Isaac de Lima Oliveira Filho

Criptoanálise Diferencial do Papílio.

Natal – RN

Fevereiro / 2010

Isaac de Lima Oliveira Filho

Criptoanálise Diferencial do Papílio.

Dissertação apresentada à Coordenação do Mestrado em Sistemas e Computação da Universidade Federal do Rio Grande do Norte para a obtenção do título de Mestre em Sistemas e Computação.

Orientador:
Prof. Dr. Benjamín René Callejas Bedregal

MESTRADO EM SISTEMAS E COMPUTAÇÃO
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Natal – RN

Fevereiro / 2010

Dissertação de Mestrado sob o título *"Criptoanálise Diferencial do Papílio."*, defendida por Isaac de Lima Oliveira Filho e aprovada em 26 de Fevereiro de 2010, em Natal, Rio Grande do Norte, pela banca examinadora constituída pelos doutores:

Prof. Dr. Benjamín René Callejas Bedregal
Departamento de Informática e Matemática Aplicada - DIMAP - UFRN.
Orientador

Prof. Dr. Ivan Saraiva Silva
Departamento de Informática e Matemática Aplicada - DIMAP - UFRN.

Prof[a]. Dr[a]. Karla Darlene Nepomuceno Ramos
Departamento de Computação - DI - Campus de Natal - UERN.

Prof. Dr. Ruy José Guerra Barretto de Queiroz
Centro de Informática - UFPE.

Prof. Dr. João de Deus Lima
Departamento de Matemática e Estatística - Campus Central - UERN.

Dedico esta dissertação a meus pais,
cujo exemplo de honestidade e trabalho
tem sido norteador para a minha vida,
e a minha noiva, pelo apoio nos momentos mais
difíceis e também pela confiança e principalmente a esperança.

Agradecimentos

A execução deste trabalho contou com ajuda de muitas pessoas:

Primeiramente a Deus e aos meus pais, pelo incondicional amor e apoio em todos os momentos de minha vida nas horas que mais precisei.

Ao meu orientador Benjamín René Callejas Bedregal, com o qual aprendi muito e muito fez por mim, pela paciência e insistência necessária para a conclusão deste trabalho.

Aos Professores, que me ensinaram e me guiaram ao longo deste mestrado.

A minha noiva Lisa Cristina, pelo carinho, apoio, esforço e muita paciência, para o término deste trabalho.

Aos funcionários do DIMAP, pelas boas condições de trabalho dadas de forma a permitir a conclusão deste trabalho.

Ao meu amigo Cleverton Hentz, pelo auxílio na implementação dos métodos requeridos e pela partilha dos seus conhecimentos para conclusão deste trabalho.

Aos colegas em especiais, Cássio, Anchieta e João Paulo, pelos conselhos dados.

Ao professor Ivan Saraiva pela concessão do laboratório para fins da realização do trabalho.

Aos demais colegas de mestrado, pelo convívio harmonioso durante o período do mestrado.

“A codificação é como estudar uma língua estrangeira. No início, o texto parece incompreensível, mas aos poucos você aprende as regras que definem sua estrutura e começa a extrair o sentido.”

Fortaleza Digital - Dan Brown

Resumo

Neste trabalho será aplicada a técnica de Criptoanálise Diferencial, introduzida por Biham e Shamir, sobre o algoritmo de criptografia Papílio, desenvolvido por Karla Ramos, a fim de testar e, principalmente, provar sua relevância em relação a outras cifras de blocos como DES, Blow-Fish e FEAL-N(X). Esta técnica tem por base a análise das diferenças entre os pares de textos claros e a diferença entre as suas respectivas cifras, em busca de padrões que auxiliarão nas descobertas das sub chaves e consequentemente na descoberta da chave mestra, na qual está a segurança da cifra. Estas diferenças são obtidas através de operações XOR. Busca-se, com esta análise, além da obtenção de padrões do Papílio, obter-se também as principais características e o comportamento do Papílio durante seus 16 ciclos, identificando e substituindo quando necessário, os fatores que podem ser melhorados de acordo com as definições pré estabelecidas do mesmo, para oferecer maior segurança na utilização de sua cifra.

Abstract

In this work will applied the technique of Differential Cryptanalysis, introduced in 1990 by Biham and Shamir, on Papílio's cryptosystem, developed by Karla Ramos, to test and most importantly, to prove its relevance to other block ciphers such as DES, Blowfish and FEAL-N (X). This technique is based on the analysis of differences between plaintext and theirs respective ciphertext, in search of patterns that will assist in the discovery of the subkeys and consequently in the discovery of master key. These differences are obtained by XOR operations. Through this analysis, in addition to obtaining patterns of Papílio, it search to obtain also the main characteristics and behavior of Papilio throughout theirs 16 rounds, identifying and replacing when necessary factors that can be improved in accordance with pre-established definitions of the same, thus providing greater security in the use of his algoritm.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O algoritmo de criptografia desenvolvido por Karla Ramos ([Ramos 2002]) foi proposto como um eficiente algoritmo de criptografia. Para que esse fato fosse comprovado, o Papílio necessitaria submeter-se a testes para avaliar seu desempenho. Estes testes, que já foram aplicados por outras cifras, demonstraram as qualidades e fraquezas dos algoritmos de criptografia. Estes testes são necessários para mostrar a eficiência do Papílio em relação as demais cifras de blocos, identificando em que pontos o Papílio é melhor ou não dos que as outras cifras.

Os primeiros testes e resultados obtidos acarretaram melhorias ao algoritmo original, os testes foram realizados por Frederiko Estênio (referencia sobre Frederiko) em relação ao modos de operação do Algoritmo e quanto a premissas básicas de qualquer algoritmo, que é a Confusão, Difusão e Avalanche, resultando em significativas melhorias para o mesmo. Contudo ainda não foi possível destacar seu algoritmo e sua eficiência das demais cifras de blocos.

1.1 Motivação

A expectativa com este trabalho é determinar se o algoritmo Papílio é uma cifra forte perante as demais. Esta dissertação abrange uma técnica conhecida com Criptoanálise diferencial, que foi utilizada nos algoritmos de criptografia de mais destaque nas comunidades acadêmicas. Com esta técnica é possível descobrir as sub chaves utilizadas utilizadas através das diferenças dos textos que serão criptografados com seus respectivos resultados pós cifragem. Com esta técnica é possível analisar, comparar e obter significantes resultados sobre o perfil, ou mesmo propriedades do algoritmo de forma a implementar melhorias no seu funcionamento.

1.2 Objetivo

Analisar o algoritmo de Criptografia Papílio em relação as outras cifras de blocos, que tenham o algoritmo semelhante ao do Papílio e através das criptoanálises diferenciais aplicadas

sobre os algoritmo DES, BlowFish e NEAL, comparar as características, propriedade, performance, complexidade de quebra de cifra em relação ao Papílio.

Com os resultados obtidos da criptoanálise diferencial sobre o Papílio, será possível determinar em que pontos o Papílio é melhor ou não do que as outras cifras, e assim será possível inserir ao Papílio melhorias, e conseqüentemente colocar o algoritmo do Papílio em uma escala de mais complexidade de seu funcionamento, proporcionando assim uma cifra forte, rápida e robusta.

1.3 Estrutura do Trabalho

Após as considerações iniciais sobre criptografia, algoritmos cifradores de blocos como o DES, e alguns conceitos chaves básicos como chaves, plaintext, caixas S, ciclos e etc., os próximos capítulos serão abordados da seguinte forma, o capítulo 3 abordará a apresentação do algoritmo Papílio, o qual será feito a Criptoanálise Diferencial, no Capítulo 4 será apresentado a técnica de Criptoanálise Diferencial, com resultados referidos ao DES, no Capítulo 5 será apresentado a Criptoanálise Diferencial sobre o Papílio, no entanto resultados e demais trabalhos serão apresentados no Capítulo 6, e em seguida um capítulo com a conclusão deste trabalho.

2 *Definições*

> It is possible to build a cabin with no foundations but not a lasting building.
>
> Eng. Isidor Goldreich (1906-1995)

O homem ao longo do tempo, sentiu a necessidade de proceder com cuidado e de maneira sigilosa suas informações, tenderam desenvolver métodos e técnicas para confidencializar suas mensagens, como foi demonstrado ao longo do tempo e da historia, e.g, na Grécia em meados do século VI AC, utilizou-se para encriptar, ou mesmo para confidencializar parte do conteúdo da mensagem, uma vara chamada *scylate*.

Nesta técnica o emissor grava uma mensagem em um pedaço de papel que estava preso de forma particular ao longo da vara com comprimentos especificados. Somente alguém que tivesse uma vara com as mesmas proporções conseguiria visualizar a mensagem de maneira correta. Neste caso, a chave é conhecida somente pelo emissor e o destinatário e satisfaz um dos principais fundamentos da criptografia: a segurança da criptografia utilizada deve ser diretamente proporcional à segurança da chave que é utilizada para encriptar um texto.

Seguindo este exemplo, o que poderia ser feito por uma terceira pessoa que conhecesse o método mas não conhecesse o valor (tamanho) da chave, para ela decifrar uma mensagem? Pode-se tirar uma conclusão sobre este exemplo: a qualidade e segurança de um algoritmo criptográfico não está somente no segredo do seu algoritmo, mas sim na complexidade da sua chave, ou seja, quanto mais difícil ou impossível for descobrir a chave utilizada por uma determinada cifra, mais forte e segura a mesma será. Neste caso a chave a ser quebrada seria o comprimento e o diâmetro da vara.

Com o surgimento e forte utilização de meios computacionais para a disseminação de conteúdos, e suas respectivas importâncias, mostrou-se necessário a questão de defender estes conteúdos dos invasores de maneira concreta e segura. Então o relacionamento da criptografia com sistemas computacionais tiveram seu relacionamento diretamente proporcional quanto à performance adquirida pelo sistema no que se diz um “sistema completamente seguro”. Com

isso tem-se o surgimento de diversos tipos de algoritmos criptográficos e suas respectivas formas de obter sucesso nessas invasões contra esses algoritmos. Mas para melhor entender os conceitos que serão posteriormente definidos ao longo deste trabalho, far-se-á necessário primeiramente algumas definições básicas, como as seguintes:

- Texto Branco - Conhecidas também como *plaintext*, ou seja, textos sem qualquer tipo de criptografia.
- Texto Cifrado - Conhecido também como *ciphertext*, ou seja, textos resultantes do processo de criptografia aplicada por algum sistema criptográfico.
- Cifra - São resumidamente chamados os algoritmos de criptografia que podem ser ou não cifras de blocos[1],e.g DES (*Data Encryption Standard*) em [Konheim 2007], FEAL (*Fast Data Encipherment Algorithm*) em [Nakaraha 2006], AES (*Advanced Encryption Standard*) em [Konheim 2007].
- Chave - Os algoritmos criptográficos não serão tão seguros se suas respectivas chaves forem conhecidas, em [Corporation, Adler e Gailly 2004] uma chave é um valor que funciona com um algoritmo criptográfico para produzir um determinado texto cifrado. As chaves são geralmente grandes números e de uma grande complexidade e são aplicada de maneira à protege-la. O seu tamanho é dado em quantidade de bits.
- Encriptar - Conhecida também como codificação. As cifras são responsáveis por encriptar um *plaintext* em um *ciphertext* utilizando as chaves e uma cifra, esse processo é conhecido como encriptar um texto.
- Decriptar - Conhecida também como decodificação. As mesmas cifras que são responsáveis por encriptar um *plaintext* em um *ciphertext* utilizando as chaves também são utilizadas para decriptar um *ciphertext* em um *plaintext*, sem a necessidade de outro algoritmo para fazê-lo. Algoritmos com essas características são conhecidos como *algoritmos de chave simétrica*.
- Cripto sistema - Um algoritmo criptográfico, acrescido de todas as chaves possíveis e todos os protocolos que fazem este trabalho compreende um *cripto sistema*.

Como observado no exemplo do *scylate* a necessidade de quebrar a chave seria descobrir as dimensões exatas da vara, e isso poderia ser caracterizado com um tipo de ataque. Segundo [Nakaraha 2006, NAK06 apud De88] quebrar uma cifra:

[1] Os de blocos mapeiam cada bloco de *n* bits de texto claro e uma chave num bloco de n bits de texto cifrado.

> "Significa deduzir ou obter alguma informação sobre o texto legível (total ou parcialmente), a partir de alguma informação sobre o texto cifrado correspondente, sem o conhecimento da chave, ou então, obter (total ou parcialmente) a chave criptográfica utilizada em um par (texto legível-texto cifrado) a partir do conhecimento deste par. Segundo o contexto da análise deve ser possível determinar qual dos parâmetros está sendo procurado, o texto legível ou a chave."

O tipo de cifra que se usa na encriptação de um texto é extremamente importante. As cifras podem ser classificadas como: *incondicionalmente segura* caso não seja possível ser quebrada, ou seja, independente dos meios ou recursos necessários para o mesmo; *computacionalmente segura* que apesar de não ser como a cifra classificada anteriormente, ela é segura independente dos recursos reais, e.g os computadores com poder de processamento relativamente alto ou mesmo recursos financeiros, mantendo-se segura em um tempo determinado. Contudo, cada cifra de fato, antes de ser exposta para sua real utilização, passa através de vários processos e ataques para saber o tamanho ou mesmo se é considerada fraca. Por exemplo, antes de lançarem o DES como sendo uma cifra de blocos poderosa, desenvolvida pela IBM, e antes de ser definitivamente utilizada pela National Bureau of Standards, foi realizado um grande número de testes para sua aprovação (mais detalhes sobre estes testes ver [Smid. e Branstad 1998]).

Estes testes envolveram padrões de segurança que na época não eram superáveis, como o tamanho da chave, a complexidade da cifra, os textos escolhidos e os hardwares utilizados para uma tentativa de quebrar a chave. O status de confiabilidade da cifra também depende de como o ataque é feito e qual seu objetivo. No caso de um possível intruso conhecer o algoritmo da cifra, pode-se tentar quebrar a chave através dos seguintes ataques:

- Somente texto ilegível conhecido - Ataque passivo, onde o intruso apenas tem o conhecimento das mensagens recebidas pelo destinatário.
- Texto legível conhecido - Este ataque é considerado passível, mas, diferentemente, neste caso o intruso também tem alguns textos legíveis do emissor, podendo assim compará-los.
- Texto legível escolhido - Ataque ativo, pelo fato do intruso ter as condições de escolher um determinado texto legível cifrado e possa utilizar o dispositivo criptográfico através da indução de um remetente com uma chave particular.

