razão para o uso de uma função de dispersão dupla é que elas retornam índices que são equiprováveis para um vasto conjunto de entradas, conforme [SZW]. As entradas usadas na função de dispersão dupla são o valor da rodada corrente e o semibloco que serve como entrada para a função de substituição na rede de Feistel. A estrutura de Feistel sempre garante o retorno deste semibloco rodada à rodada o que significa que o algoritmo sempre acertará máquina de estados que será usada no processo de encriptação, e sempre acertará as máquinas para o processo de decriptação. Outra modificação é o estado inicial que a máquina de estados assume na função de substituição Viterbi Modificado; na abordagem tradicional ela sempre usava um valor fixo, (zero) para qualquer chave e em todas as situações. A modificação proposta é que se aplique também uma função de disperção para se determinar o estado inicial da função Viterbi Modificado. A função de disperção utilizada é a mesma para a seleção das máquinas de estado nas rodadas da rede de Feistel, ele usa com entradas o bloco que será processado pela função Viterbi Modificado e um inteiro de valor constante para a aplicação. Esta modificação na função de substituição não lhe tira a condição de função, pois a função de disperção também é uma função e sabemos da matemática elementar que a composição de funções também é uma função. Essas duas modificações conferem ao PapílioXP um versatilidade incrível. Imagine na abordagem tradicional usávamos apenas uma função de substituição. Agora temos a possibilidade de escolha de $(4x8)^{16}$ possibilidades de funções de substituição o que nos dá um total de 2^{80} escolhas para funções de substituição para a encriptação de cada bloco. O mais

interessante é que essas configurações são feitas dinamicamente, em função do texto que se quer codificar, o que torna a quebra de um bloco por análise do texto codificado virtualmente impossível pelo fato do processo não ser dependente apenas da chave criptográfica, mas também da própria mensagem que se deseja recuperar.

Capítulo 6 – Desempenho do PapílioXP

6.1. Desempenho do PapílioXP frente ao Papílio Tradicional

Para comprovar o ganho de performance do PapílioXP em relação a abordagem Tradicional de Papílio foi confeccionado um programa chamado de "sheha" que mede a avalanche a cada rodada, comparando os blocos gerados pela codificação de um bloco e do mesmo bloco alterado em um bit no início. O valor mostrado no gráfico por rodada é o somatório das distâncias de Hamming entre todos os blocos possíveis gerados alterando-se um bit (são 64 possibilidades) e o bloco original. "Sheha" também mede o Hamming Interno, que é o percentual de bits alterados com relação ao bloco de entrada a cada rodada, não se fazendo alteração alguma no bloco de entrada.

6.1.1. Análise dos resultados (sheha)

O gráfico que exibe o comparativo entre PapílioXp e Papílio Tradicional , figura 6.1, mostra que a avalanche é progressiva rodada após rodada e converge em ambos após a sexta rodada. Repare que a curva do Papílio Tradicional é majorada pela curva exibida pelo PapílioXP. Isto implica PapílioXP tem um desempenho melhor em relação ao Tradicional, atingindo a avalanche de 50% com menos rodadas. Para 64 blocos de 64 bits cada a Avalanche é atingida quando o somatório das distâncias de Hamming é igual a 2048 bits alterados.

O gráfico que exibe o comparativo do Hamming Interno entre PapílioXp e a abordagem Tradicional, figura 6.2, mostra que neste ponto eles são equivalentes.

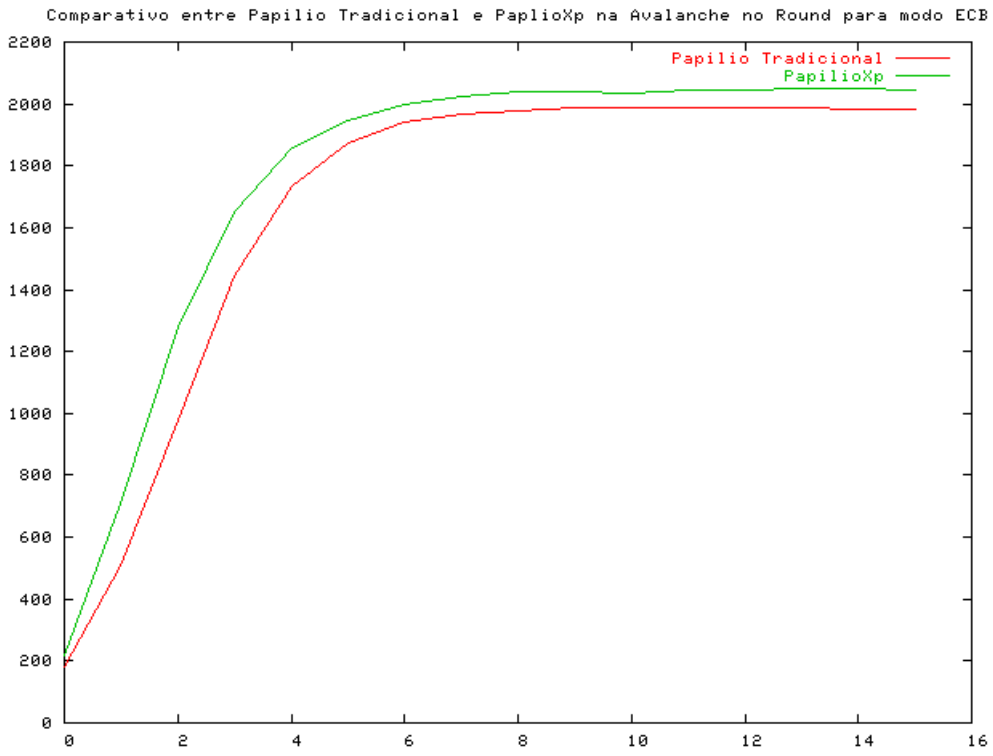

Figura 6.1: Papílio Tradicional x PapílioXP na avalanche no Round no Modo ECB

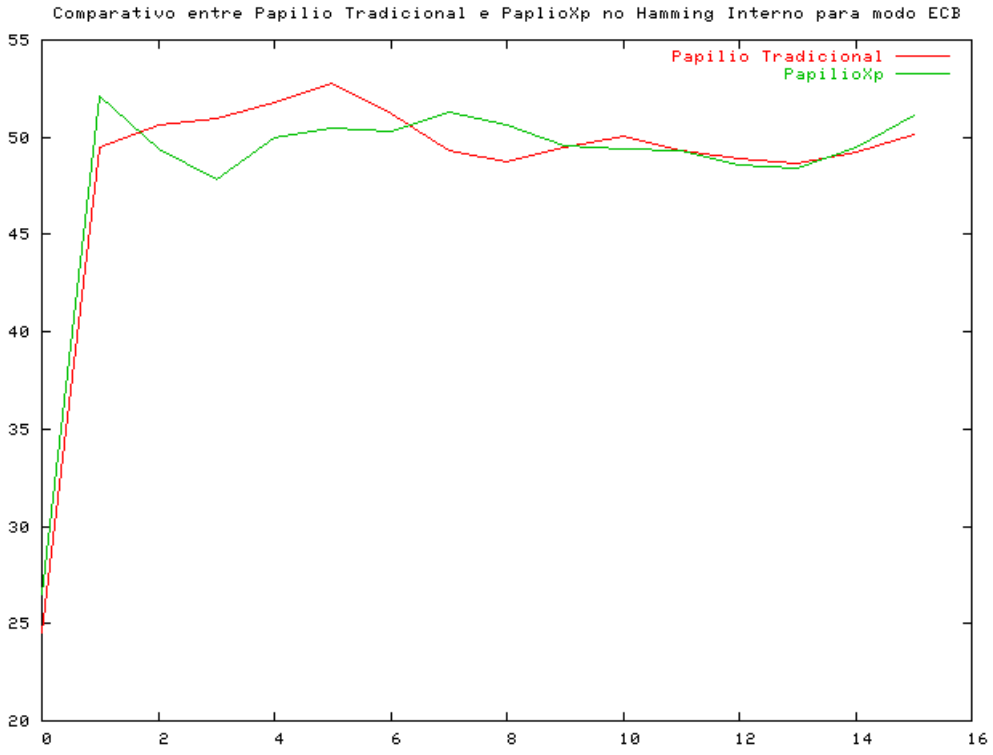

Figura 6.2: Papílio Tradicional x PapílioXP no Hamming Interno no Modo ECB

6.2. Bench-mark do PapílioXP

Para sabermos de fato em que patamar está o PapílioXP nós o comparamos com o mais eminente representante do estado da arte da criptografia simétrica que é o Rijndael, o padrão criptográfico de chave privada do século XXI.

6.2.1 O algoritmo Rijndael (AES)

Em outubro de 2000 o NIST (National Institute for Standards and Technology) anuncia oficialmente a adoção do algoritmo Rijdael como novo padrão Criptográfico para emprego em aplicações criptográficas militares culminando em um processo de mais de três anos destinado a proporcionar a comunidade internacional um novo algoritmo de criptografia, potente, eficiente e fácil de implementar. A palavra Rijndael usada para referenciar este algoritmo é um acrônimo formado pelos nomes dos seus autores, os belgas Joan Daemen e Vincent Rijmen. O critério de escolha usado pelo NIST foi um concurso público, onde o estudo, revisão e avaliação foram feitos por membros da comunidade internacional de criptografia. AES é um sistema de encriptação por blocos, projetado para manejar tamanho de chaves e de blocos variáveis, ambos compreendidos entre os 128 bits e 256 bits. Realiza várias de suas operações internas ao nível de byte interpretando estes como elementos de um corpo de Galois $GF(2^8)$.