Quanto menos o intruso souber dos padrões da chave utilizada por determinada cifra, mais segura será seu algoritmo criptográfico. Existem inúmeros ataques que podem quebrar uma determinada cifra, como por exemplo o ataque por procura exaustiva ou mesmo uma criptoanálise diferencial, estes métodos serão abordados no decorrer do trabalho.

2.1 Características de uma Cifra de Blocos

Os algoritmos criptográficos devem ter algumas características básicas para que sejam considerados uma cifra realmente forte ou inquebrável, por exemplo, ao estudarmos uma determinada cifra devemos primeiramente ver como está implementado seu algoritmo, que tipos de variáveis ele usa, que tipos numéricos, se há a utilização de números primos nas instruções, o tamanho da chave, se há algum tipo de permutação nas mensagens a serem criptografadas, e assim por diante.

De acordo com [Rojas, Custódio e Berkenbrock 2004], a maioria dos algoritmos criptográficos são derivações das Redes Feistel que se baseia na criptografia de blocos simétricos de forma linear, mais conhecidos como *cifras de blocos*, combinando assim dois ou mais cifradores básicos em seqüências, onde cada bloco é composto por um número determinado de bits, os quais podem ser divididos em partes iguais de N bits cada. Para uma cifra mais forte desenvolve-se um bloco cifrado com uma chave de comprimento k bits e um bloco de comprimento n bits, permitindo um total de 2^k possíveis transformações, mais do que a 2^n transformações disponíveis. A cifra de bloco IDEA (*International Data Encryption Algorithm*) desenvolvida por James Massey e Xuejia Lai, em [Lai e Massey 1990], basear-se neste processo, no entanto, ela é relativamente lenta prejudicando seu desempenho. Observou-se a necessidade de utilizar algum tipo de permutação[2] ou mesmo a substituição[3] para aumentar ainda mais a segurança. Um dos ataques que servem como parâmetros de comparação em relação a outros tipos é o *Busca Exaustiva* ou *Força Bruta*, segundo a definição em [Nakaraha 2006].

Definição 2.1.1 *Este tipo de ataque é um ataque genérico que pode ser aplicado a uma cifra, onde um possível intruso tem o conhecimento dos algoritmos de ciframento(E_k) ou decifra-mento (D_k), como é de características das cifras baseadas em Feistel, um plaintext (P) e seu respectivo ciphertex (C) e calcula para todas a chaves K_i possíveis deste algoritmo (dependendo do tamanho da chave), seus possíveis valores para ($E_{k_i}(P) = C_i$) ou ($D_{k_i}(C) = P_i$), ou seja, se para algum K_i obter-se $C_i = C$ ou $P_i = C$ então K_i é uma chave provável.*

Para evitar este tipo de ataque, algumas características interessantes são desejáveis em uma cifra, como as seguintes abaixo apresentadas:

- Confusão - Esta técnica permite em tornar a relação estáticas entre os *plaintexts*, *ciphertext* e o valor da chave k a mais complexa possível, dificultando assim um possível ataque.

[2] Aplica-se uma permutação no texto antes e após do processo de cifragem, esta permutação é condicionada ao algoritmo, pode ser uma movimentação dos elementos no sentido horário, anti-horário etc.

[3] Substituir determinados elementos do texto por outros antes e após o processo de cifragem

- Difusão - Já a difusão consiste em eliminar a estatística do *plaintext*, visando assim deixar as relações entre o *plaintext* e o *ciphertext* ainda mais complexas.
- Efeito Avalanche - É a característica atribuível a cifra, quando qualquer modificação em um bit em um *plaintext*, tem como conseqüência a alteração aproximada da metade dos bits do seu respectivo *ciphertext*.

Contudo, mesmo que um algoritmo criptográfico detenha estas características, é possível ainda quebrar uma cifra e descobrir qual chave foi utilizada e como foi utilizada no processo de cifragem ou decifragem. A seguir serão descritos dois algoritmos de criptografia simétricos baseados nas Redes Feistel com DES em [Smid. e Branstad 1998], BlowFish em [Oliveira 2003], IDEA apresentado em [Lai e Massey 1990]. Foi possível quebrar suas respectivas chaves utilizando a técnica de Criptoanálise Diferencial, que será posteriormente definida no decorrer deste trabalho.

2.2 Cifras de DES

O Padrão de Encriptação de Dado (Data Encryption Standard - DES) foi desenvolvido em 1977 pela IBM, no período de guerra fria, baseou-se no algoritmo anterior chamado *Lucyfer* que era implementado utilizando as cifras de Feistel. DES foi desenvolvido por necessidade de uma maior segurança na comunicação, sendo seu algoritmo classificado como algoritmo simétrico[4]. A partir da idéia que o DES é uma cifra de blocos e também baseia-se nas redes Feistel, há a necessidade de conhecimento sobre estas redes e seu funcionamento. Feistel é um exemplo de cifra de blocos, com uma característica de alternar as permutações (transformações) e substituições feitas no processo de cifragem, fato esse, já colocado por Shannon[5] em 1949 para garantir características como difusão e a confusão.

2.2.1 Cifras Feistel

De acordo com [Nakaraha 2006] segue-se a seguinte definição:

Definição 2.2.1 *As cifras Feistel consistem em utilizar substituições e transposições, alternada e interativamente. com o intuito de uma combinação satisfatória de ambas para uma cifra mais forte que com a utilização de apenas uma.*

[4]Classe de algoritmos de criptografia, nos quais as chaves criptográficas são relacionadas para a decifragem e a cifragem

[5]Claude Elwood Shannon - engenheiro eletrônico e matemático americano, conhecido como ”o pai da teoria da informação”

O algoritmo das Redes Feistel foi originalmente utilizado em Lucifer[6] e em muitos algoritmos criptográficos como DES, FEAL, GOST, Khufu e Khafre, Loki, Cast, Blowfish, RC5 e outros. A estrutura de uma cifra Feistel é apresentada na Figura 2.1. As entradas deste algoritmo de criptografia são os *plaintext* com tamanho $2w$ bits e com uma chave K. Estes blocos são divididos em duas partes, L_0 e R_0, assim cada metade do texto claro passa através dos n ciclos (rodadas) pertencentes à rede Feistel, onde em cada ciclo, cada saída de uma função F é aplicada operação XOR com a outra metade do texto do ciclo anterior, o que conseqüentemente leva após o processo de todos os ciclos, uma combinação das metades transformando-a em um *ciphertext*, tendo assim a utilização do próprio texto para compor o processo de encriptação, dificultando ainda mais a quebra da cifra. Note também que cada entrada de um ciclo i tem entradas L_{i-1} e R_{i-1}, que são derivadas dos ciclos anteriores, como também as sub chaves utilizadas em cada ciclo.

Todos os ciclos têm a mesma estrutura, aplicando uma substituição através dos suas caixas S na função F, onde a única diferença entre as outras funções pertencentes aos outros ciclos é o valor da chave utilizada naquele ciclo. As chaves são relacionadas pelo gerenciador de sub chaves que, no caso da encriptação, a ordem de utilização das chaves é crescente e proporcional ao número do ciclo, enquanto na decriptação a ordem das chaves é alterada e passa a ser a ordem inversa, ou seja, decrescente em relação ao número de ciclos.

2.3 Data Encryption Standard

O *DES* (*Data Encryption Standard*) é um dos esquemas mais usado em sistemas que utilizam criptografia para segurança. Em 1973, a *National Bureau of Standards* (NBS) emitiu um pedido de propostas para uma cifra nacional padrão, pelo fato do aumento de transações bancárias e do risco das invasões comprometerem a segurança financeira. Buscava-se então um padrão, porque vários fabricantes começavam a criar suas próprias estruturas criptográficas o que dificultava as transações entre arquiteturas diferentes. Segundo [Oliveira 2003] este padrão tinha que atender os seguintes requisitos:

- Básico: Especificar as funções com serviços, métodos e resultados como requerimentos para um objetivo comum.
- Interoperabilidade : especifica as funções e formatos técnicos que permitam a transação de dados sem erros de transmissão, codificação ou qualquer coisa de natureza semelhante entre dois comutadores;

[6]Nome dado ao algoritmo baseado em cifras de blocos desenvolvido por Horst Feistel em parceria com a IBM

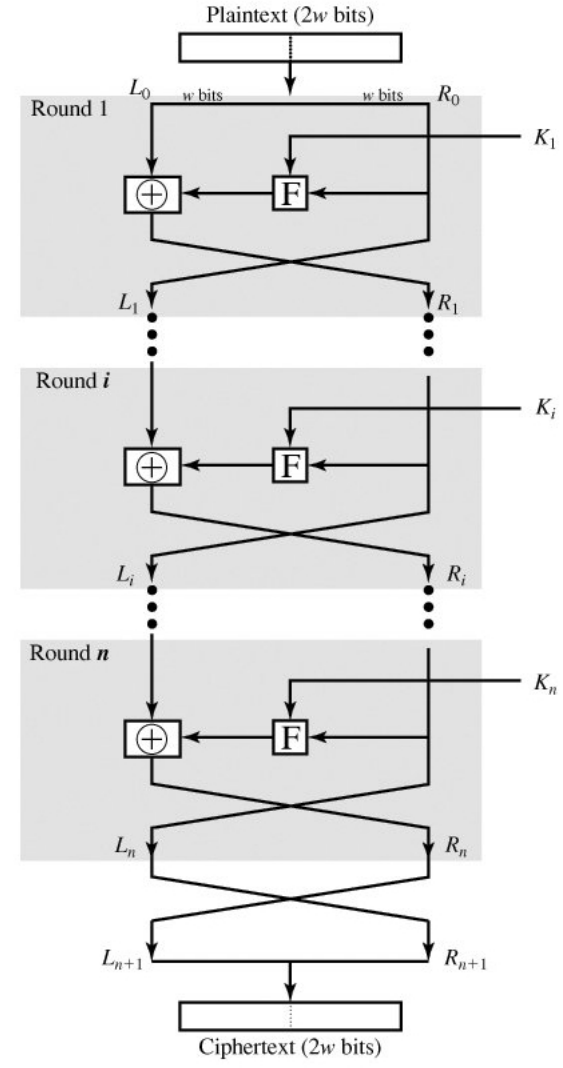

Figura 2.1: Representação clássica de uma rede Feistel
Fonte: [Stallings 2005]

- Interface: especifica além dos formatos dos dados, as especificações lógicas, físicas, elétricas suficientes para possíveis alterações de implementações em um determinado lado de uma interface;

- Implementação: além de especificar também as interfaces, funções e formatos, especifica também a estrutura da implementação.

O algoritmo escolhido, além destes requisitos, também deveria fornecer um alto nível de segurança, que fosse especificado através de notação formal e que fosse de fácil compreensão, onde a segurança residisse no desconhecimento da chave e não no desconhecimento do algoritmo, deveria ser eficiente, adaptável e sua implementação em dispositivos eletrônicos fosse economicamente viável. Então a IBM apresentou os resultados do seu projeto Tuchman-Meyer, assim a NBS aceitou-o e foi o algoritmo proposto e sendo então adaptado em 1977 como o DES, logo depois o DES foi alvo de inúmeras criptoanálises para que fossem possíveis encontrar falhas na sua estrutura, mesmo assim ainda continuou a ser utilizado pelo governo e posteriormente pelo setor privado. O algoritmo do DES baseia-se na rede Feistel, tendo, a priori, o mesmo comportamento no processo de cifragem. O DES tem características peculiares diferentes de Lucifer como, por exemplo, o tamanho da chave que passa de 128 bits para 56 bits e uma mudança nas caixas S que compõem a função F, pertencente ao toda cifra criptográfica.

A idéia principal do algoritmo DES é que os textos claros (blocos de cifras) tenham um tamanho de 64 bits e como sua chave de 56 bits, gere sub chaves de 48 bits através dos gerenciadores de chaves. O primeiro procedimento do algoritmo é aplicar sobre texto claro uma permutação inicial usando uma tabela de permutação (veja Tabela 1), e conseqüentemente uma divisão do texto claro em duas partes iguais L e R, que após a entrada na função de cada ciclo a parte direita seja expandida para 48 bits através da utilização da tabela 1.

Juntamente com a chave de 48 bits, provinda do gerenciador de chaves que utiliza uma chave mestra de 56 bits, é aplicada através de uma operação XOR, a qual será posteriormente divida em 8 partes que servirá como entrada nas 8 caixa S pertencentes à função F em cada ciclo, como mostra a figura 2.2. Verificamos nessa figura, que texto claro é aplicado em cada ciclo com uma sub chave k diferente de tamanho de 48 bits, totalizando 16 chaves respectivamente uma para cada ciclo, ou seja, tem-se 16 ciclos e no fim da encriptação ocorre uma última permutação que é inversa à primeira aplicada no início do processo, gerando assim um texto cifrado, *ciphertex*, com 64 bits.

Cada ciclo é descrito como mostra a figura 2.3, após a divisão do texto claro, os 32 bits correspondentes à parte direita do texto que serve como entrada na função F, ocorrendo primei-

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Tabela 2.1: Tabela de permutação inicial P
Fonte:[Stallings 2005].

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Tabela 2.2: Tabela de expansão de 32 bits para 48 bits em cada função F.
Fonte:[Stallings 2005].

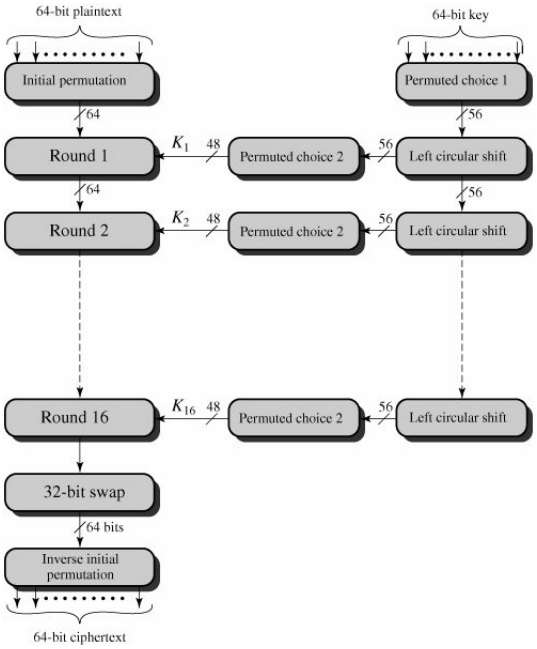

Figura 2.2: Visão geral sobre o algoritmo DES
Fonte: [Stallings 2005]

ramente uma expansão para gerar um texto com 48 bits que em seguida será aplicada através de uma operação XOR[7] com os 48 bits referentes à chave estabelecida para um determinado ciclo. O próximo passo é dividir o resultado desta operação em 8 caixas S, como mostra a figura 2.4 ou caixas de substituição, com cada caixa S recebendo 6 bits de entrada e tendo como saída 4 bits, totalizando assim 32 bits de saída de todas caixas S. Por último, é aplicada uma permutação P que será a saída da função F que em seguida será aplicada uma operação XOR com os outros 32 bits pertencentes à parte esquerda do texto claro, gerando assim após sua concatenação uma texto cifrado resultante apenas do primeiro ciclo, com o tamanho de 64 bits, ou seja, cada ciclo terá a seguinte seqüência:

$L_i = R_{i-1}$
$R_i = L_{i-1} \oplus F(R_{i-1}, K)$

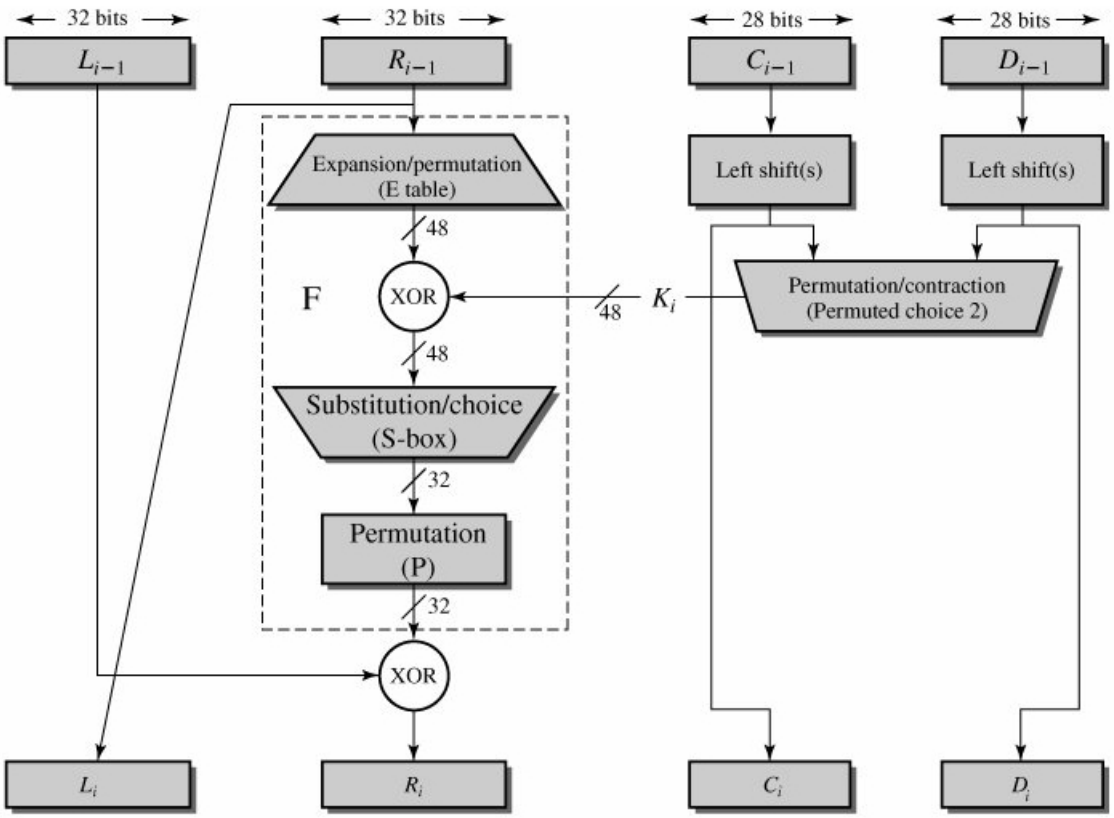

Figura 2.3: Um ciclo no algoritmo DES
Fonte: [Stallings 2005]

A estrutura de cada caixa S é composta de tabelas de substituições[8], onde os seis primeiros bits pertencentes ao resultado após a operação XOR com a chave, são as entradas da primeira

[7] Operação ou-exclusivo, denotada pelo símbolo $\oplus$.
[8] Para mais detalhes sobre o processo de substituição em [Stallings 2005]

caixa S, denominado S_1, que tem sua própria tabela de substituição, após utilizar essa tabela, gera como resultado 4 bits ao invés de 6 bits, isso implica nos conceitos de difusão e confusão, onde atribui-se a mudança em mais da metade do resultado da cifra por mudança em pelo menos um bit da entrada. A criptoanálise diferencial, aplicada ao DES, como será visto no capítulo 3, é primeiramente baseada na descoberta das sub chaves de cada caixa S, por isso a importância de entendê-los. A estrutura do DES é apresentada na figura 2.5 para uma compreensão do comportamento dos textos claros ao longo do processo de cifragem.

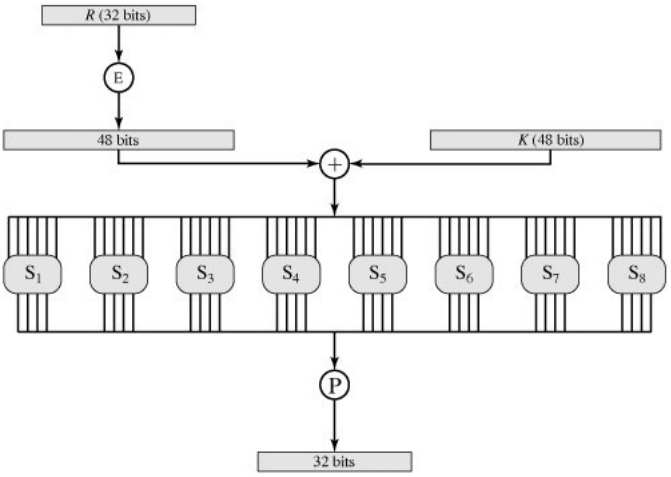

Figura 2.4: A função F com caixas S
Fonte: [Stallings 2005]

O processo de decriptação do DES ocorre de maneira inversa à apresentada na encriptação. No texto cifrado é aplicada a permutação inversa e o primeiro ciclo de entrada não será o ciclo 1, mas sim o ciclo de ordem 16 com sua respectiva sub chave e assim por diante, até o ciclo 1, onde será aplicada novamente a permutação originando assim o texto claro decodificado.

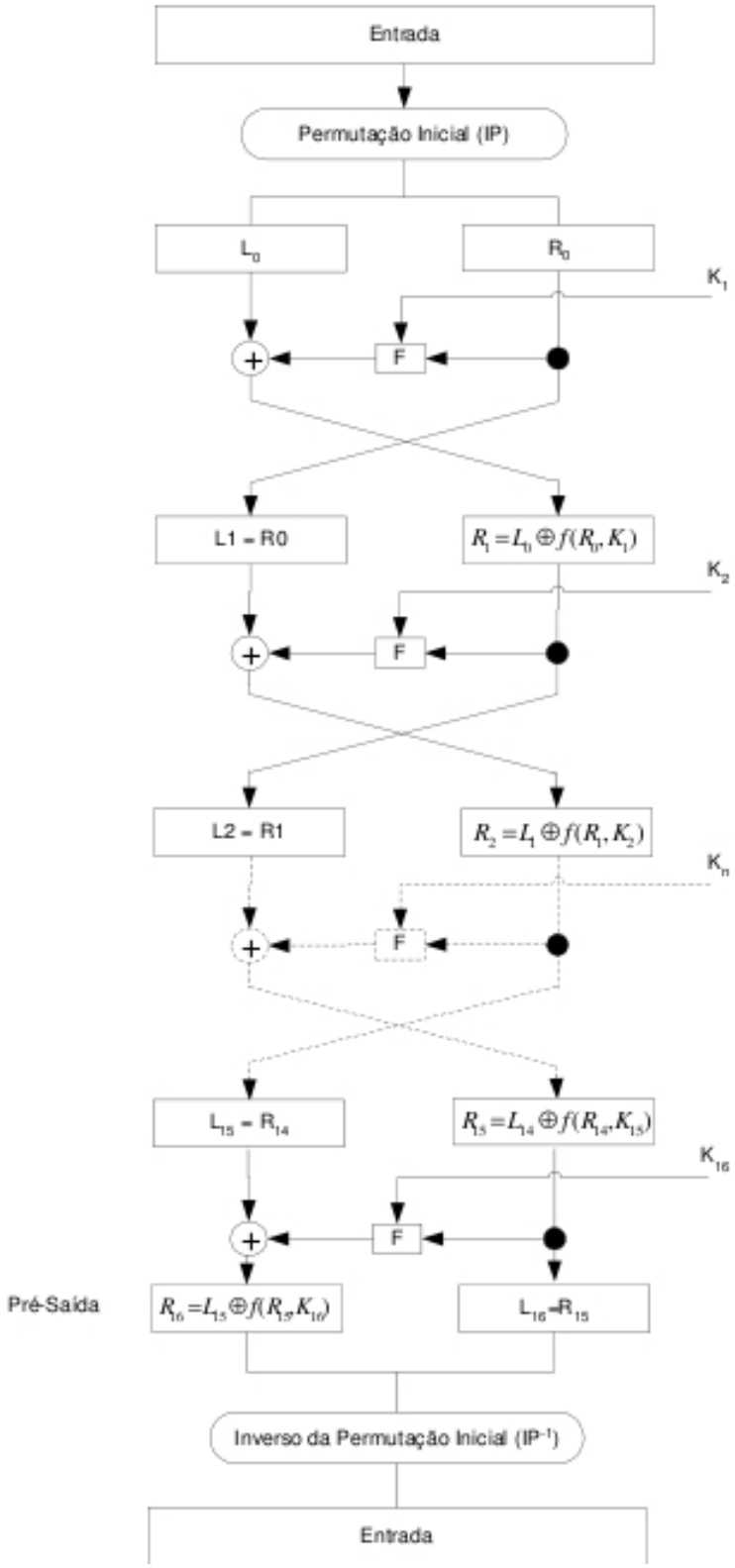

Figura 2.5: Estrutura geral do algoritmo DES
Fonte: [Stallings 2005]

2.4 FEAL-NX

As cifras FEAL-N e FEAL-NX (*Fast Data Encipherment Algorithm*), desenvolvidas por [Nakaraha 2006] são cifras simétricas baseadas nas redes Feistel, ou seja, são cifras de blocos, que foram projetadas para serem implementadas em hardware e software, com tamanhos mínimo de 8 bits, onde seus blocos e chaves têm 64 bits de tamanho. Nakahara [Nakaraha 2006] usa notações semelhante para FEAL-N e FEAL-NX conhecida como FEAL-N(X).

2.4.1 Descrição das Caixas S

Assim com nas outras cifras baseadas nas redes Feistel, a cifra FEAL-NX utiliza também caixas de substituições (caixas S).

Definição 2.4.1 ([Nakaraha 2006]) *As caixas* $S_i : \{0,1\}^8 \times \{0,1\}^8 \rightarrow \{0,1\}^8$*, para* $i \in \{0,1\}$*, ou caixas de substituição, são funções não-lineares definidas por: se* $\alpha, \beta \in \{0,1\}^8$*, então*

$$S_i(\alpha,\beta) = ROL2((\alpha + \beta + i)(mod\ 256))$$

$i \in \{0,1\}$*, onde* $ROL2 : \{0,1\}^8 \rightarrow \{0,1\}^8$ *representa uma rotação de dois bits para a esquerda e* (Y) *mod Z representa o resto da divisão de Y por Z.*

Definição 2.4.2 ([Nakaraha 2006] apud Fu87) *Uma função criptográfica F é completa se cada um dos seus bits de saída depende de cada bit de entrada.*

Nas cifras FEAL-N(X) e DES a função F é composta por caixas S, ou seja, caixas de substituições, o que demonstra que a função F em ambos algoritmos, pela definição 1.4.2, não é completa. Isso ocorre pelo fato dos bits de entrada em cada caixa S serem menos significativos, ou seja, são bits da etapa imediatamente anterior, logo; as operações de soma só dependem dos bits menos significativos em relação à sua ordem, ou seja, desconsiderando os bits mais significativos que são aqueles bits que foram considerados como entradas na função F.

2.4.2 Geração de Sub Chaves

Nas cifras FEAL-N(X) existe uma função F_k, que é responsável pela geração das sub chaves para a cifra. Esta função é definida da seguinte forma:

Definição 2.4.3 ([Nakaraha 2006]) *A função* $F_k :: \{0,1\}^{32} \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ *recebe : um valor* $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$*, e um valor* $\beta = (\beta_0, \beta_1, \beta_2, \beta_3)$*, que são os semi blocos esquerdo e direito, nessa ordem, de uma chave k, ou, valores intermediários obtidos durante a geração das sub chaves. A função F é composta de duas caixas* S_i *para a seguinte forma:*

$$F(\alpha, \beta) = \psi = (\psi_0, \psi_1, \psi_2, \psi_3) \text{ tal que,}$$

$\psi_1 = S_1(\alpha_0 \oplus \alpha_1, \alpha_2 \oplus \alpha_3 \oplus \beta_0)$
$\psi_0 = S_0(\alpha_0, \psi_1 \oplus \beta_2)$
$\psi_2 = S_0(\psi_1 \oplus \beta_1, \alpha_2 \oplus \alpha_3)$
$\psi_3 = S_0(\psi_2 \oplus \beta_3, \alpha_3)$

A função F_k é apresentada com mais detalhes em [Nakaraha 2006], onde é apresentada um pseudocódigo para a geração de sub chaves de tamanho de 64 bits para FEAL-N e 128 bits para FEAL-NX, para uma certa quantidade de interação N.

2.4.3 Função F FEAL-N(X)

Definição 2.4.4 *A função* $F : \{0,1\}^{32} \times \{0,1\}^{16} \rightarrow \{0,1\}^{32}$ *para as cifras FEAL-N(X) recebe com parâmetros um valor* $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ *de 32 bits, logo cada* α_i *tem 8 bits de tamanho, que correspondem ao semi bloco direito, ou parte direita de cada interação e uma sub chaves* $\beta = (\beta_0, \beta_1)$ *de 16 bits.* $F(\alpha, \beta) = \gamma = (\gamma_0, \gamma_1, \gamma_2, \gamma_3)$*, para* $\alpha \in \{0,1\}^{32}$ *e* $\beta \in \{0,1\}^{16}$*, tem-se:*
$\gamma_1 = S_1(\alpha_0 \oplus \alpha_1 \oplus \beta_0, \alpha_2 \oplus \alpha_3 \oplus \beta_1)$
$\gamma_0 = S_0(\alpha_0, \gamma_1)$
$\gamma_2 = S_0(\gamma_1, \alpha \oplus \alpha_3 \oplus \beta_1)$
$\gamma_3 = S_1(\gamma_2, \alpha_3)$

Logo pela definição 2.4.2 se as caixas S não são funções completas, e a função F é composta pelas mesmas, podemos concluir que F também não é completa. As prioridades para que uma função F seja segura, são as mesmas adotadas às demais funções de outras cifras de blocos, são o efeito avalanche que já foi descrito neste capítulo, a difusão e a confusão. Para mais informações sobre o efeito avalanche do FEAL-N(X), ver [Nakaraha 2006].