6.2.2. PapílioXP comparado ao Rijndael (AES)

PapílioXP foi comparado com o Rijndael nos modos criptográficos ECB, CBC,CFB e OFB, usando chave de comprimento 128 bits. A implementação usada do Rijndael foi a da biblioteca criptográfica **crypto** disponível no site da organização OpenSSL [OPE]. Para se efetuar a comparação entre os dois algoritmos foi escrito um programa em linguagem C chamado “mortalKombat” que tem como entrada um arquivo que diz quantos textos devem ser processados, o seu tamanho, o caminho deles e a quantidade de avalanches que é executada. O “mortalKombat” aplica avalanche em 50 blocos sorteados aleatoriamente de comprimento 128 bits, pois esse é o tamanho mínimo que o Rijndael trabalha, fazemos então com que PapílioXP codifique dois blocos ao invés de um. Nos testes da Confusão e da Difusão são utilizados textos provenientes do site Projeto Gutenberge [GUT]. As chaves para todos os teste são sorteadas de maneira aleatória.

6.2.2.1. Análise dos resultados (mortalKombat): PapílioXP x Rijndael(AES)

No que se refere ao índice Avalanche (o ideal é que se atinja 50%) vemos que o PapílioXP tem desempenho praticamente o mesmo do que Rijndael nos modos ECB e CBC (figuras 6.3, 6.4). Nos modos CFB e OFB (figuras 6.5 e 6.6) podemos considerar um empate técnico.

No quesito Confusão (ganha que tiver em torno dos 50%) a curva do Rijndael majora a do PapílioXP, mas por uma diferença muito pequena (ver figuras 6.7, 6.8, 6.9, 6.10).

No quesito Difusão (ganha quem obter o menor índice) a disputa se deu na casa dos centésimos (veja as figuras 6.11, 6.12, 6.13, 6.14) onde hora o PapílioXP ganhava, hora o Rijdael.