Segundo [Nakaraha 2006][apud MaYa92], para garantir tais características à função F, foi necessária modificá-la, passando a ser denominada função mF, a sua alteração é relativa às

transformações iniciais e finais dos blocos de textos com suas respectivas sub chave. Ver figura 2.6.

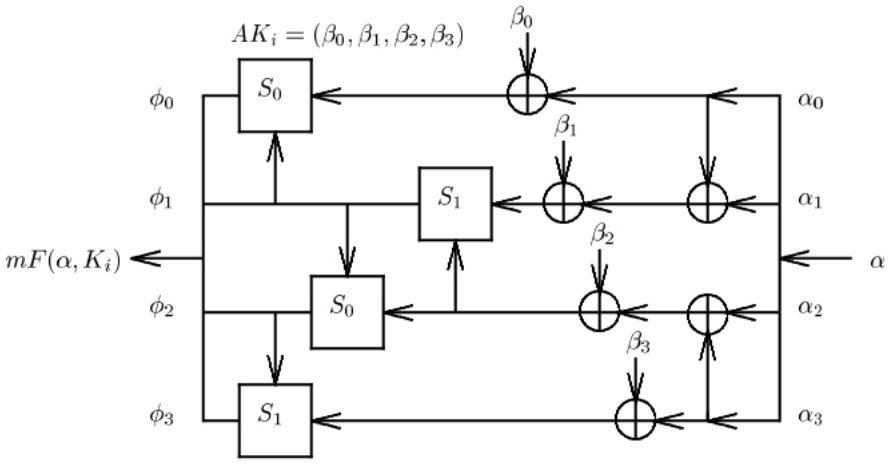

Figura 2.6: Função *mF* para FEAL-N(X)
Fonte:[Nakaraha 2006]

Definição 2.4.5 *A função* $mF : \{0,1\}^{32} \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ *tem os seguintes parâmetros com entrada: um valor de 32 bits* $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$*, correspondente ao semi bloco direito, ou parte direita de cada interação e uma sub chave* $\beta = (\beta_0, \beta_1, \beta_2, \beta_3)$ *de 32 bits. Tem-se para mF: Para* $\alpha, \beta / in \{0,1\}^{32}$ *tem-se* $mF(\alpha, \beta) = \phi = (\phi_0, \phi_1, \phi_2, \phi_3)$*, onde:*
$\phi_1 = S_1(\alpha_0 \oplus \alpha_1 \oplus \beta_0, \alpha_2 \oplus \alpha_3 \oplus \beta_1)$,
$\phi_0 = S_0(\alpha_0, \gamma_1)$,
$\phi_2 = S_0(\gamma_1, \alpha \oplus \alpha_3 \oplus \beta_1)$ *e*
$\phi_3 = S_1(\gamma_2, \alpha_3)$

A função *mF* tem como propósito alterar a ordem e a posição em que são realizadas as operações de transformação inicial com uma sub chave especificada e a transformação final do bloco de texto cifrado com a sub chave especificada. Para que isso ocorra, as sub chaves são deslocadas aos pares pela estrutura da rede Feistel e combinadas com as sub chaves da função F em cada interação, mais detalhes em [Nakaraha 2006]. A estrutura final, após a função modificada, é apresentada na figura 2.7.

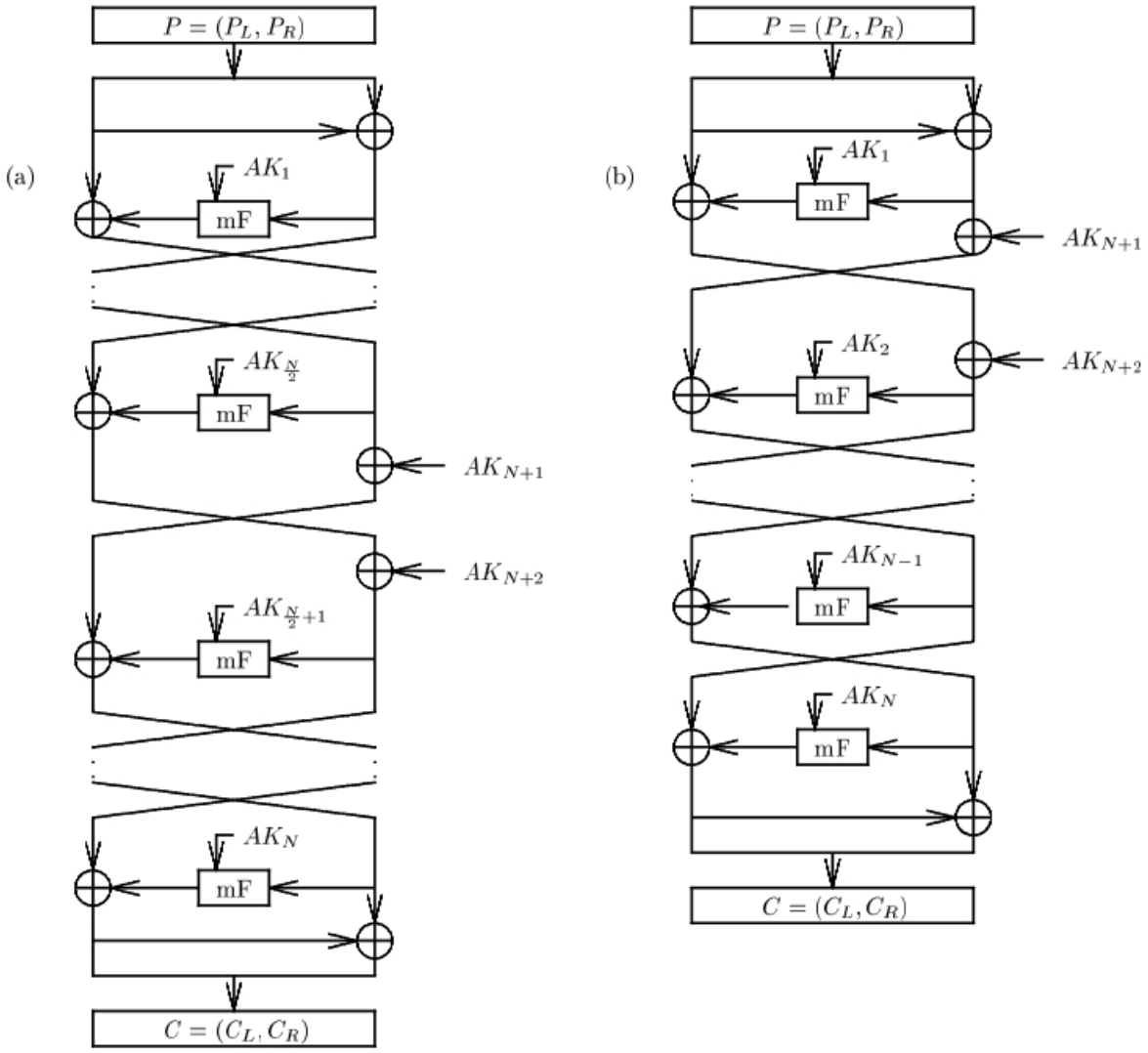

Figura 2.7: Estrutura Feistel FEAL-N(X) com a função *mF*
Fonte: [Nakaraha 2006]

2.5 BlowFish

Outra cifra baseada nas redes Feistel, o algoritmo de criptografia BlowFish também possui 16 ciclos e também trabalha com blocos de cifras com 64 bits de tamanho, porém sua chaves podem ter tamanho variável até 448 bits. Segundo [Oliveira 2003][apud BS94] o BlowFish foi projetado por Bruce Schneier em 1993, é aceitável em aplicações onde a chave não mude frequentemente, no entanto é mais rápido que o DES quando ambos são implementados em computadores de 32 bits. Ele consiste em duas partes, sendo um processo de expansão da chave e outro para a cifragem de dados, onde aquele converte a chave de 448 bits em sub chaves de tamanho menor. O processo de cifragem consisti, assim como em FEAL-N(X), em 16 ciclos, assim como mostra a figura 2.9, o texto de 64 bits é divido em duas partes de 32 bits (L_0, R_0), onde cada rodada é da forma:

$$R_i = K_i \oplus L_{i-1}$$
$$L_i = R_{i-1} \oplus F(R_i)$$

para $1 \leq i \leq 16$. A peculiaridade do BlowFish em relação ao DES e ao FEAL, é que a operação XOR da sub chave com a parte **esquerda** do texto, ocorre antes da entrada na função F. Outra diferença presente no BlowFish é que são geradas 18 sub chaves, porém como existem apenas 16 ciclos, duas das 18 sub chaves são usadas ao final e fazem uma combinação e geram o texto cifrado. O processo de decifragem ocorre da mesma forma, pelo BlowFish ser simétrico, só que de maneira inversa na utilização das sub chaves. A função **F** (ver figura 2.8) é definida segundo [Oliveira 2003] da seguinte forma:

$$F(L) = ((S_1(a) + S_2(b) mod 2^{32}) \oplus S_3(c)) + S_4(d) mod 2^{32}$$

onde a, b, c, d são as quatro partes iguais de L_i com 8 bits cada. As caixas S recebem como entrada cada um desses 8 bits e têm como saída uma cifra de 32 bits.

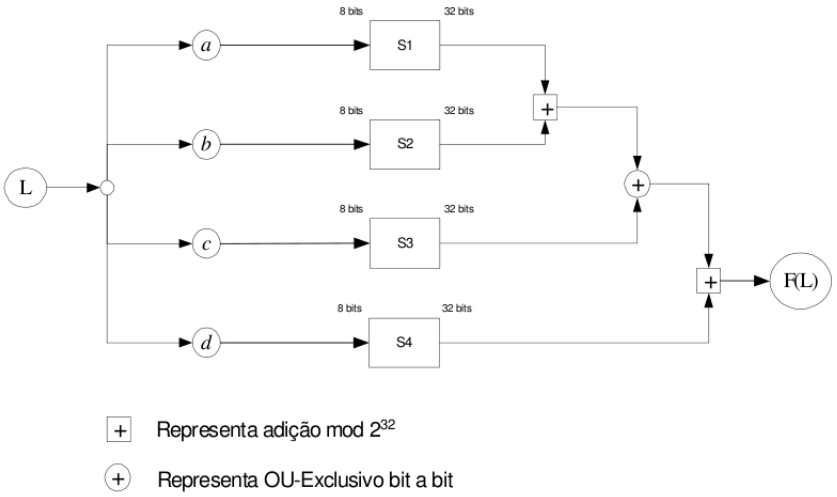

Figura 2.8: Função F do BlowFish
Fonte: [Oliveira 2003]

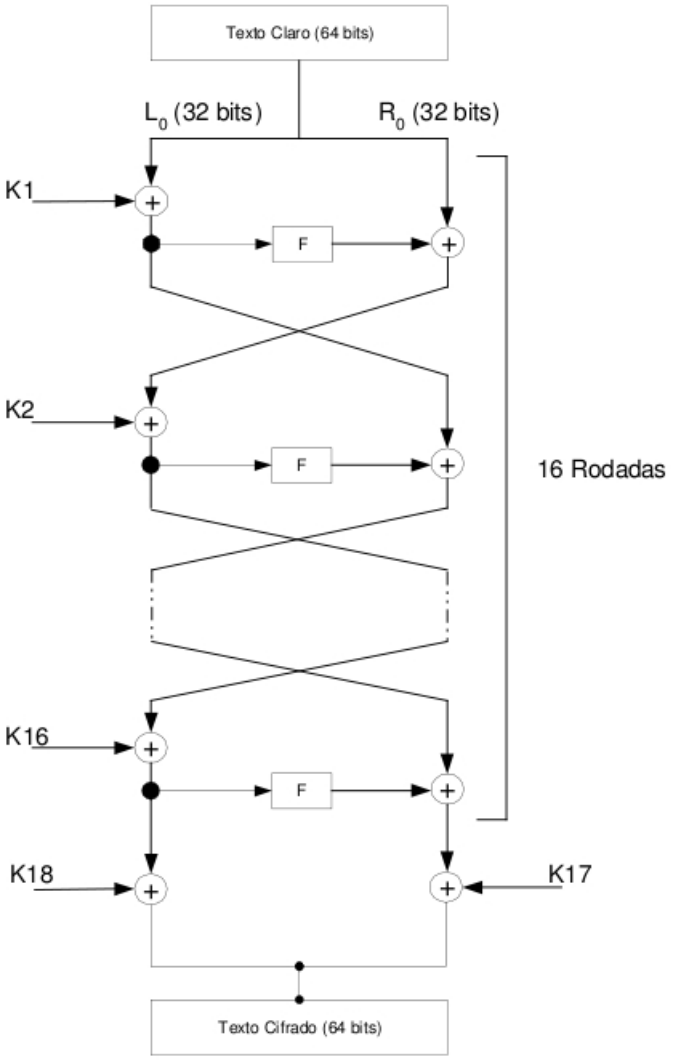

Figura 2.9: Estrutura Feistel BlowFish
Fonte: [Oliveira 2003]

3 *Algoritmo Papílio*

> The great things are often easier than what you think
>
> François-Marie Arouet

O algoritmo de criptografia Papílio foi desenvolvido por [Ramos 2002]) em 2002.

> Inspirado 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 e em todo o estado do Rio Grande do Norte.[Ramos 2002]

Este algoritmo tem sua funcionalidade igual a todos os tipos de cifras que têm como base a rede Feistel, ou seja, usa cifras de blocos, porém a peculiaridade do Papílio é dada através da utilização do codificador convolucional Viterbi na função F utilizada em cada ciclo. O texto a ser cifrado é divido em duas partes iguais, após a operação XOR com a chave entra no codificador Viterbi modificado, passando então posteriormente aos boxes seguintes os valores resultantes do processo interno do mesmo. A seguir será apresentado o algoritmo Viterbi Modificado (Função F) em conjunto como o Papílio e seus funcionamentos de forma a permitir a compreensão da criptoanálise aplicada sobre o mesmo.

3.1 Viterbi Modificado

O mecanismo diferencial do Papílio para sua encriptação, como já foi mencionado, é a utilização do algoritmo Viterbi Modificado. Sua utilização está presente tanto na geração de sub chaves, como também dentro da função F, aplicada em cada ciclo. De posse deste algoritmo, seu funcionamento é baseado em máquinas de estado e tabelas de substituição como será explicado a seguir.

Segundo [Ramos 2002] o algoritmo Viterbi, foi desenvolvido por Andrew J. Viterbi para

soluções convolucionais[1], ou seja, para correções antecipadas de erros (FEC). Como foco deste trabalho, cita-se somente a codificação convolucional, ao invés da codificação por blocos.

3.1.1 Codificação Convolucional

O processo de codificação/decodificação utilizado no Papílio é acompanhado do processo de decodificação utilizando as técnicas de codificação convolucional e decodificação Viterbi, elas compõem a técnica de correção antecipada do erro (FEC). A codificação convolucional produz uma seqüência de bits codificados na saída em função de bits não codificados na entrada, ou seja, torna complexa tanto sua codificação como a decodificação. A seguir serão descritos os principais parâmetros de um codificador convolucional segundo [Ramos 2002]:

- R: É a restrição de comprimento,
- s/n : É a taxa de codificação onde n é a quantidade de bits que entra e s a quantidade de bits que sai,
- Q : Indica quantos bits da saída após a codificação, um bit da entrada pode influenciar,
- m : indica por quantos ciclos a informação será armazenada, ou seja, pode ser entendido como o tamanho da memória do codificador.

A codificação da informação no algoritmo Papílio é feita por dois tipos de componentes básicos que são o registrador e o módulo somador. Para um melhor entendimento deste assunto, aborda-se um exemplo que é encontrado com mais detalhes em [Ramos 2002]. O exemplo aborda um codificador convolucional simples o qual está exibido na figura 3.1. O codificador deste exemplo tem as seguintes características: Taxa de codificação (n/s) = 1/2, Q=3, m=2.

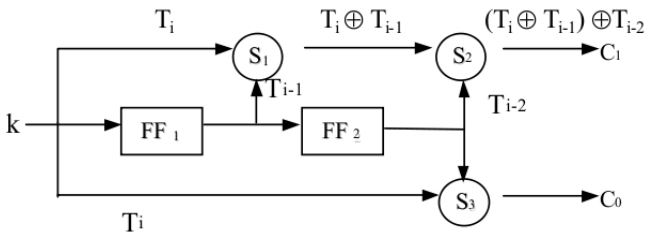

Figura 3.1: Esquema do codificador convolucional
Fonte:[Ramos 2002].

[1]Uma das técnicas de correção de erros aplicada a transmissão de mensagens, tem por característica que os bits a dependem não só dos bits de informação como também da informação armazenada pela memória do código.

Observa-se que para cada bit de entrada n tem-se como saída dois bits s, ou seja, a taxa de codificação é $s = 2n$ símbolos. Outra informação que pode ser extraída da figura 3.1, é o fato do bit de entrada ser estável durante o ciclo, ou seja, no ciclo T_i o *flip-flop* FF2 guarda a entrada do ciclo T_{i-2} e o *flip-flop* FF1 guarda a entrada do ciclo T_{i-1} sendo assim a entrada do codificador receberá a entrada atual. Portanto temos como saída para $C_1 = (T_i \oplus T_{i-1}) \oplus T_{i-2}$ e para saída $C_0 = T_i \oplus T_{i-2}$.

Contudo, este codificador pode ser visto como uma máquina de estados, ou seja, pode tratar fluxos rotuláveis tendo neste exemplo quatro possíveis estados 00_2, 01_2, 10_2 e 11_2, mais informações em [Ramos 2002]. A figura representa uma máquina de estado capaz de atender as necessidades do esquema do codificador convolucional.

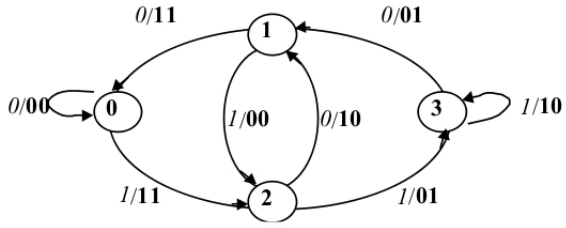

Figura 3.2: Máquina de estado do esquema do codificador convolucional
Fonte:[Ramos 2002].

Como o objetivo deste trabalho é a criptoanálise aplicada sobre o processo de encriptação do Papílio, aborda-se, a nível de conhecimento o decodificador Viterbi a seguir.

3.1.2 Decodificador Viterbi

O decodificador Viterbi trabalha com os mesmos parâmetros do codificador convolucional. Com o objetivo de reconstruir os bits de entrada do codificador convolucional, a decodificação Viterbi retira a redundância adicionada ao fluxo de bits, através de um processo probabilístico. Segundo [Ramos 2002] apud [Ryan e Nudd 1993] 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. Para uma estimativa de qual é a possível 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 com os mesmos valores para os parâmetros, como $Q = 3$, taxa $s/n = 1/2$ e $m = 2$ para uma mensagem de 15 bits.

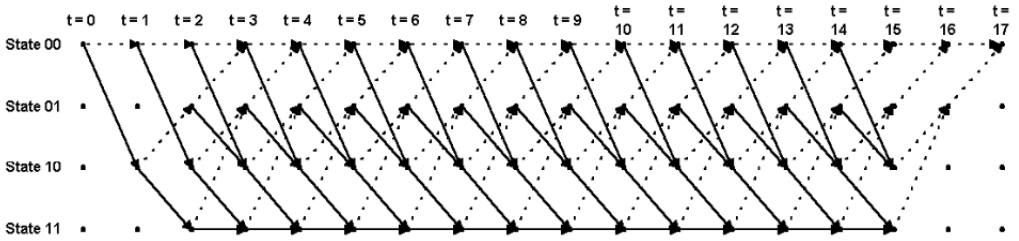

Figura 3.3: Treliça referente a máquina de estado do esquema do codificador convolucional
Fonte:[Ramos 2002].

O decodificador Viterbi analisa os rótulos e de acordo com o estado atual, rotula todos os próximos possíveis estados, ou seja, todas as transições possíveis e avalia-os através de distâncias que são calculadas de maneira linear, que é a diferença entres os bits de entrada e os bits de cada possível rótulo. 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*), esta distância pelo fato de ser linear pode ser calculada através da distância de Hamming[2]. Porém para encontrar de fato a possível seqüência de bits da entrada que geraram o fluxo de bits de saída, o decodificador Viterbi utiliza três unidades funcionais, são elas:

Branch Metric (BM)

Segundo [Ramos 2002] a unidade funcional *branch metric* calcula, a cada instante, a distância entre o valor do código recebido e os valores dos rótulos das transições. Estes resultados são armazenados e associados aos seus respectivos estados. Como base o exemplo adotado em [Ramos 2002], a figura 3.4 tem duas transições possíveis (estados futuros 00 e 10),a partir do estado inicial 00 e tempo $t = 0$, rotuladas 00_2 e 11_2, respectivamente. Caso ocorra que o símbolo recebido na entrada do decodificador seja igual a 00_2 tem-se os resultados 0 e 2 obtidos através de Hamming, logo 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.

Como a técnica utilizada é FEC, a distância de cada rótulo representa a quantidade de erros existentes em um determinado estado, num dado instante de tempo, que são denominados medida de erros acumulados. Como mostra a figura 3.4, esta medida de erro possui valores de 0 a 2.

[2]Cálculo aplicado de forma a contar a quantidade de bits diferentes entre a entrada e saída das transições

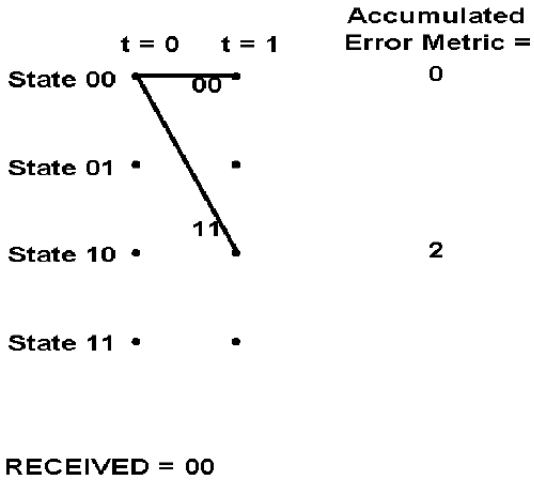

Figura 3.4: Transições com erro acumulados
Fonte:[Ramos 2002].

Add-Compare-Select (ACS)

Esta unidade é considerada em [Ramos 2002][apud Giulietti 98] como a mais importante de toda a decodificação. Esta unidade desempenha três funções: adição,comparação e seleção.

- Adição: somar o valor da distância (*branch metric)* atual ao valor acumulado anteriormente em cada estado. Ao ver a figura 3.3 nota-se que a treliça está no estado 00_2, onde tem duas transições possíveis, ou seja, 00_2 ou 10_2. O ACS adiciona o valor da métrica aos valores das distâncias calculadas através de Hamming.

- Comparação: Após a adição é feita a comparação que verifica qual estado contém a menor medida de erro acumulado.

- Seleção: Há a seleção dos estados que apresentarem menor medida de erro acumulado que serão armazenados em uma tabela de estados.

A medida que a treliça avança no tempo sua complexidade aumenta. E mais estados vão sendo armazenados e suas medidas de erro acumulado serão atualizadas a fim de se gerar uma tabela com dados e estados com menor quantidade de erros acumulados que serão usados na terceira unidade do decodificador Viterbi.

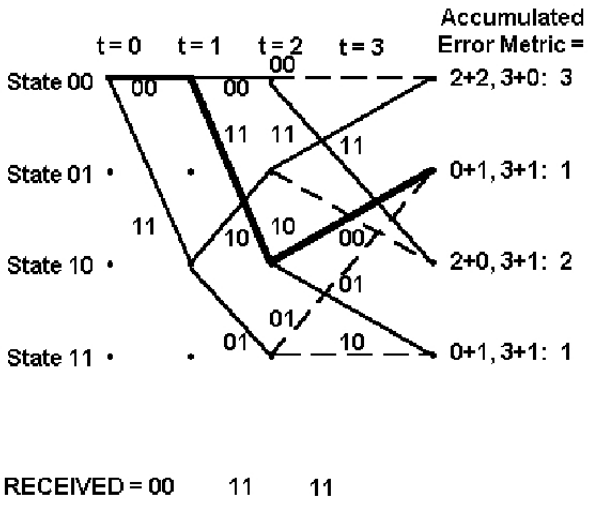

Figura 3.5: Transições com erro acumulados
Fonte:[Ramos 2002].

Traceback

O processo de decodificação Viterbi possui 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* segundo [Ramos 2002][apud Feygin 96a]. A técnica *register exchange* é a mais simples, porém a desvantagem está na questão de acada bit decodificado precisa ser lido e reescrito na memória, já *traceback* utiliza ponteiros para interpretação dos símbolos e assim não precisa mover os dados na memória. A unidade traceback recria a seqüência de bits que estavam na entrada do codificador convolucional, através de três passos:

1. Selecionar em cada instante de tempo, o estado que tiver a menor medida de erro acumulado e salvar o número deste estado. A tabela 1 mostra o erro acumulado para cada estado, como também para o fluxo de dados inicial 0 1 0 1 1 1, dado como exemplo na seção do codificador convolucional apresentado em [Ramos 2002]. A tabela 1 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		0	3	1	2	1	1	3	4

Tabela 3.1: Medidas de erro acumulado
Fonte:[Ramos 2002].

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

Tabela 3.2: Medidas de erro acumulado
Fonte:[Ramos 2002].

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 1 em 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	$00_2 = 0$	$01_2 = 1$	$10_2 = 2$	$11_2 = 3$
$00_2 = 0$	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 3.3: Medidas de erro acumulado
Fonte:[Ramos 2002].

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

Tabela 3.4: Medidas de erro acumulado
Fonte:[Ramos 2002].

3.1.3 Algoritmo Viterbi Modificado

Segundo [Ramos 2002] a função do algoritmo Viterbi Modificado no Papílio é ampliar o espaço de decodificação, a fim de possibilitar a geração de tabelas que serão utilizadas na criptografia de dados. Esta ampliação consiste do estado atual tratar qualquer fluxo de bits de entrada. O algoritmo Viterbi Modificado adota a Codificação Convolucional e a Decodificação Viterbi utilizando os seguintes componentes básicos: registradores de deslocamento e módulos somadores onde determinam-se os polinômios geradores do par de bits da saída.

Tomando-se como base o exemplo encontrado em [Ramos 2002], 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 componentes básicos e os polinômios definidos, a tabela da codificação convolucional é gerada, onde m é

a quantidade de estados. Um exemplo para uma melhor compreensão está em [Ramos 2002], porém, em síntese, esse exemplo faz com que em sua execução no algoritmo Viterbi gere um erro toda vez que o rótulo de entrada não pode ser tratado pelo estado atual, ou seja, a execução do algoritmo Viterbi Modificado inicia-se neste exato momento, onde no estado 0 executa-se o algoritmo Viterbi, descrito na subseção 3.1.2, até que um rótulo de entrada não seja tratável no estado atual e assim os rótulos que não foram tratados através do método padrão, serão tratados pelo Viterbi modificado. Portanto a partir da tabela gerada pela codificação convolucional em [Ramos 2002], identifica-se os rótulos que poderão ser tratados por cada estado, acrescentando a eles um novo fluxo de saída S1 a informação 0. Com o mesmo exemplo, no caso da tabela-4.2 em [Ramos 2002], apresenta 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. Ou seja, os valores contidos nas caixas S, representam quando um rótulo de bits de entrada, no Viterbi Modificado, foi tratado ou não de forma correta pelo Viterbi Modificado, indicando se houve ou não a necessidade de adicionar um novo estado inicial para tratar os dois bits de entrada..

3.2 Algoritmo Papílio

Visando dificultar possíveis ataques e análises, este algoritmo foi desenvolvido com algumas mudanças, que serão descritas a seguir:

- Tamanho da Chave e sub chaves - Atualmente 128 bits tornou-se o tamanho comum nos algoritmos de criptografia, sendo portanto 128 bits o tamanho da chave utilizada neste algoritmo, entretanto com esta chave, através do gerador de sub chave (ver figura 3.6) é capaz de gerar 16 sub chave diferentes através do algoritmo Viterbi Modificado, onde as quatro primeiras Sub chaves, rotuladas SC_1, SC_2, SC_3 e SC_4 são geradas da chave inicial de 128-bits, a qual gera dois fluxos de dados de 64 bits, com isso os dois fluxos de 64 bits geram através do algoritmo Viterbi mais dois fluxos cada, com isso tem-se 4 sub chaves com 32 bits de tamanho cada, dessa maneira concatena-se as 4 cavu e através do mesmo processo obtém-se mais 4 chaves até um total de 16 chaves.