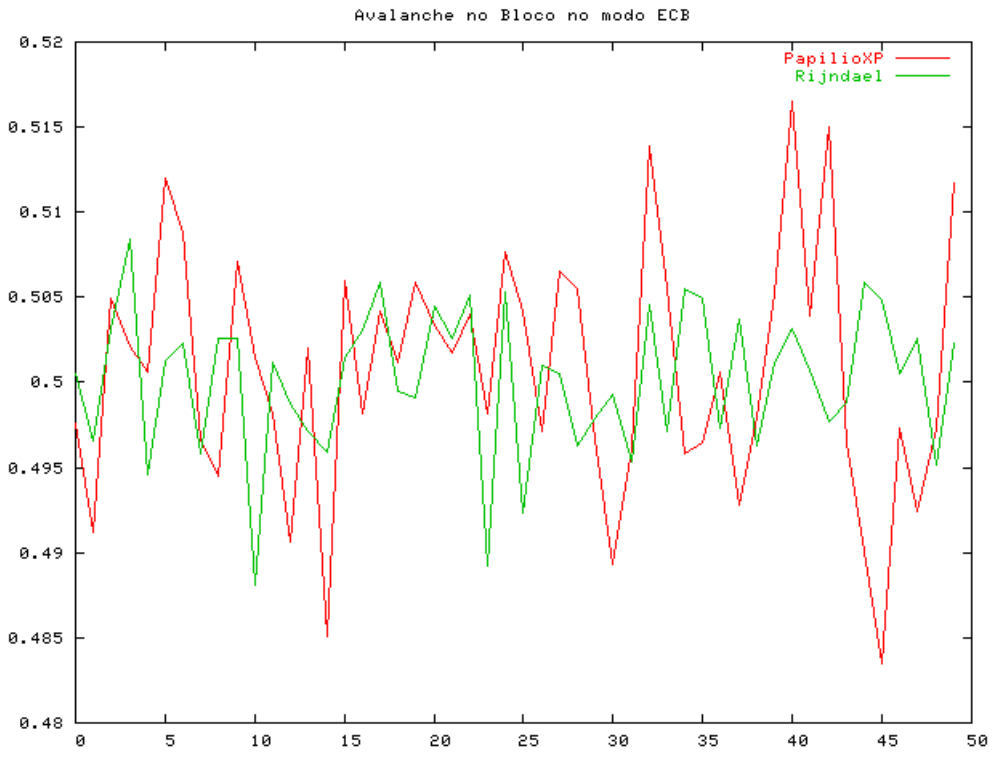

Figura 6.3: PapílioXP x Rijndael - Avalanche no Bloco no modo ECB

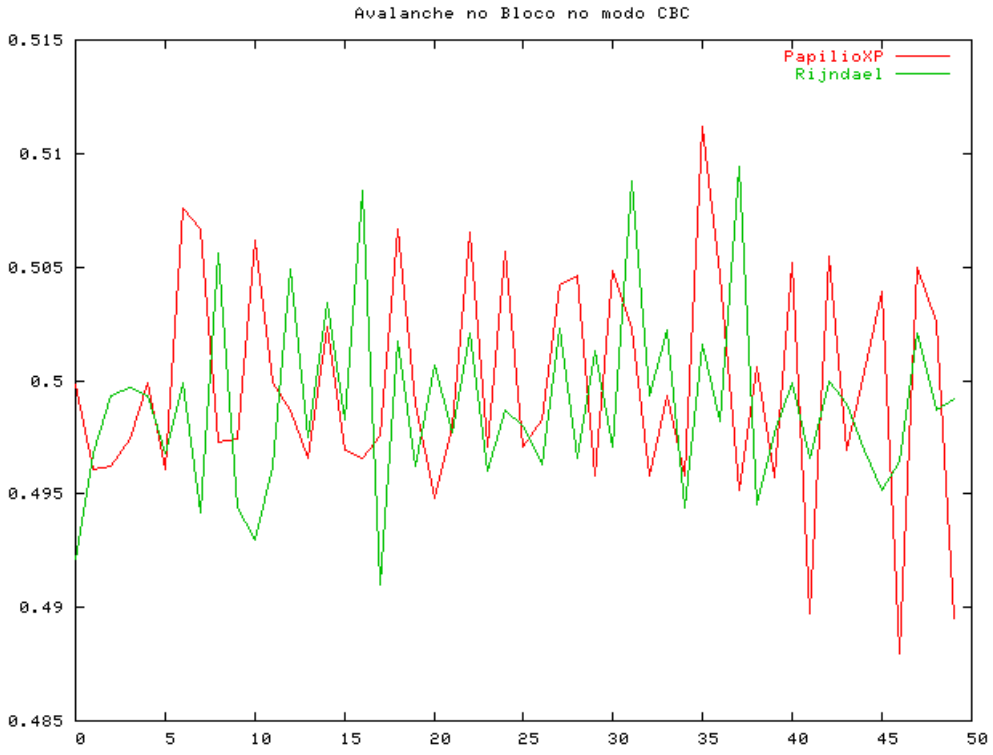

Figura 6.4: PapílioXP x Rijndael - Avalanche no Bloco no modo CBC

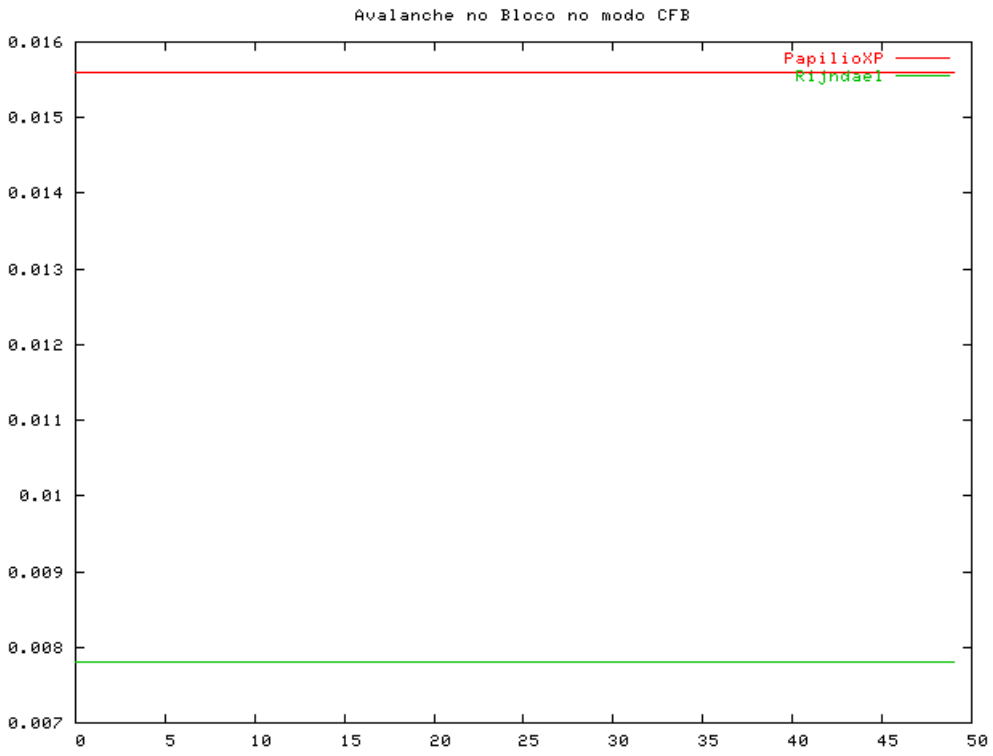

Figura 6.5: PapílioXP x Rijndael - Avalanche no Bloco no modo CFB

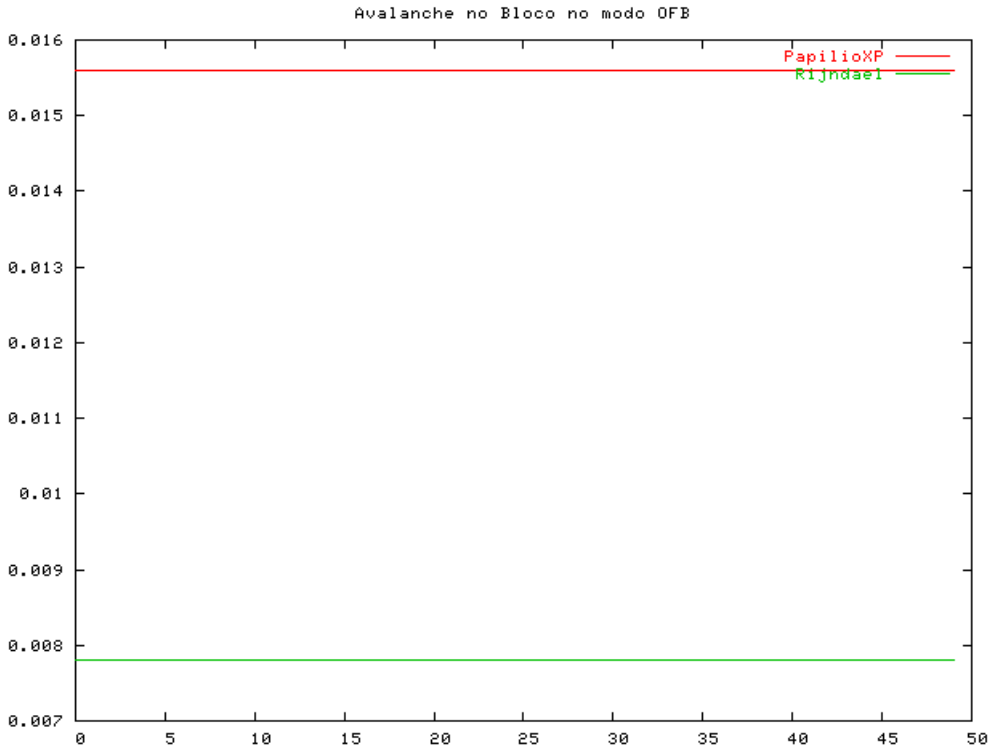

Figura 6.6: PapílioXP x Rijndael - Avalanche no Bloco no modo OFB

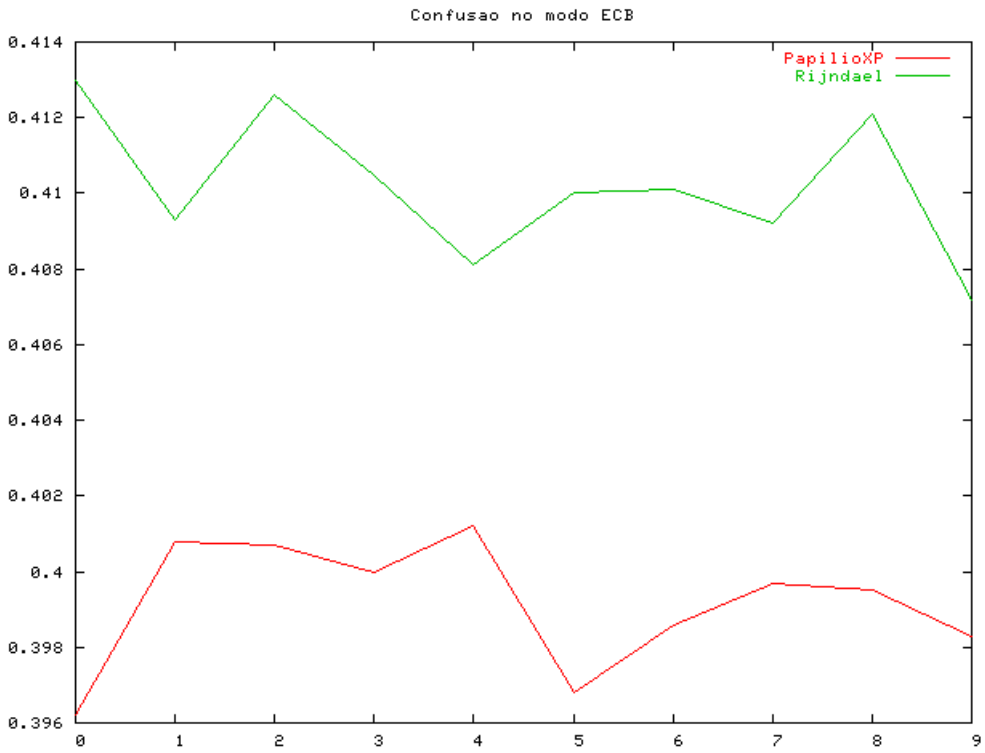

Figura 6.7: PapílioXP x Rijndael - Confusão no modo ECB

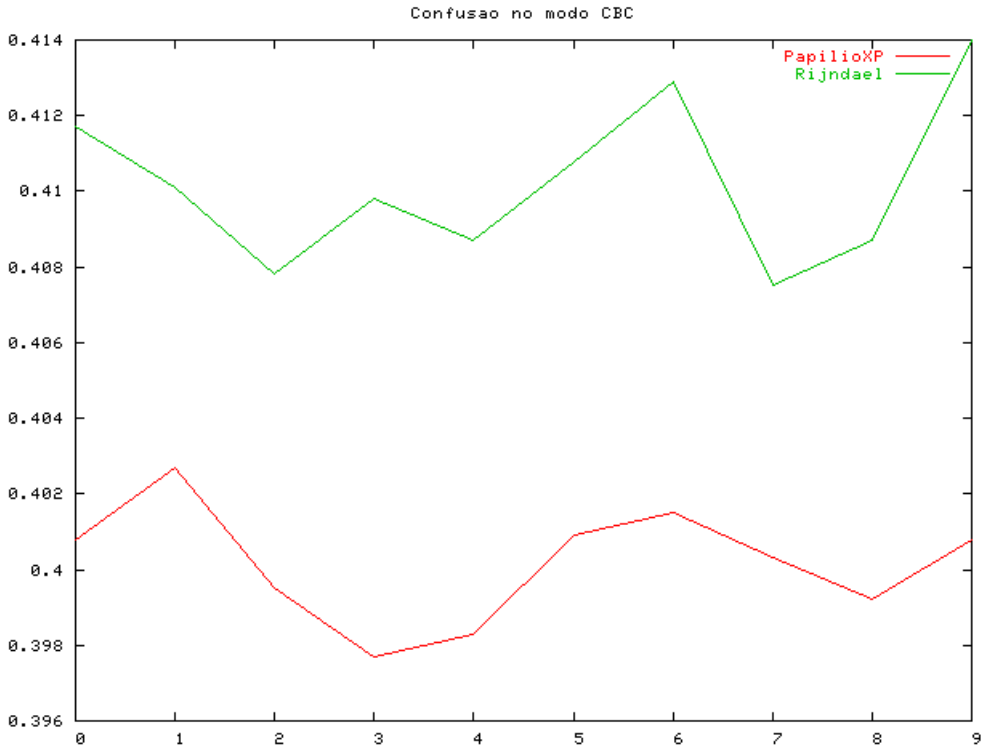

Figura 6.8: PapílioXP x Rijndael - Confusão no modo CBC

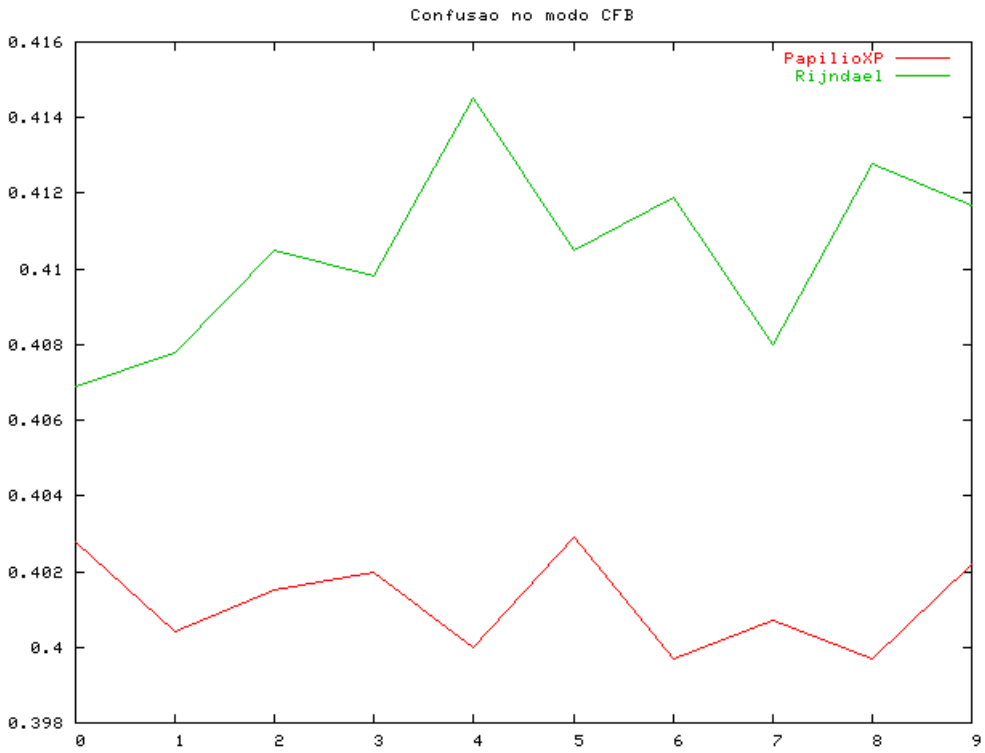

Figura 6.9: PapílioXP x Rijndael - Confusão no modo CFB

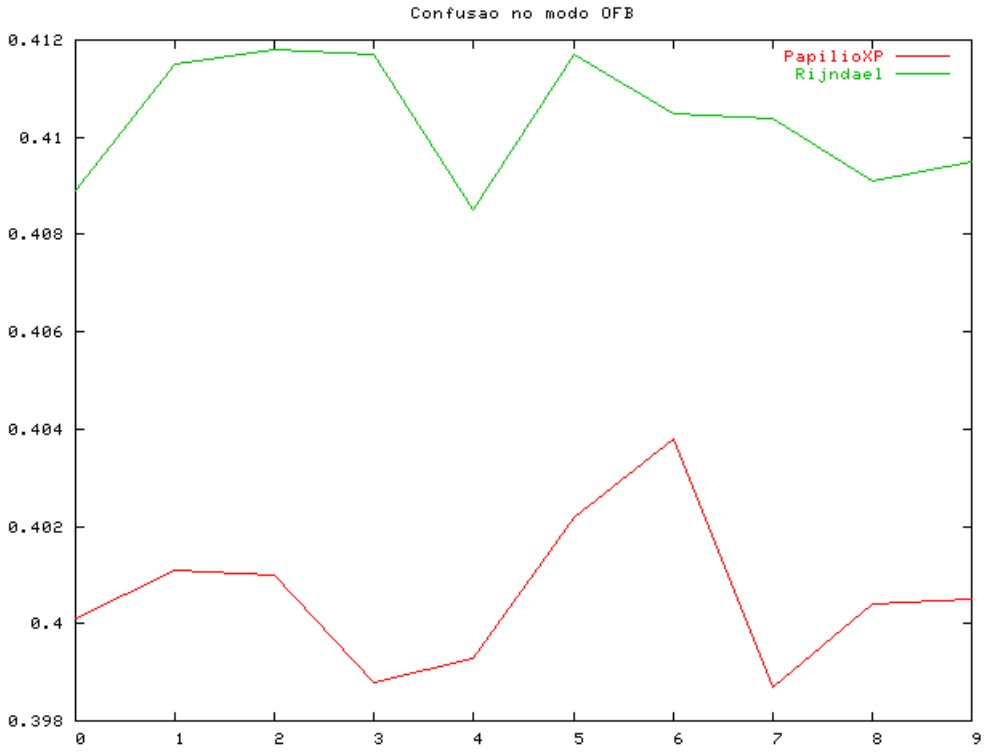