- Número de Voltas - Este algoritmo utiliza 16 voltas (ciclos) visando dificultar uma possível criptoanálise como mostra a Figura 3.7. O texto a ser criptografado (texto claro) tem o tamanho de 64 bits, na inicialização do processo de encriptação o texto é divido em partes iguais de 32 bits cada, que são denominados parte direita e esquerda como respectivamente DE_i e EC_i para $i \mid 0 \leqslant i \leqslant 16$. No entanto, apenas o semi bloco direito DE_i é

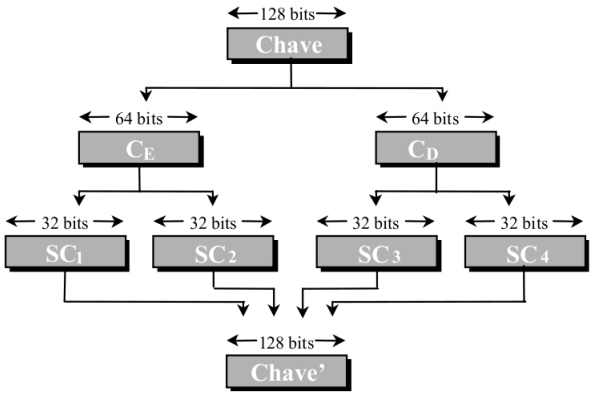

Figura 3.6: Estruturas do codificador Papílio
Fonte:[Ramos 2002].

entrada da função F durante todos os ciclos. Já EC_i sempre é aplicada à uma XOR com a saída da função F, como é mostrado na Figura 3.7.

- Função Utilizada - Utiliza o codificador Viterbi apesar de derivar de uma estrutura básica de um codificador Feistel, como mostra a Figura 3.7, onde no primeiro ciclo, é aplicada a chave SC_1 e a entrada de 32 bits na função F é denominada D_1, tendo como resultado da função um texto de 32 bits que será então aplicado através de operação XOR com a outra metade do *texto claro* denominada E_1, sendo então concatenadas D_1 e E_1 ao final do ciclo, elas serão novamente dividas em textos de tamanho 32 bits e passarão novamente por todo o processo até o ciclo 16, que por fim será feita a última permutação das partes, tendo como resultado sua concatenação, mais detalhes sobre a estrutura e funcionamento deste codificador e da função F correspondente, ver [Stallings 2005]. A função F utilizada neste algoritmo é diferente da adotada por [Stallings 2005] como é apresentada em figura 3.8[3], mas preserva a característica de permitir que o texto codificado seja dependente não somente da chave como também do *texto claro*[4], dificultando ainda mais descoberta da chave.

Como todo algoritmo criptográfico, o Papílio também detém características como a Difusão e a Confusão, sendo as mesmas conseguidas através de permutações e substituições feitas de maneira a dotar uma perfeita combinação das duas operações, mais detalhes sobre essas características podem ser vistas em [Ramos 2002].

[3]Figura que representa uma reavaliação do funcionamento da função F
[4]Texto legível anterior ao processo de cifragem

$$E_{i+1} = D_i$$

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

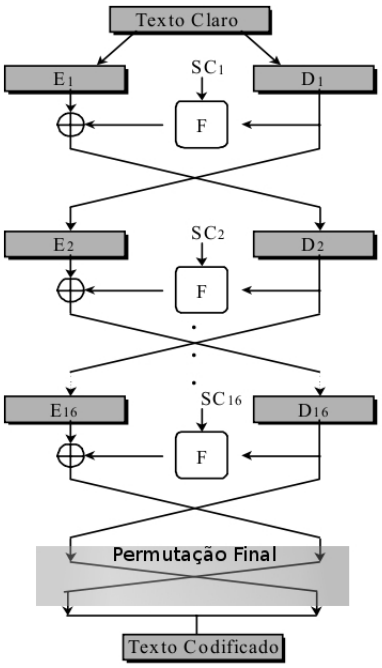

Figura 3.7: Estruturas do codificador Papílio
Fonte:[Ramos 2002].

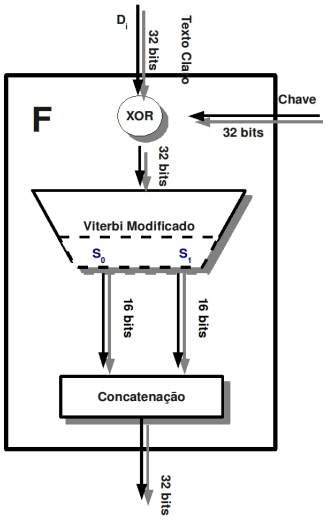

Figura 3.8: Estrutura da função F.

4 *Criptoanálise Diferencial*

> The failure breaks the small souls and it exalts the greats, such like the wind turn off the sail and induce the flames in the forest.
>
> Benjamin Franklin

Uma das técnicas utilizadas para analisar uma cifra durante seu desenvolvimento, de forma a torná-los algoritmos criptográficos eficientes e que utilizam chaves difíceis de serem quebradas, é a criptoanálise diferencial, pois a mesma é considerada uma das técnicas mais poderosas de ataque. Esta técnica, que utiliza-se das diferenças pares de *textos claros* e suas respectivas *cifras* juntamente com o conhecimento do algoritmo e das funções como também o gerenciador de chaves, consegue gerar relações de probabilidades entre os pares de *textos claros* e *textos cifrados*. Com estas relações é possível retirar conclusões sobre as possíveis chaves ou sub chaves de cada ciclo e, posteriormente, à chave utilizada para geração das demais chaves utilizadas pelo algoritmo. A seguir serão apresentadas as principais definições a respeito desta técnica, que inicialmente foram utilizadas por Biham e Shamir em 1990 [Biham e Shamir 1990] e que ao longo deste trabalho será objeto de comparação e eventuais análises, isto, pelo fato de ambos os algoritmos, Papílio e DES, utilizarem a rede Feistel na suas respectivas estruturas. Outros algoritmos a serem detalhados neste trabalho serão os FEAL-N e FEAL-NX, nos quais foi aplicada a mesma técnica de criptoanálise em [Nakaraha 2006] e posteriormente será objeto de análise comparativa em relação ao Papílio.

4.1 Definições Principais

Como visto no capítulo anterior existem inúmeros algoritmos com suas principais características, porém a característica inerente a todas as cifras é a chave, e a criptoanálise diferencial aplicada foi a grande preocupação Biham e Shamir em 1990 [Biham e Shamir 1990], antes

mesmo de lançarem o DES no mercado, aplicaram essa técnica, que consiste em analisar as diferenças entre pares de textos claros e os pares de suas respectivas cifras, analisando tão somente as possíveis chaves aplicadas naqueles textos claros que resultaram nas respectivas cifras.

Extraindo desta análise probabilidades, características e padrões que levarão a escolher as sub chaves mais prováveis de uma cifra e assim descobrir a chave utilizada pela cifra. Este capítulo descreverá esta técnica, apresentando resultados sobre o DES e FEAL-NX, de acordo com [Biham e Shamir 1990] e [Nakaraha 2006] respectivamente.

Toma-se um texto claro $X \in \{0,1\}^n$ formados por bits 0's ou 1's, onde os bits de X serão numerados da direita para a esquerda, sendo o primeiro bit menos significativo, isso é necessário para facilitar quando forem feitas as comparações entre os algoritmos.

Segundo [Biham e Shamir 1990] as principais notações usadas na técnica criptoanálise diferencial são:

- X^*, X: Quaisquer valores intermediários respectivos a cada etapa do processo de encriptação, $X^{'}$ é definido como $X^{'} = X \oplus X^*$
- P: Denota o texto claro depois da permutação inicial, que neste caso, foi dispensada das definições iniciais. P^* denota qualquer outro texto claro $P^{'} = P \oplus P^*$ é a XOR do par de textos claros.
- T: Denota o texto cifrado antes da permutação final, que neste caso, também foi dispensada das definições iniciais. T^* denota qualquer outro texto cifrado correspondente aos textos claros $T^{'} = T \oplus T^*$ é a XOR do par de textos cifrados.
- (L,R): Corresponde às metades esquerdas e direitas respectivamente de cada texto claro P;
- $a, ..., j$: São os 32 bits referentes à entrada na função F em cada ciclo;
- $A, ..., J$: São os 32 bits referentes à saída da função F em cada ciclo;
- S_i: Denota as caixas S da função F, no caso do Papílio existe apenas uma caixa S com duas saídas chamadas (S_0, S_1);
- $S_{i_{KX}}, S_{i_{IX}}, S_{i_{OX}}$: $S_{i_{IX}}$ é a entrada em S_i em um ciclo X para $X \in a, ..., j$. $S_{i_{OX}}$ denota a saída do S_i no ciclo X.$S_{i_{KX}}$ denota os valores dos bits de cada sub chaves utilizada por cada função em um ciclo X.

Definição 4.1.1 ([Biham e Shamir 1990]) *Uma chave independente é uma lista de n sub chaves que não são necessariamente derivadas do gerenciador de chaves do algoritmo.*

Definição 4.1.2 ([Biham e Shamir 1990]) *A tabela de distribuição das saídas e entradas XOR's será formada pelas XOR's das entradas dos textos claros nas linhas e nas colunas as saídas XOR's dos pares de texto cifrados.*

As tabelas de distribuição de XOR de cada caixa S, apenas contem valores referentes ao algoritmo DES, previstos em [Biham e Shamir 1990] No caso do Papílio o Viterbi Modificado atua com uma caixa S que têm sempre a mesma estrutura, ou seja, independente do ciclo elas terão sempre as mesmas variações para os dados que nelas sejam guardados. No DES, a tabela de distribuição de cada caixa S contém 64 possíveis pares em 16 diferentes entradas, ou seja, a média de cada linha é exatamente 4, isto ocorre devido a cada caixa S ter no máximo um número com 6 bits de tamanho, podendo seu valor ser de 0 a 63, como mostra a figura 4.1.

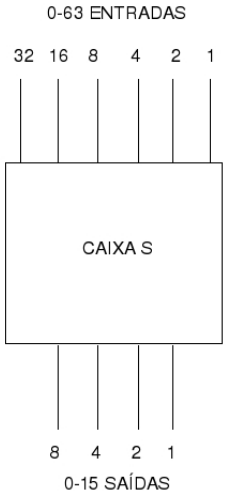

Figura 4.1: Caixa S

4.1.1 Diferença Entre os Pares de Textos

As entradas e saídas da tabela de distribuição são chamadas de diferenças dos pares de entrada em cada caixa S e diferença das saídas de cada caixa S. Logo se pode denominar como sendo ΔX a diferença dos pares de textos claros e ΔY a diferença de pares dos textos cifrados. Esta diferença é tratada da seguinte forma: Escolhidos dois textos claros legíveis, aplica-se sobre os mesmos o algoritmo de criptografia, em dois processos em paralelos, e a partir deste ponto analisam-se as entradas e saídas de cada etapa, aplicando a operação XOR[1] de cada

[1] Operação do ou - exclusivo conhecido pelo símbolo $\oplus$.

etapa referente a seu respectivo par. A figura 4.2 mostra como é aplicada e em que pontos são aplicados as operações XOR sobre as respectivas entradas e saídas em cada ciclo. Para melhor

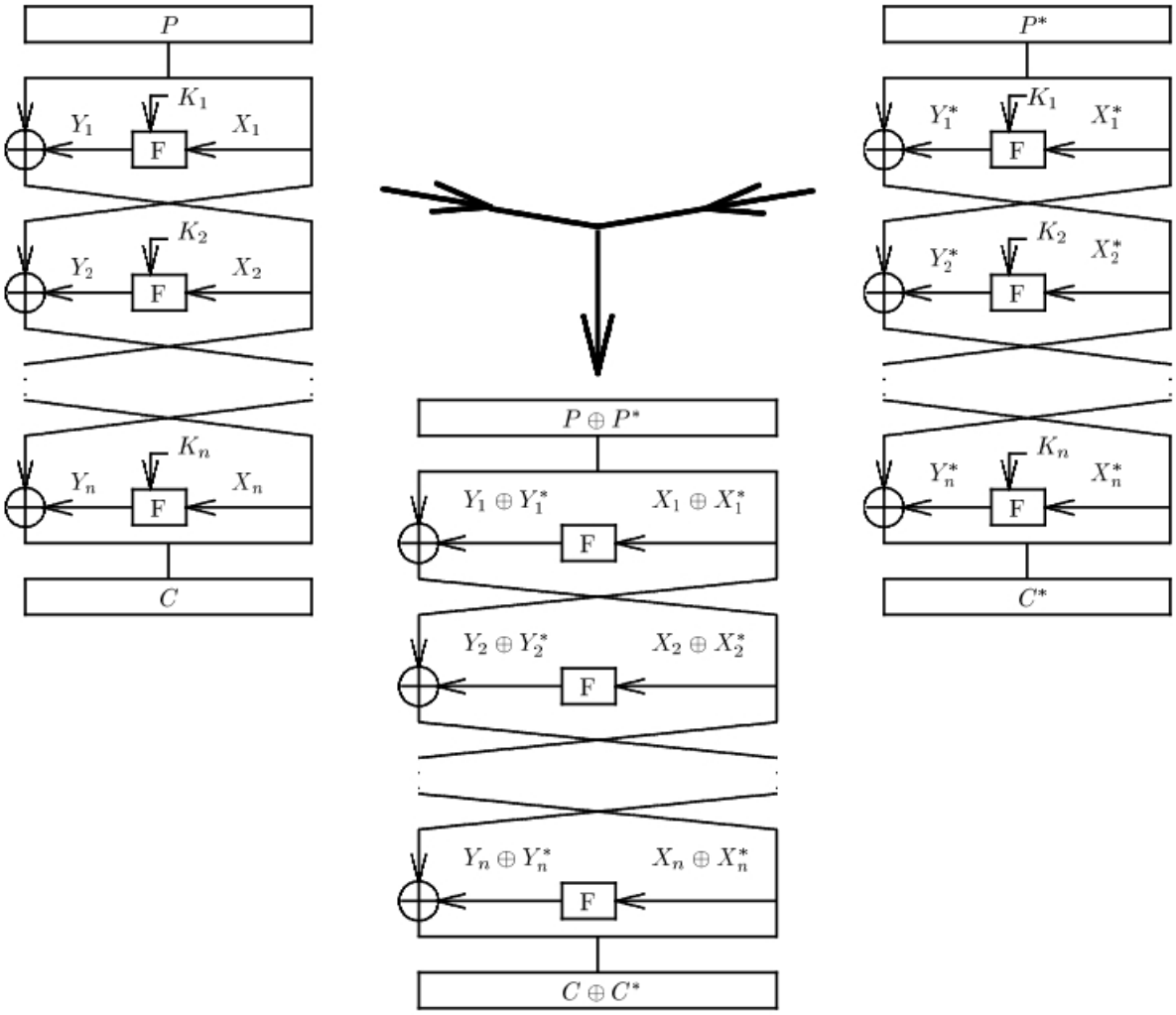

Figura 4.2: A diferença de textos aplicada em dois processos de cifragem com um número *n* de ciclos.

exemplificar as definições posteriores, adota-se a tabela de distribuição da caixa S_1 da função F do DES, como é vista em [Biham e Shamir 1990].

Definição 4.1.3 ([Biham e Shamir 1990]) *Diz-se que uma determinada diferença de textos claros (ΔX) **pode causar** uma diferença de textos cifrados (ΔY), se há na tabela 4.1, alguma entrada ΔX que tenha como saída um ΔY, denominado $\Delta X \rightarrow \Delta Y$, onde o valor de cada célula da tabela mostra quantos pares podem causar esta propriedade.*

Definição 4.1.4 ([Biham e Shamir 1990]) *Diz-se que uma determinada diferença de textos claros (ΔX) **não pode causar** uma diferença de textos cifrados (ΔY), se não há na figura 4.1, nenhuma entrada ΔX que tenha como saída um ΔY, denominado $\Delta X \nrightarrow \Delta Y$.*

Input XOR	Output XOR															
	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2_x	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3_x	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4_x	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5_x	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6_x	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7_x	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
8_x	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
9_x	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
A_x	0	8	6	2	2	8	6	0	6	4	6	0	4	0	2	10
B_x	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
C_x	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
D_x	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
E_x	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
F_x	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
									⋮							
30_x	0	4	6	0	12	6	2	2	8	2	4	4	6	2	2	4
31_x	4	8	2	10	2	2	2	2	6	0	0	2	2	4	10	8
32_x	4	2	6	4	4	2	2	4	6	6	4	8	2	2	8	0
33_x	4	4	6	2	10	8	4	2	4	0	2	2	4	6	2	4
34_x	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6
35_x	2	2	4	0	8	0	0	0	14	4	6	8	0	2	14	0
36_x	2	6	2	2	8	0	2	2	4	2	6	8	6	4	10	0
37_x	2	2	12	4	2	4	4	10	4	4	2	6	0	2	2	4
38_x	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39_x	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
$3A_x$	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
$3B_x$	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
$3C_x$	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
$3D_x$	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
$3E_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
$3F_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

Tabela 4.1: Tabela de distribuição da caixa S do DES.
Fonte: [Biham e Shamir 1990]

Definição 4.1.5 ([Biham e Shamir 1990]) *Diz-se que uma determinada diferença de textos claros (ΔX)* ***pode causar*** *uma diferença de textos cifrados (ΔY)* ***com certa probabilidade*** *p se uma fração p_0 dos pares os quais tiverem suas entradas na caixa S iguais a ΔX e suas saídas iguais a ΔY.*

Por exemplo, na figura 4.1 tem-se que $3_x \rightarrow 2_x$ resulta de 2 pares, ou seja, a probabilidade p de $3_x \rightarrow 2_x$ é igual a $\frac{1}{32}$. Segundo [Nakaraha 2006][apud ElSh93], esta definição não é válida necessariamente para qualquer cifra interativa, supõe-se, que nestes casos a fração dos valores de sub chaves seja muito próxima ao de p (probabilidade). Em [Biham e Shamir 1990], o 7º exemplo, apresentada a descoberta dos primeiros 6 bits da sub chaves do respectivo ciclo, através dos pares de textos de tamanho 6 bits que entram na caixa S_1 e de suas respectivas diferenças dos 4 bits de saída da mesma caixa S, inerentes com a probabilidade p e com os textos claros que tem a característica de acordo com a definição 4.1.3.

4.1.2 Características

Enquanto as probabilidades pertenciam apenas a uma única interação, as características servem para unir certas probabilidades em um ciclo com outra probabilidade no ciclo posterior,

probabilidade essa, que será diferente da anterior caso a sua entrada seja diferente da mesma.

Definição 4.1.6 ([Nakaraha 2006]) *Uma característica de N ciclos é uma tupla na forma de* $\Omega = (\Omega_P, \Omega_\Delta, \Omega_T)$ *onde* Ω_P *é a diferença de pares de textos claros,* Ω_T *é a diferença entre os pares de textos cifrados e* Ω_Δ *é uma lista de n elementos* $\Omega_\Delta = (\Delta_1, \Delta_2, \ldots, \Delta_n)$*, para cada* $\Delta_i = (\Delta X_i, \Delta Y_i)$*, onde* ΔX_i *e* ΔY_i *são as diferenças de entrada e saída de cada ciclo (na função F) respectivamente, sendo seus valores* $\frac{m}{2}$ *bits, como m sendo o tamanho de um bloco da cifra. Uma característica* Δ *de N ciclos satisfaz:*

$$\Omega_P = \begin{cases} (\Delta Z, \Delta X_1) & \text{se } N = 1 \\ (\Delta Y_1 \oplus \Delta X_2, \Delta X_1) & \text{se } N > 1 \end{cases}$$

$$\Omega_T = \begin{cases} (\Delta Z \oplus \Delta Y_1, \Delta X_1) & \text{se } N = 1 \\ (\Delta X_{N-1} \oplus \Delta Y_N, \Delta X_N) & \text{se } N > 1 \end{cases}$$

com $\Delta Z \in \{0,1\}^{\frac{m}{2}}$*, e, para todo i tal que* $2 \leq i \leq N-1 : \Delta X_{i-1} \oplus \Delta X_{i+1}$.

Definição 4.1.7 ([Nakaraha 2006] apud ElSh93) *O n-ciclo de uma característica* Ω *tem uma probabilidade associada* ${p_i}^{\Omega}$ *se* $\Delta X_i \rightarrow \Delta Y_i$ *tem uma probabilidade* ${p_i}^{\Omega}$ *pela função F.*

Definição 4.1.8 ([Nakaraha 2006] apud ElSh93) *Uma característica* Ω *de N ciclos tem probabilidade associada* p^{Ω} *se* p^{Ω} *é o produto das probabilidades de seus N ciclos.*

$$p^{\Omega} = \prod_{i=1}^{N} p_i^{\Omega}$$

Uma característica Ω em um determinado ciclo, pode valor 0_x (valor em hexadecimal), ou seja, tanto as entradas como as saídas de cada ciclo das XOR dos pares de texto claro e dos pares de textos cifrados são iguais a zero. Por exemplo, segundo [Nakaraha 2006] a cifra FEAL-N(X) assim como em DES ([Biham e Shamir 1990]) toda XOR de entrada em uma função F que seja igual a 0_x, tem obrigatoriamente sua XOR de saída também igual a 0_x. por exemplo, a figura 4.3 apresenta o caso que sempre a saída da função F será igual a 0_x desde que sua entrada também a seja. A figura 4.4 apresenta a estrutura com as respectivas possíveis características de cada ciclo e sua simetria. O lema de acordo com [Nakaraha 2006] demonstra a simetria nas características.

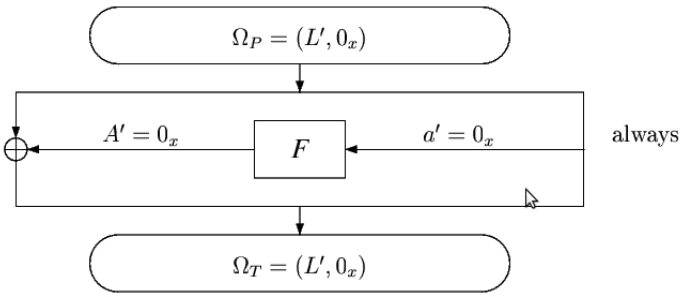

Figura 4.3: Um exemplo com probabilidade 1, no caso da entrada igual a O_x
Fonte: [Biham e Shamir 1990]

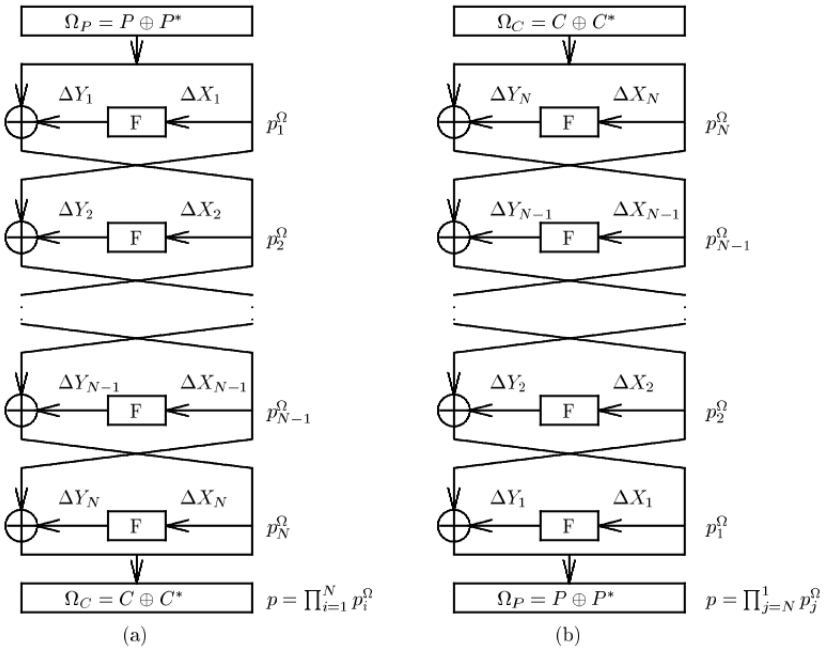

Figura 4.4: a) Característica de N ciclos. b) Característica simétrica correspondente
Fonte: [Nakaraha 2006]

Lema 4.1.9 ([Nakaraha 2006]) *Se $\Omega = (\Omega_P, \Omega_\Delta, \Omega_T)$ é uma característica diferencial de N ciclos com probabilidade p, então, $\Omega^* = (\Omega_T, \Omega_\Delta^{-1}, \Omega_P)$ também é uma característica diferencial de N ciclos com a mesma probabilidade p, onde Ω_Δ^{-1} corresponde à lista invertida de diferenças Ω_Δ de um ciclo, i.e. $\Omega_\Delta^{-1} = (\Delta_N, \Delta_{N-1}, \ldots, \Delta_1)$, onde se substituiu o par de diferenças Δ_i por $\Delta_{N-i+1}, 1 \leq i \leq N$.*

Definição 4.1.10 ([Biham e Shamir 1990]) *A concatenação de uma característica de N ciclos $\Omega^1 = (\Omega_P^1, \Omega_\Lambda^1, \Omega_T^1)$ com uma caraterística de M ciclos $\Omega^2 = (\Omega_P^2, \Omega_\Lambda^2, \Omega_T^2)$ é possível se Ω_T^1 for igual Ω_P^2 com as semi diferenças trocadas, caso seja possível a característica resulta em $\Omega^1 = (\Omega_P^1, \Omega_\Lambda, \Omega_T^2)$ onde Ω_Λ é a concatenação das listas Ω_Λ^1 e Ω_Λ^2.*

Desta forma é permitido saber a probabilidade de qualquer N ciclos através do produto das probabilidades das características pelo produto de N ciclos separadamente.

Definição 4.1.11 ([Biham e Shamir 1990]) *Um **par correto** em relação a uma característica $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ de N ciclos, e a uma chave independente[2] K e um par $P' = \Omega_P$ para os primeiros N ciclos da cifragem do par usando a chave independente K, a entrada XOR do i-esimo ciclo é λ_I^i, e a saída XOR da função F é igual a λ_O^i . Todo par que não possua essas características é chamado de **par errado** em relação à característica e à chave independente.*

Definição 4.1.12 ([Biham e Shamir 1990]) *Um i-esimo ciclo de uma característica Ω tem probabilidade p_i^Ω se $\lambda_I^i \rightarrow \lambda_O^i$ tiver probabilidade p_i^Ω da função F.*

Definição 4.1.13 ([Biham e Shamir 1990]) *A probabilidade definida formalmente de uma característica $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ é a probabilidade atual para qualquer par de textos claros fixados que satisfazem $P' = \Omega_P$ em um par correto quando sub chaves aleatórias independentes são usadas.*

O que a definição acima quer dizer é que a probabilidade de $P' = \Omega_P$ é igual em todos os N ciclos onde $\Delta X_n \rightarrow \Delta Y_n$, isto implica que a probabilidade de cada ciclo independe do valor exato da entrada e também independe do ciclo anterior, pelo fato das sub chaves utilizadas também serem independente, tornando as caixas S_i também independentes. Segundo [Nakaraha 2006] utilizando-se uma característica considerada longa, e certo número de pares corretos, pode-se calcular a diferença de entrada da função F no último ciclo. Como somente a XOR dos pares

[2]Uma chave independente representa uma lista de sub chaves, suficientes para todos os ciclos de um processo de cifragem onde cada bit é independente dos demais, ou seja, não há correlação entre os bits da sub chave.

corretos são conhecidas, apenas os pares que tenham como saída algo esperado pela característica, será um par correto e cada par correto fornece o valor correto da última sub chave. Caso contrário será considerado par errado quando informa valores aleatórios, logo pode-se identificar a sub chave pelo maior número de repetições nas saídas dos pares. Ainda segundo [Nakaraha 2006] a questão a ser analisada para que possa ocorrer dentro dos padrões a descoberta ou a possível descoberta da sub chave, é a possibilidade de determinar os valores de entrada em cada caixa S antes da combinação dos bits da sub chave. O objetivo desta estratégia é conhecer a XOR de saída de cada caixa S, $S_i, S_i(X \oplus K) \oplus S_i(X \oplus K)$ que é obtida pela saída parcial de F, e a partir deste ponto determinar os prováveis valores que causaram a XOR. Com isso sabendo-se os valores de um par $(X^{'} \oplus K, X^{*} \oplus K)$ provável e os valores $(X^{'}, X^{*})$ conhecidos, obtém-se os possíveis bits das sub chaves $K_1 = (X^{'} \oplus K) \oplus X^{'}$ e $K_2 = (X^{*} \oplus K) \oplus X^{*}$. Resumindo, ao saber as possíveis saídas das caixas S antes da XOR com as chaves, é possível através de verossimilhança descobrir os possíveis bits de cada sub chave naquele ciclo. Mais detalhes sobre este tipo de ataque em [Nakaraha 2006].

4.1.3 Características Diferenciais Iterativas

Definição 4.1.14 ([Biham e Shamir 1990]) *Uma característica $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ é dita iterativa quando as partes (L_0, R_0) de Ω_P forem iguais às partes de Ω_T.*

Pode-se concatenar este tipo de característica a si mesmo inúmeras vezes, o que permite construir uma característica para n ciclos para qualquer valor de n.