Figura 6.10: PapílioXP x Rijndael - Confusão no modo OFB

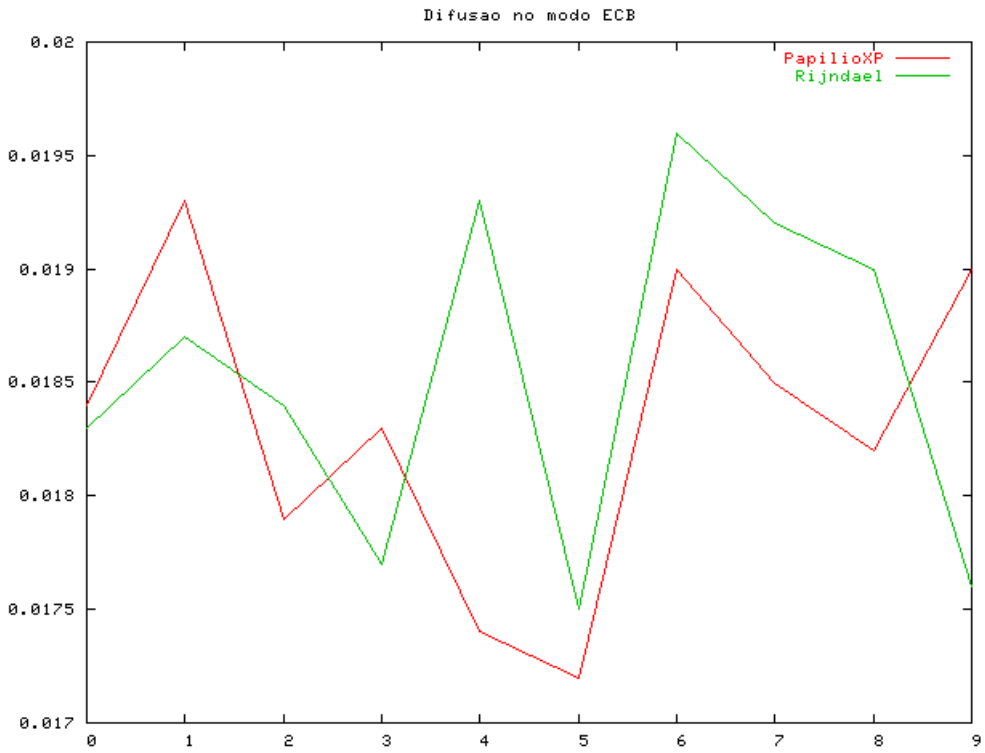

Figura 6.11: PapílioXP x Rijndael - Difusão no modo ECB

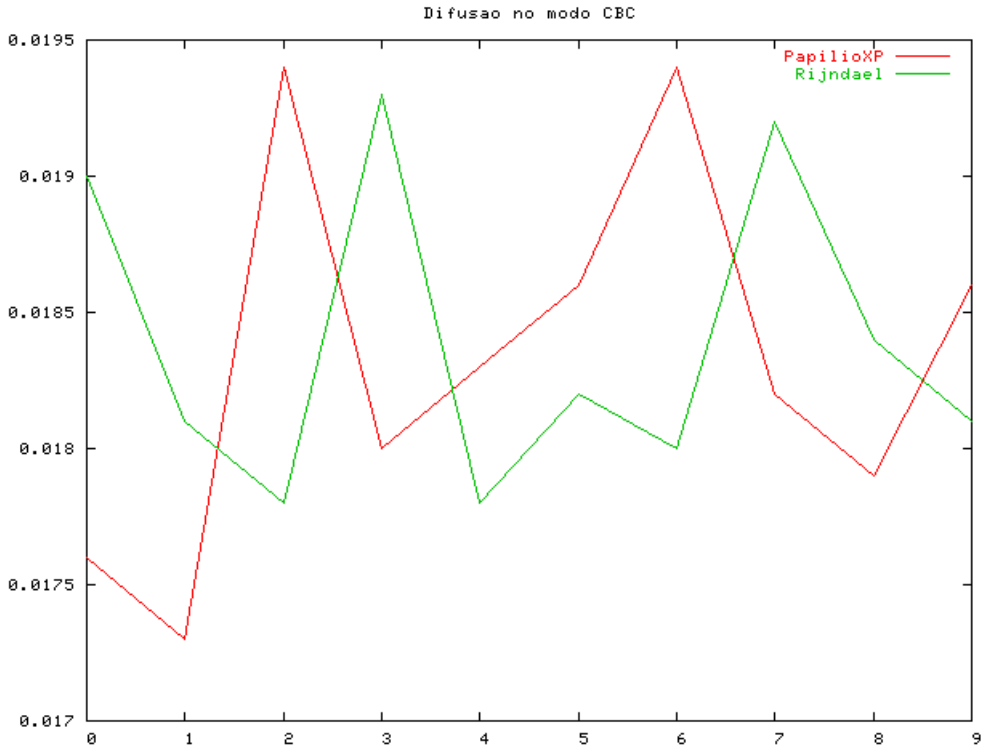

Figura 6.12: PapílioXP x Rijndael - Difusão no modo CBC

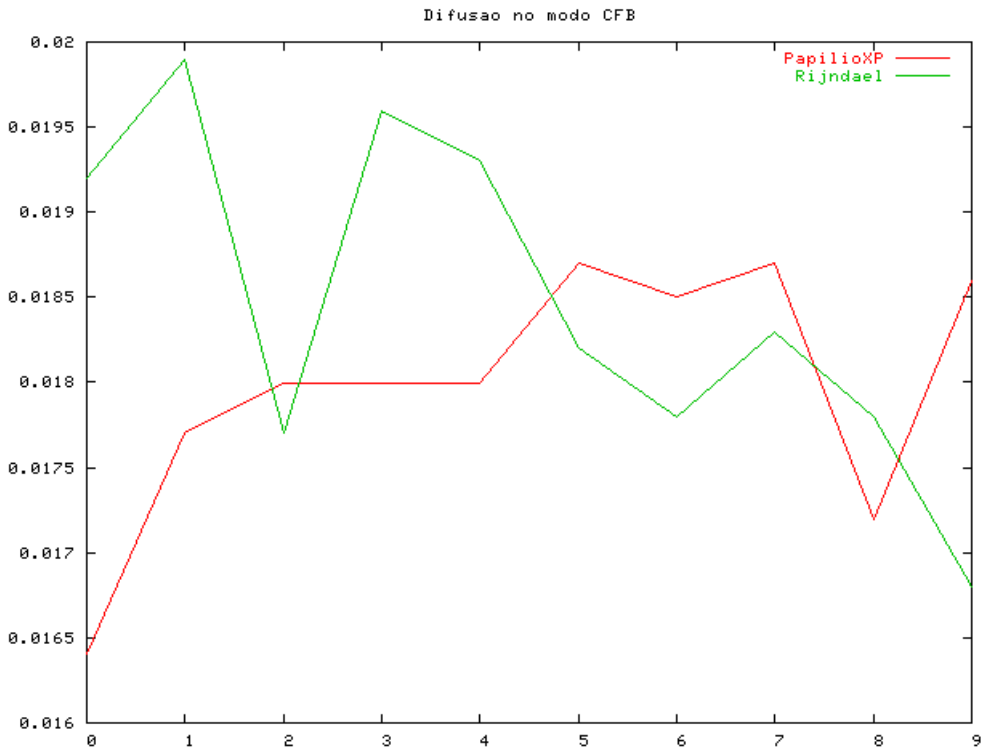

Figura 6.13: PapílioXP x Rijndael - Difusão no modo CFB

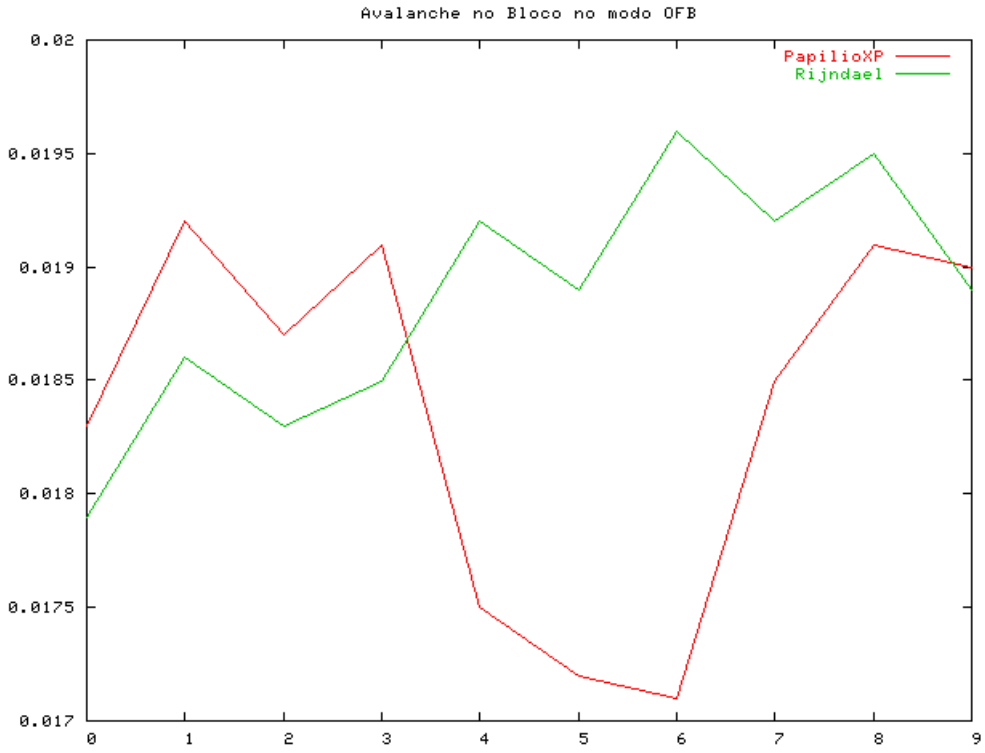

Figura 6.14: PapílioXP x Rijndael - Difusão no modo OFB

6.2.3. PapílioXP comparado ao Papílio Tradicional

No intuito de constatar até que ponto as alterações incorporadas ao PaílioXP valem a pena, ele foi confrontado com o seu predecessor o Papílio Tradicional. O confronto foi da mesma forma que o feito entre PapílioXP e Rijndael.

6.2.3.1. Análise dos resultados (mortalKombat): PapílioXP x Papílio Tradicional

Para Avalanche vemos que o PapílioXP tem desempenho sensivelmente melhor do que o Papílio Tradicional tendo um valor médio mais próximo do ideal de 50% de bits alterados nos modos ECB e CBC (figuras 6.15, 6.16). Nos modos CFB e OFB (figuras 6.17 e 6.18) podemos considerar um empate técnico.

Na análise da Confusão (ganha que tiver em torno dos 50% de dissimilaridade entre os textos) a curva do PapílioXP majora a do Papílio Tradicional, mas por uma diferença muito pequena mas que não pode ser desprezada (ver figuras 6.19, 6.20, 6.21, 6.22).

Na Difusão (ganha quem obter o menor índice) o PapílioXP se mostrou forte em relação ao Tradicional no modo ECB (veja figura 6.23). Regular no modo CBC e CFB (veja figuras 6.24, 6.25) e teve desempenho inferior ao Tradicional no modo OFB (veja figura 6.26).

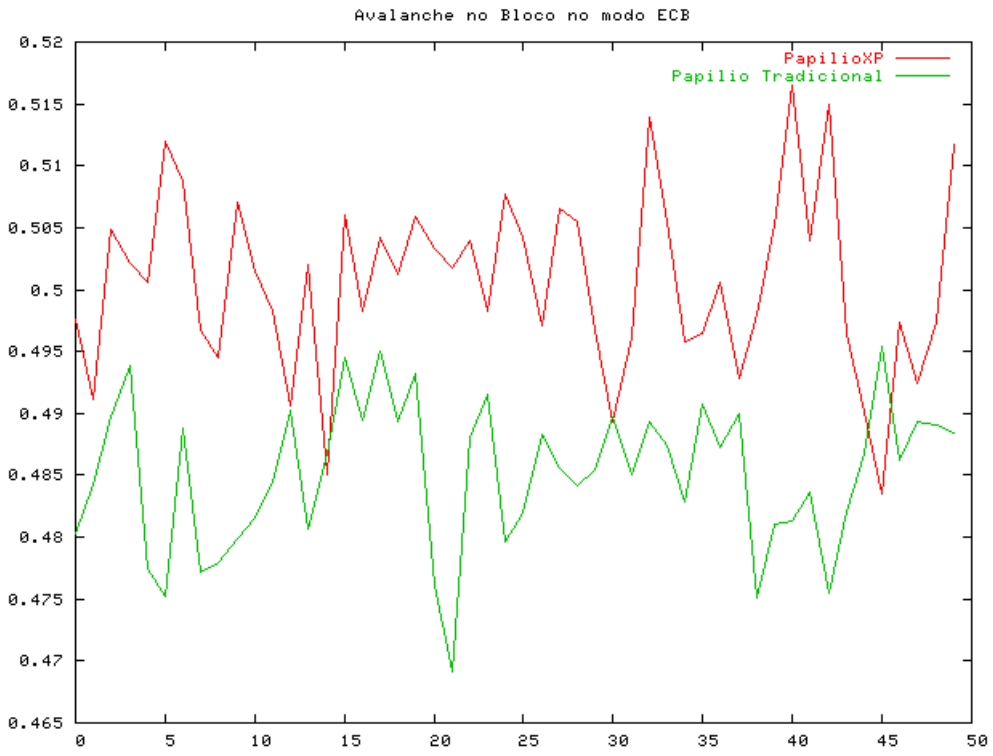

Figura 6.15: Papílio Tradicional x PapílioXP - Avalanche no Bloco no modo ECB

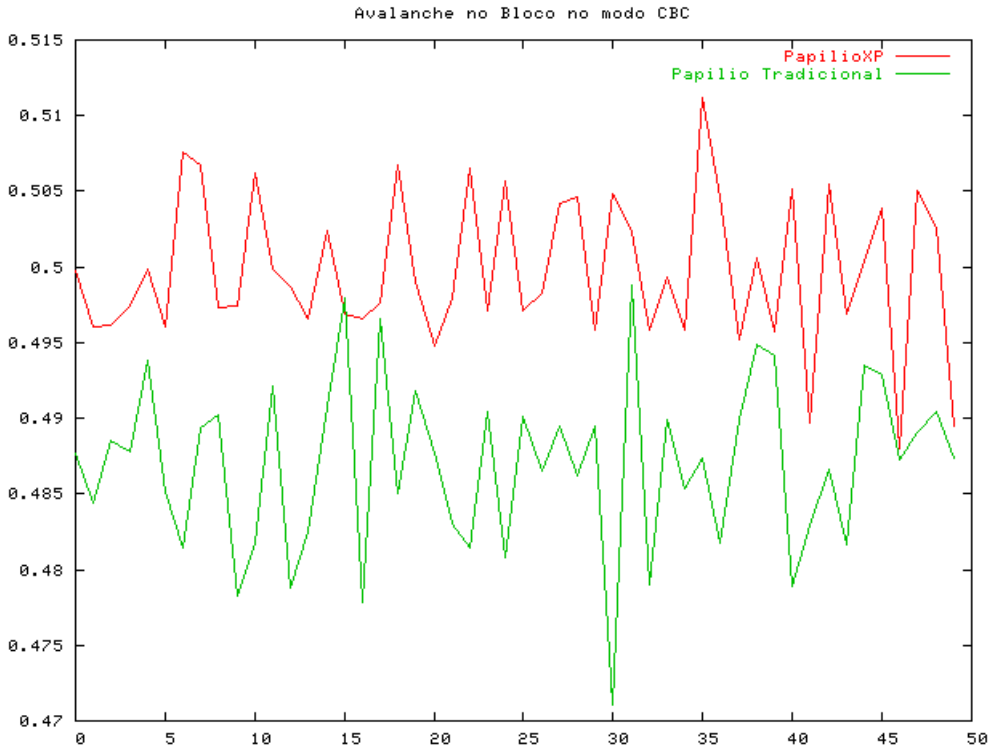

Figura 6.16: Papílio Tradicional x PapílioXP - Avalanche no Bloco no modo CBC

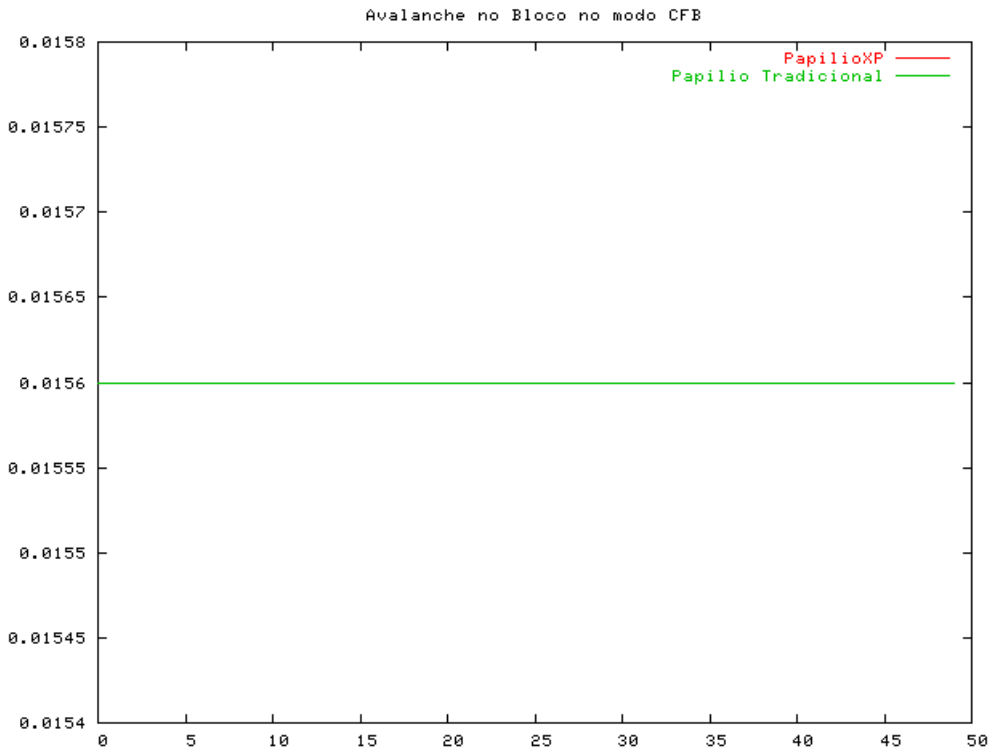

Figura 6.17: Papílio Tradicional x PapílioXP - Avalanche no Bloco no modo CFB

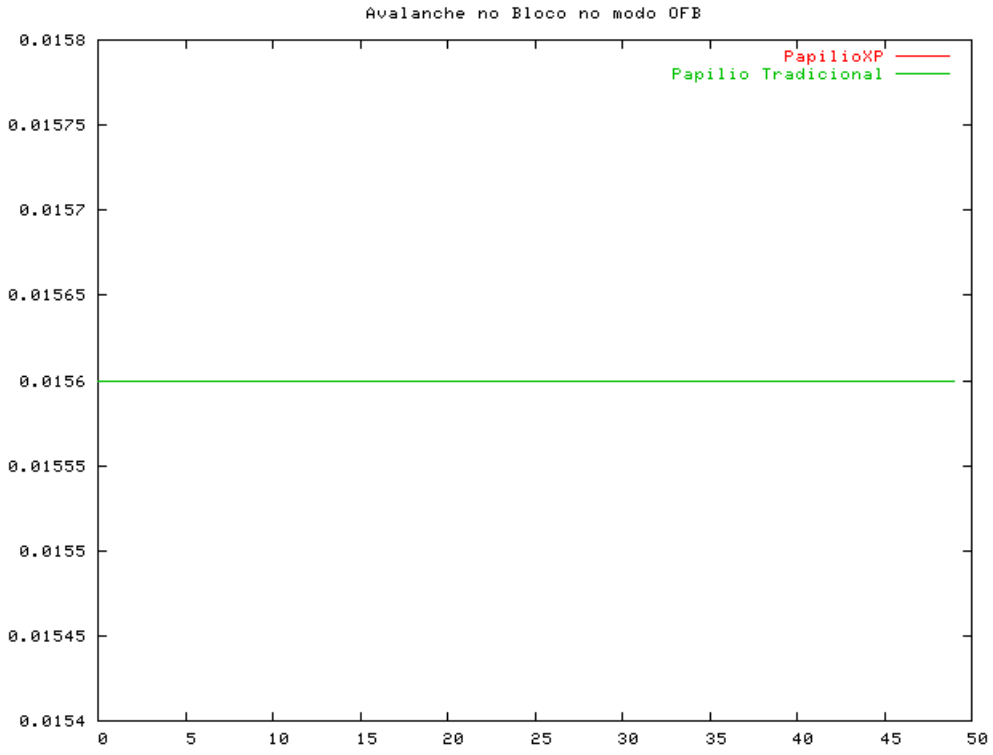

Figura 6.18: Papílio Tradicional x PapílioXP - Avalanche no Bloco no modo OFB

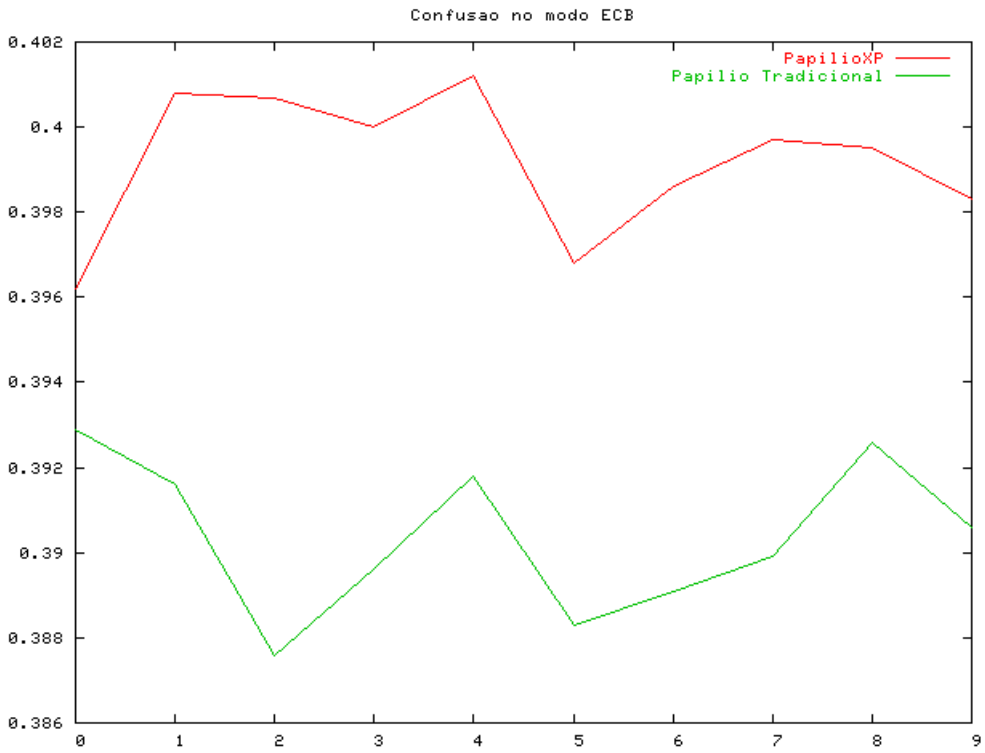

Figura 6.19: Papílio Tradicional x PapílioXP - Confusão no modo ECB

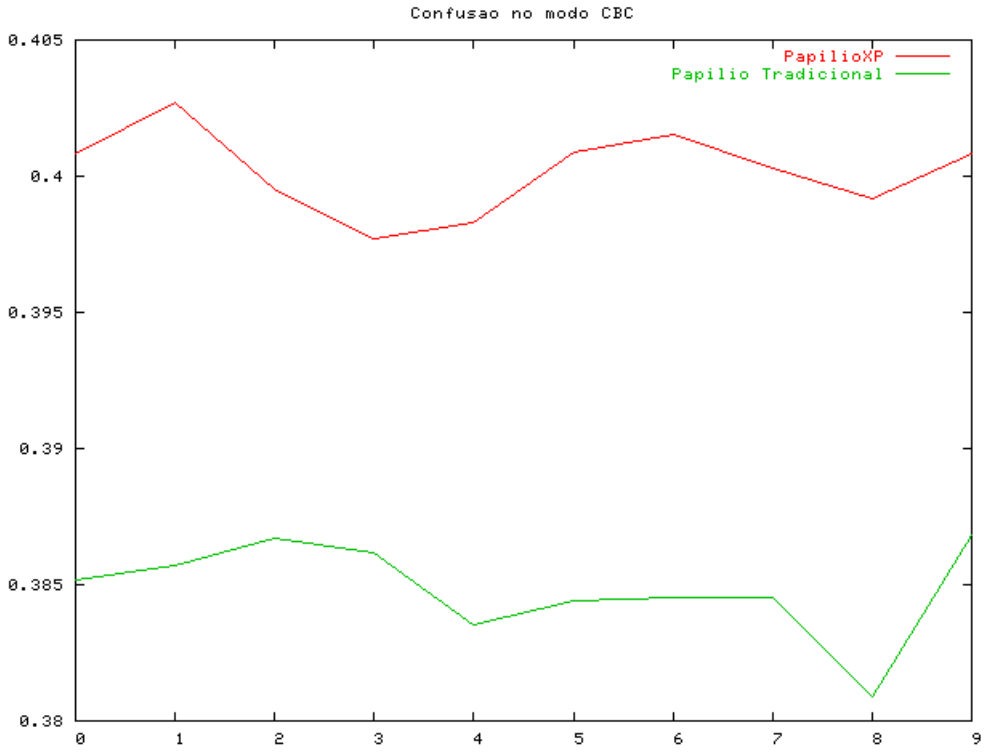

Figura 6.20: Papílio Tradicional x PapílioXP - Confusão no modo CBC

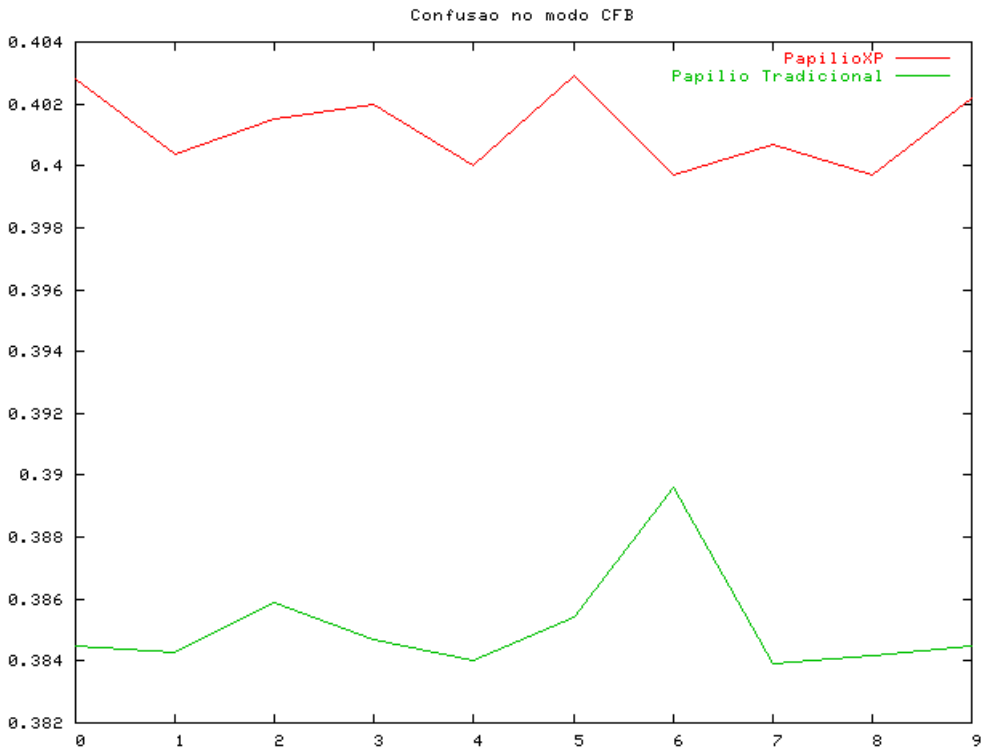

Figura 6.21: Papílio Tradicional x PapílioXP - Confusão no modo CFB

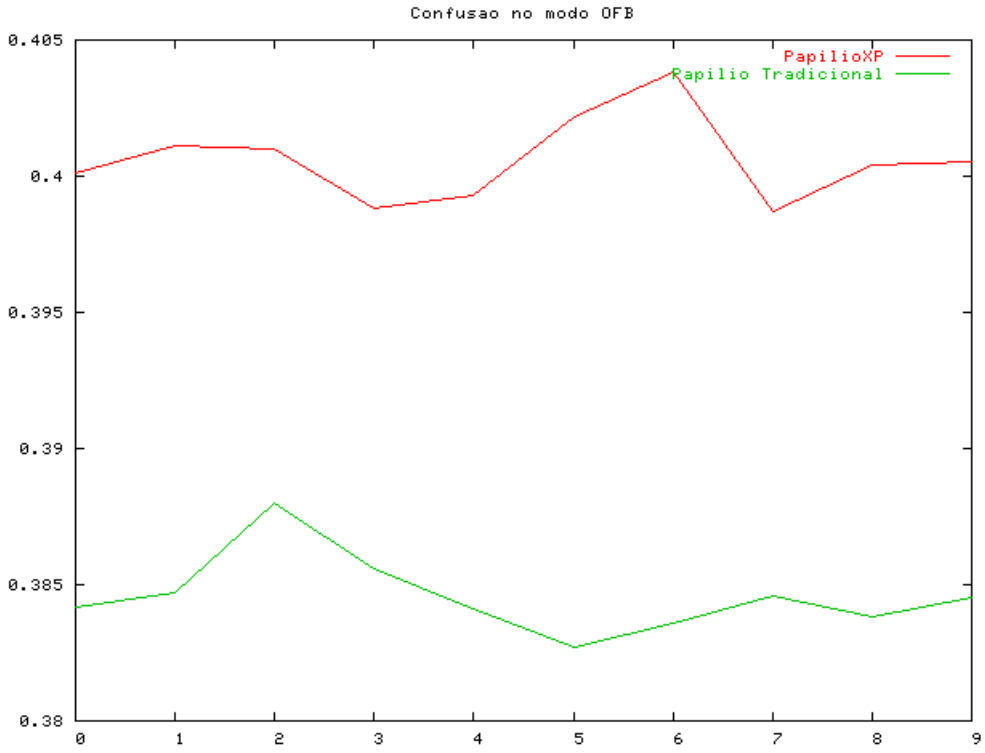

Figura 6.22: Papílio Tradicional x PapílioXP - Confusão no modo OFB

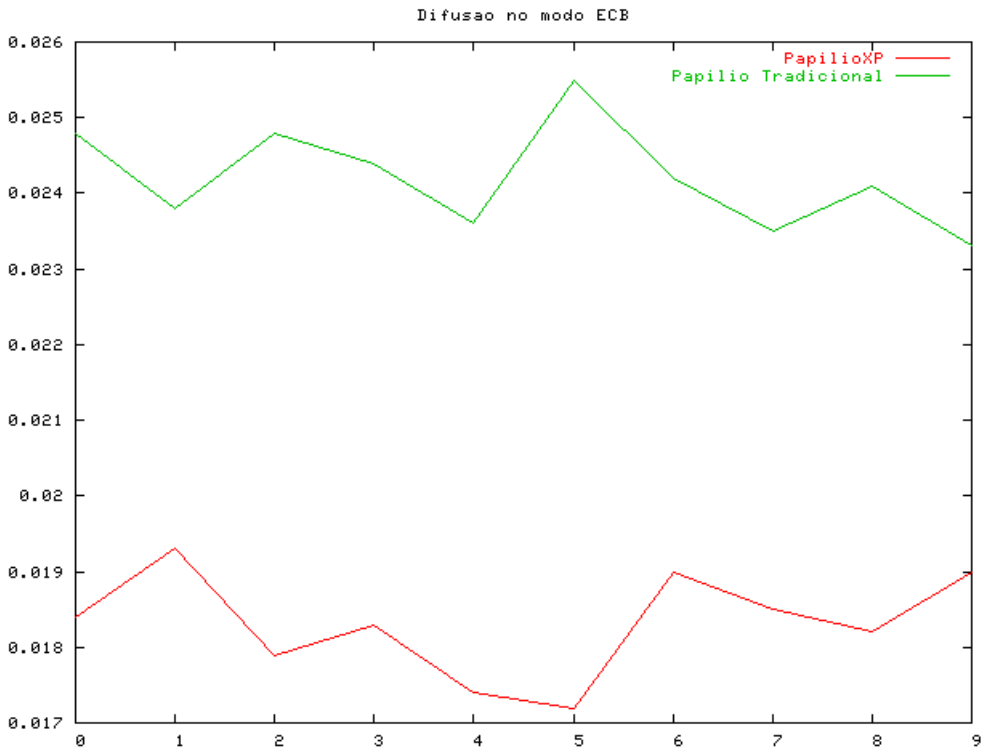

Figura 6.23: Papílio Tradicional x PapílioXP - Difusão no modo ECB

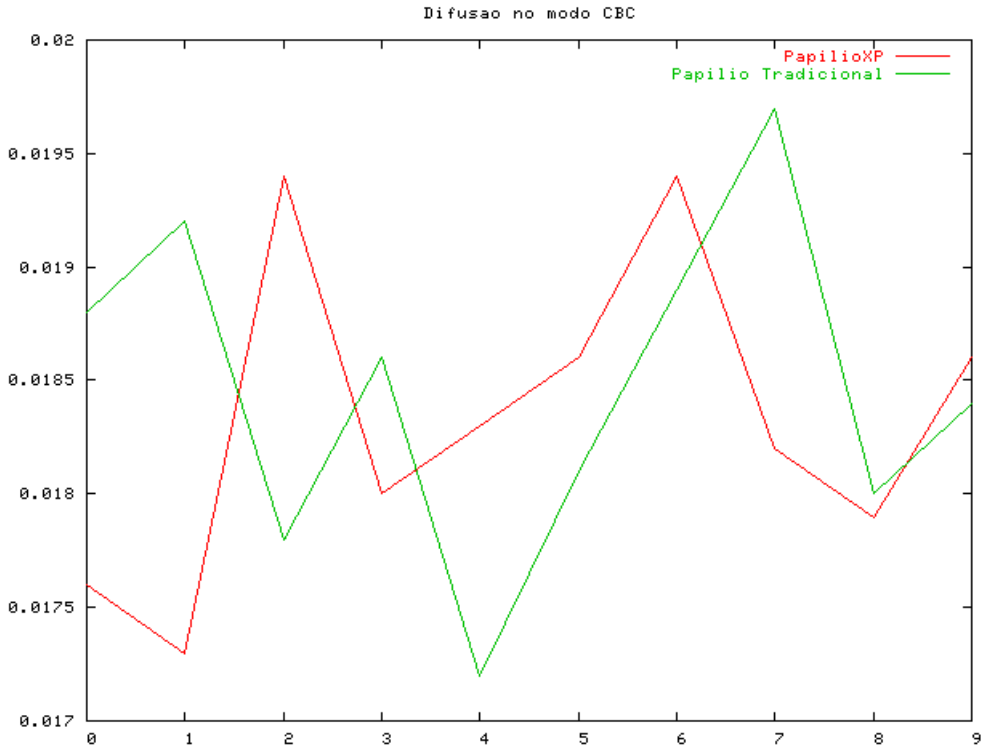

Figura 6.24: Papílio Tradicional x PapílioXP - Difusão no modo CBC

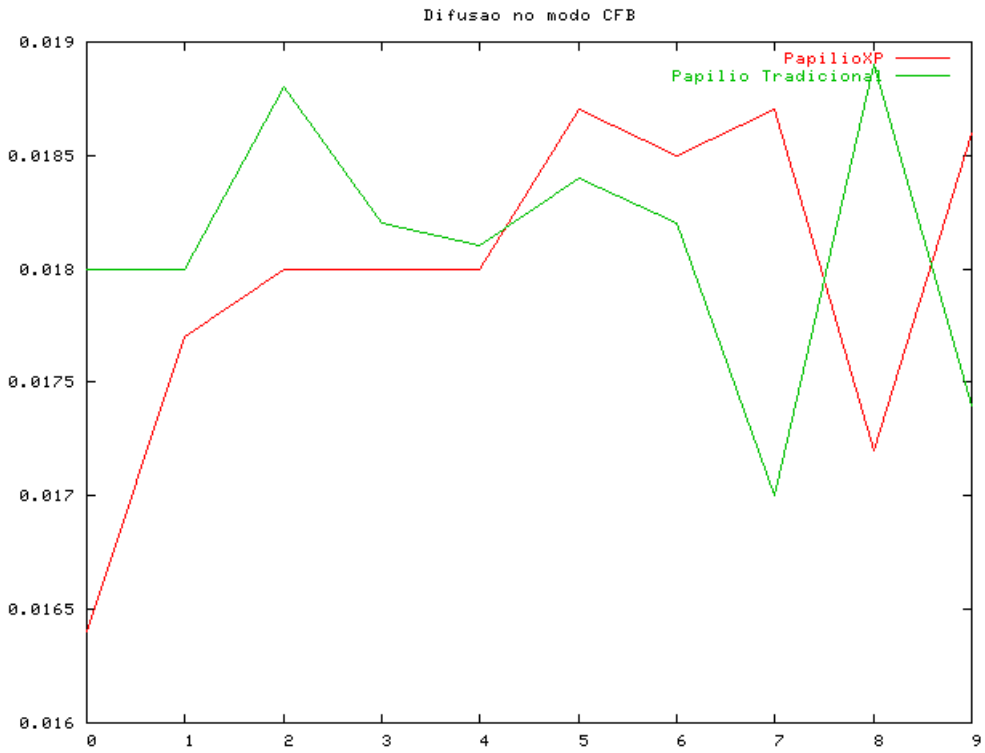

Figura 6.25: Papílio Tradicional x PapílioXP - Difusão no modo CFB

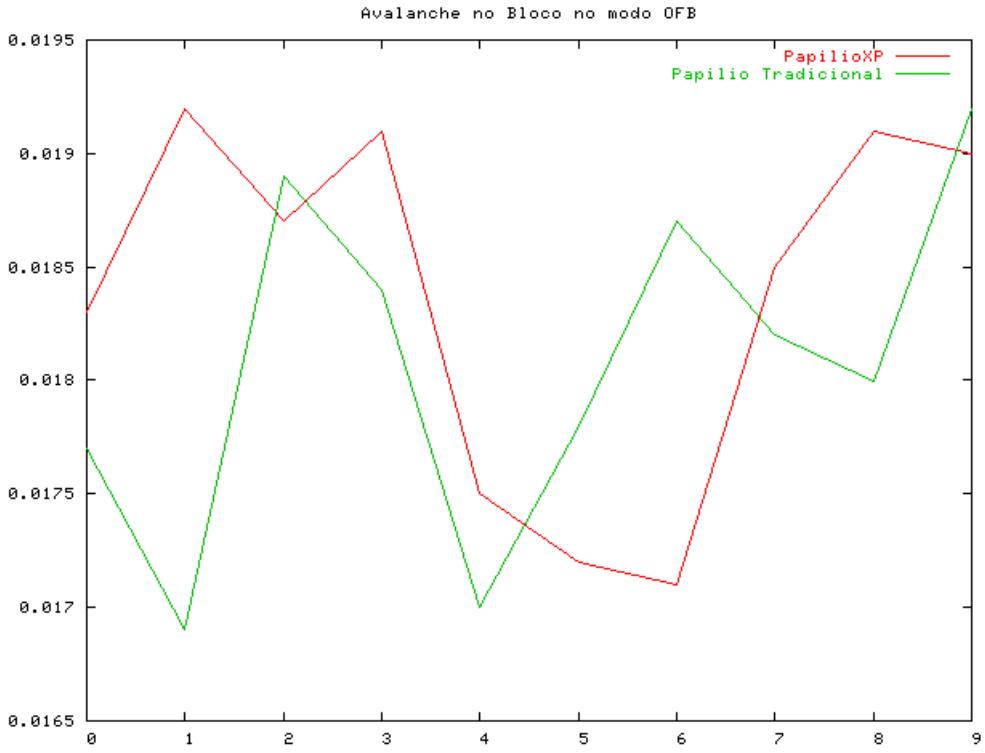

Figura 6.26: Papílio Tradicional x PapílioXP - Difusão no modo OFB

Conclusão

A idéia de seleção de Polinômios do algoritmo Papílio, de acordo com os índices Difusão, Confusão e Avalanche o confere uma sensível melhoria em relação a estes. Contudo a inovação principal incorporada ao algoritmo Papílio é o uso de um conjunto de funções de substituição que são escolhidas a cada rodada na Rede de Feistel em função do bloco que se quer codificar. A escolha do estado inicial da função Viterbi Modificado é também em função do bloco que se quer codificar na entrada. Isto conferem uma versatilidade enorme ao algoritmo permitindo que se tenha a opção de 2^{80} funções de substituição diferentes para a codificação de cada bloco. Com isso temos convicção de que o único ataque possível para se quebrar a codificação de um bloco feita por PapílioXP é a força bruta, já que a codificação é fortemente dependente do texto de entrada, que é o segredo que se quer descobrir.

Trabalhos Futuros

Uma das extensões propostas para Papílio é o número de rodadas variado, o que permite a se optar por um processo de ciframento mais rápido e com uma boa segurança (aplicações domésticas) ou um processo um pouco mais lento, mas com alto grau de segurança (aplicações comerciais e militares). Outra extensão é a procura de outros conjuntos de máquinas de estado, o que pode resultar em Papílios personalizados ou Papílios mais fortes caso se amplie o tamanho do conjunto de máquinas de estado que o PapílioXP gerencia. Por último seria interessante ao Papílio incorporar idéias inseridas pelo Rijndael como comprimento do bloco variável e tamanho da chave variável, o que o tornaria extremamente versátil oferecendo uma gama de opções para operação.

Bibliografia

[SZW]: SZWARCFITER, Jayme; MARKENZON, Lilian. **Estruturas de Dados e Seus Algoritmos**. Rio de Janeiro, LTC, 1994.

[JAM]: SPILKER, James J. Jr. **Digital Communications by Satellite**. Englewood Cliffs, New Jersey. Prentice – Hall. 1977.

[SCN]: SCHNEIER, Bruce. **Applied Cryptography Second Edition: protocols, algorithms, and source code in C**. USA. John Wiley & Sons, inc. 1996.

[RAM]: RAMOS, Karla, Darlene Nepomuceno. **Proposta de um Algoritmo de Criptografia Baseado no Algoritmo Viterbi e Codificação Convolucional**. Dissertação de Mestrado. UFRN. 2002.

[GLO]: FERNANDES, Francisco; LUFTY, Celso; GUIMARÃES, Marques. **Dicionário Brasileiro Globo**. São Paulo. Globo. 1991

[NUM]: **numaboa**. Acessado em 26 de julho de 2003
http://www.numaboa.com/criptologia

[CIE]: **cinenciaj**. Acessado em 26 de julho de 2003
http://www.ajc.pt/cienciaj/n32/escrita.php

[OPE]: **Projeto OpenSSL**
Copyright (c) 1998-2000 The OpenSSL Project, http://www.openssl.org/
Author: OpenSSL Project (openssl@openssl.org)
Modified: 2002-04-17 16:00:05.
Generated from ``index.wml'' via WML 2.0.9 (18-Oct-2002).
by OpenSSL Project (openssl@openssl.org)
on 2003-04-10 23:17:48.

[GUT]: Relação dos 50 Livros usados nos testes Difusão e Confusão, provenientes do projeto **Gutemberg**. Acessado em - 17-05-2003
http://www.veritel.com.br/gutenberg/etext02/

03tom10.txt - Tom Swift and His Airship, Victor Appleton
1mlaz10.txt - The Poems of Emma Lazarus, Vol.I, Narrative, Lyric, and Dramatic, Emma Lazarus
1ward10.txt - The Complete Works of Artemus Ward, Part 1, Charles Farrar Browne
2cpns10.txt - The Two Captains, La Motte-Fouque, Friedrich Heinrich Karl, Freiherr de
2ndsm10.txt - The Second-Story Man, Upton Sinclair
42pom10.txt - Forty-Two Poems, James Elroy Flecker
acdbh10.txt - Augustus Does His Bit, George Bernard Shaw
atblf10.txt - The Author of Beltraffio, Henry James
birds10.txt - The Birds, Aristophanes
bncor10.txt - Mr. Bonaparte of Corsica, John Kendrick Bangs
cambp10.txt - Samuel Butler's Cambridge Pieces, Samuel Butler
cvrly10.txt - DAYS WITH SIR ROGER DE COVERLEY, JOSEPH ADDISON and RICHARD STEELE
cwanp11.txt - American Newspaper, Charles Dudley Warner
cwcdc11.txt - Causes of Prevailing Discontent, Charles Dudley Warner
cwfpg11.txt - Mr. Froude's Progress, Charles Dudley Warner
cwins11.txt - Indeterminate Sentence, Charles Dudley Warner
cwlcr10.txt - Literary Copyright, Charles Dudley Warner
cwpoc10.txt - The Story of Pocahantas, Charles Dudley Warner
cwsne10.txt - How Spring Came in New England, Charles Dudley Warner
drkga10.txt - Drake's Great Armada, Walter Biggs
gartt10.txt - The Garotters, William D. Howells
glads10.txt - William Ewart Gladstone, James Bryce
gratc10.txt - Great Catherine, George Bernard Shaw
hnmtl10.txt - Old Love Stories Retold, Richard Le Gallienne
incap10.txt - The Inca of Perusalem, George Bernard Shaw
jptwn10.txt - THE JAPANESE TWINS, Lucy Fitch Perkins
lastt10.txt - The Last Stetson, John Fox Jr.
lied210.txt - How He Lied to Her Husband, George Bernard Shaw

locrn10.txt - Locrine - A Tragedy, Algernon Charles Swinburne

mt6lt10.txt - Letters Vol. 6, Mark Twain

mtbbg10.txt - A Burlesque Autobiography, Mark Twain

mtcrg11.txt - Curious Republic of Gondour, Mark Twain

mtdtl10.txt - A Dog's Tale, Mark Twain

mtfco10.txt - Fenimore Cooper Offences, Mark Twain

mthts10.txt - How Tell a Story and Others, Mark Twain

mtpbg11.txt - Essays on Paul Bourget, Mark Twain

mtswe10.txt - The Stolen White Elephant, Mark Twain

nb10v10.txt - Memoirs of Napoleon Bonaparte, V10, Louis Antoine Fauvelet de Bourrienn

nwthp10.txt - New Thought Pastels, Ella Wheeler Wilcox

oflvc10.txt - O'Flaherty V. C., George Bernard Shaw

ouamc10.txt - Our American Cousin,Tom Taylor

plghp10.txt - The Pilgrims of Hope, William Morris

pmprg10.txt - Poems of Progress, Ella Wheeler Wilcox

truth10.txt - Truth and the Myth, A.R. Narayanan

whelv10.txt - The Elevator, William D. Howells

whtyc11.txt - The Young Contributor, William Dean Howells

whvan10.txt - Notes of a Vanished Summer, William Dean Howells

whvne11.txt - First Visit to New England, William Dean Howells

wldam10.txt - Wild Animals I Have Known, Ernest Thompson Seton

wnohl10.txt - Within an Inch of His Life, Emile Gaboriau