Definição 4.1.15 ([Nakaraha 2006]) *Uma característica iterativa $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ tem **tamanho ou comprimento n** ($n > 1$) se n é o menor valor inteiro (número de ciclos) tal que:*

- *se $n = 1$, então, $\Delta X_1 = 0$ e $\Delta P_L = \Delta P_R = \Delta T_L = \Delta T_R$;*
- *se $n > 1$, então, $\Delta X_1 = \Delta X_{n-1} \oplus \Delta Y_n$ e $\Delta X_n = \Delta Y_1 \oplus \Delta X_2$ onde $\Omega_P = (\Delta P_L, \Delta P_R)$, $\Omega_T = (\Delta T_L, \Delta T_R)$ e $\Omega_\Delta = \{(\Delta Y_1, \Delta X_1), \ldots, (\Delta Y_n, \Delta X_n)\}$.*

A figura 4.5 representa um exemplo genérico que contém uma característica iterativa para certa probabilidade p. No caso de 5 ciclos, por exemplo, a característica teria sua probabilidade multiplicada com cada probabilidade dos 4 ciclos posteriores.

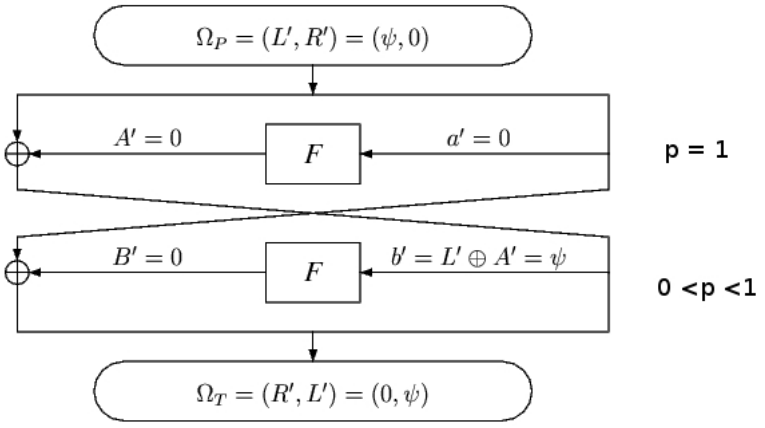

Figura 4.5: Característica iterativa
Fonte: [Nakaraha 2006]

4.1.4 Razão Sinal-Ruído (S/N)

As características têm comportamento probabilístico, logo, não é possível às vezes obter a intersecção de todos os bits das sub chaves através dos pares de textos claros, pois, quando a característica é menor que o número de ciclos é impossível identificar os pares corretos e a interseção das sub chaves é geralmente vazia. Os ataques que tenham características menores que a cifra procuram reduzir a cifra ao mesmo tamanho da característica.

Definição 4.1.16 ([Nakaraha 2006]) *Um ataque diferencial, que utilize uma característica que possui iR ciclos a menos que a cifra sendo analisada, é chamado de ataque diferencial do tipo iR.*

Definição 4.1.17 ([Biham e Shamir 1990]) *A razão entre o número de pares corretos e a contagem média em um processo de contagem é chamada razão de sinal-ruído deste esquema, é denominado por S/N.*

Para encontrar os bits da chave é necessário de uma característica com uma probabilidade ótima e pares de cifras suficientes para garantir a existência de alguns pares corretos. Como o objetivo é encontrar os bits da chave k pode-se então contar a ocorrência de 2^k possíveis valores em 2^k contadores. Como cada contador contém uma média de $\frac{m.\alpha.\beta}{2^k}$, onde m é o número de pares, α é a contagem média por par contado e β é a razão dos contados para todos os pares, tem-se que S/N é dado da seguinte forma:

$$S/N = \frac{m.p}{m.\alpha.\beta} = \frac{2^k.p}{\alpha.\beta}$$

O valor da chave correta é contado aproximadamente $m.p$ vezes usando os pares corretos onde p é a probabilidade da característica. O colário apresentado em [Biham e Shamir 1990], define que esta fórmula, o sinal para a razão do ruído de um esquema de contagem, é independente da quantidade de pares usados no esquema, já o outro é que diferentes esquemas de contagem que se baseiam na mesma característica, mas com um diferente número de bits da sub chave tem diferente S/N. O S/N quando é muito elevado mostra que poucas ocorrências de pares corretos são necessárias para identificar o valor correto dos bits da sub chave, logo caso o S/N seja muito baixo, será necessário um valor muito alto de ocorrências de pares corretos para identificar o valor da sub chave. Segundo [Nakaraha 2006] estes ataques utilizam mais de uma característica e para reduzir o numero de textos claros utilizados utilizam-se de estruturas que agrupam XOR dos pares de textos que satisfazem mais de uma característica.

Definição 4.1.18 ([Biham e Shamir 1990]) *Um **quarteto** é uma estrutura de quatro textos cifrados que simultaneamente contém dois pares de uma característica e dois pares de outra característica.*

O exemplo encontrado [Biham e Shamir 1990] tem quatro textos cifrados necessárias para chegar a um quarteto (onde ψ_1 e ψ_2 são XOR dos textos claros das características distintas), onde $\psi_1 = \Omega_P^1$ de $\Omega^1 = (\Omega_P^1, \Omega_\Lambda^1, \Omega_T^1)$ e $\psi_2 = \Omega_P^2$ de $\Omega^2 = (\Omega_P^2, \Omega_\Lambda^2, \Omega_T^2)$ então,

1. P
2. $P \oplus \Omega_P^1$
3. $P \oplus \Omega_P^2$
4. $P \oplus \Omega_P^1 \oplus \Omega_P^2$

Cada elemento cifrado forma um quarteto, sendo que $(1) \oplus (3)$ e $(2) \oplus (4)$ formam dois pares de Ω_P^2, e, $(1) \oplus (2) e (3) \oplus (4)$ formam dois pares de Ω_P^1.

Definição 4.1.19 ([Biham e Shamir 1990]) *Um **octeto** é uma estrutura de oito textos cifrados que simultaneamente contém quatro pares de uma característica e quatro pares de outra característica.*

Seguindo o exemplo anterior pode-se ter em três características distintas $\Omega^1, \Omega^2, \Omega^3$ as seguintes premissas.

1. P

2. $P \oplus \Omega_P^1$
3. $P \oplus \Omega_P^2$
4. $P \oplus \Omega_P^3$
5. $P \oplus \Omega_P^1 \oplus \Omega_P^2$
6. $P \oplus \Omega_P^1 \oplus \Omega_P^3$
7. $P \oplus \Omega_P^2 \oplus \Omega_P^3$
8. $P \oplus \Omega_P^1 \oplus \Omega_P^2 \oplus \Omega_P^3$

Com isso após a cifragem de cada elemento forma-se um octeto onde:
$(1,2),(3,5),(4,6)$ e $(7,8)$ são pares de Ω_P^1
$(1,3),(2,5),(4,7)$ e $(6,8)$ são pares de Ω_P^2
$(1,4),(2,6),(3,7)$ e $(5,8)$ são pares de Ω_P^3
De acordo com [Nakaraha 2006] a utilização de quartetos em ataques do tipo de texto claro escolhido fornece 8 pares, ou seja, permite economizar 50% dos pares de textos necessários. Através dos octetos obtém 12 pares o que reduz em $\frac{2}{3}$ o número de pares necessários.

Definição 4.1.20 ([Nakaraha 2006]) *Um tipo de quarteto que contem quatro pares de quatro características distintas* Ω^1,Ω^2,Ω^3 *e* Ω^4 *onde* $\Omega^4 = \Omega^1 \oplus \Omega^2 \oplus \Omega^3$ *é chamado de* ***quarteto especial***

No caso das redes FEAL-N(X) segundo [Nakaraha 2006] existe um quarteto especial, onde as características envolvidas em sua totalidade têm probabilidade igual a 1: São elas:
$\Omega^1 = ((\Delta L, 00000000_x), \{00000000_x \rightarrow 00000000_x\}, (\Delta L, 00000000_x))$
$\Omega^2 = ((\Delta L, 80808080_x), \{80808080_x \rightarrow 02000002_x\}, (\Delta L \oplus 02000002_x, 80808080x))$
$\Omega^3 = ((\Delta L, 80800000_x), \{80800000_x \rightarrow 02000000_x\}, (\Delta L \oplus 02000000_x, 80800000_x))$
$\Omega^4 = ((\Delta L, 00008080_x), \{00008080_x \rightarrow 00000002_x\}, (\Delta L \oplus 00000002_x, 00008080_x))$
onde ΔL representa um valor arbitrário de 32 bits. De forma genérica:

1. P
2. $P \oplus \Omega_P^1$
3. $P \oplus \Omega_P^2$
4. $P \oplus \Omega_P^3$

5. $P \oplus \Omega_P^1 \oplus \Omega_P^2$
6. $P \oplus \Omega_P^1 \oplus \Omega_P^3$
7. $P \oplus \Omega_P^2 \oplus \Omega_P^3$
8. $P \oplus \Omega_P^1 \oplus \Omega_P^2 \oplus \Omega_P^3 = P \oplus \Omega_P^4$

onde após cada elemento ter sido cifrado, forma-se um quarteto especial, sendo que:
$(1,2),(3,5),(4,6)$ e $(7,8)$ são pares de Ω_P^1
$(1,3),(2,5),(4,7)$ e $(6,8)$ são pares de Ω_P^2
$(1,4),(2,6),(3,7)$ e $(5,8)$ são pares de Ω_P^3
$(1,8),(2,7),(3,6)e(4,5)$ são pares de Ω_P^4
Logo o quarteto especial tem como resultados 16 pares, o que resulta em uma economia de 75% no número de pares para um ataque.

4.1.5 Criptoanálise Diferencial Exclusivamente Através dos Textos Claros Específicos

Como a técnica da criptoanálise diferencial baseia-se nas escolhas de textos claros e as diferenças de suas cifras, quando utiliza-se somente textos claros que têm seu comportamento esperado, esses pares podem ser ditos como ideais para a criptoanálise diferencial, ou seja, somente esses pares podem satisfazer uma determinada diferença. De acordo com [Nakaraha 2006], havendo uma necessidade de um ataque diferencial, necessita-se ter **m** pares escolhidos, que sejam fornecidos $2^{32} \cdot \sqrt{2 \cdot m}$ textos claros de forma aleatória juntamente com suas respectivas cifras. Ao considerar todos os $\frac{2^{32} \cdot \sqrt{2 \cdot m}}{2} = 2^{64} \cdot m$ possíveis pares de textos claros, isso com 64 bits de tamanho, existem 2^{64} possíveis XOR de textos claros, logo apresenta em média $m = 2^{97}$ pares de cada XOR possível entre os pares inicialmente fornecidos. Em [Nakaraha 2006] é apresentada uma tabela com a probabilidade de características utilizada e a quantidade de textos claros escolhidos para ataques 1R e 2R (características reduzidas de 1 e 2 ciclos), para FEAL-N(X) para certos valores de N.

Definição 4.1.21 ([Nakaraha 2006]) *Uma cifra será dita **imune** ou **resistente** a um determinado tipo de ataque se o custo computacional (memória, tempo de processamento, número de operações aritméticas) para quebrar a cifra pelo ataque for maior ou igual ao custo de uma busca exaustiva.*

Assim como em qualquer ataque, e não diferentemente da criptoanálise aplicada tanto por [Biham e Shamir 1990] ou [Nakaraha 2006], a busca exaustiva é colocada como limite sobre o status de uma determinada técnica ou ataque quanto a sua complexidade ou viabilidade.

4.1.6 Variante de Ataque Diferencial

Existe uma variante da técnica de criptoanálise diferencial quanto à questão das diferenças dos pares de textos claros e de suas cifras, que é certo diferencial entre os mesmos.

Definição 4.1.22 ([Nakaraha 2006]) *Um diferencial de n ciclos é um par* (α,β) *onde* α *representa uma diferença de dois textos claros* (X,X^*) *e* β *representa uma diferença possível dos dois textos cifrados resultantes após n ciclos* (Y_n,Y_n^*) *A probabilidade de um diferencial de n ciclos* (α,β) *é a probabilidade condicional de que* $\beta = (Y_n \oplus Y_n^*)$ *após n ciclo dado que o par de textos claros* (X,X^*) *tem diferença* $(X \oplus X^*)$ *onde o texto claro X e as sub chaves* $K_1,K_2,\ldots,K_n$ *utilizadas são consideradas independentes e uniformemente aleatórias. Denota-se tal probabilidade por:*

$$P(\Delta Y_n = \beta | \Delta X = \alpha)$$

Esta característica de n ciclos pode ser vista como uma $(i+1)$-upla $(\alpha,\beta_1,\ldots,\beta_n)$, ao invés de $(\Delta X,\Delta Y_1,...,\Delta Y_n)$. A seguir, o procedimento básico de um ataque do tipo diferencial para uma determinada cifra iterativa de n ciclos é um tanto quanto semelhante ao da técnica de criptoanálise diferencial, que utiliza-se de características e não de diferenciais.

1. encontrar um diferencial (α,β) de (r-1) ciclos tal que $P(\Delta Y_{r-1} = \beta | \Delta X = \alpha)$ seja o maior possível exemplo de um ataque 1R;

2. escolher um texto claro X de forma uniformemente aleatória é calcular X^* através da diferença pré estabelecida $\Delta X = \alpha = X \oplus X^*$. Cifras X e X^* com a mesma chave K. A partir dos textos cifrados Y_r e Y_r^* verificar cada valor possível da sub chave K_r do último ciclo em relação à diferença $\Delta Y_{n-1} = \beta$ esperada. Incrementar um contador do número de vezes em que tal valor de K_r^j resulta na diferença esperada.

3. repetir o passo 2 até que um ou mais valores dos bits de sub chave K_r^j sendo explorados sejam contados mais frequentemente que os demais candidatos. Considere este candidato ou conjunto de candidatos cujo(s) contador(es) é(são) o (os) maior(es) possível(eis) como o(s) valor(es) mais provável(eis) ao valor de K_r.

Segundo [Nakaraha 2006] a tabela 4.2 mostra o esquema da técnica do diferencial, sem se preocupar com os ciclos intermediários, considerando somente as entradas e saídas mais relevantes.

verifica se valor de diferença de saída $(\Delta C_R, \Delta C_L \oplus \Delta F(C_R, K_n))$ é a esperada						
N pares de textos cifrados	candidatos a bits de subchave					
	K_n^1	K_n^2	K_n^3	...	$K_n^{2^{t-1}}$	$K_n^{2^t}$
(C_1, C_1^*)	ok		ok	...	ok	
(C_2, C_2^*)		ok		...	ok	ok
(C_3, C_3^*)	ok	ok		...		ok
...	...	...	...	...	...	...
(C_{N-1}, C_{N-1}^*)		ok	ok	...	ok	
(C_N, C_N^*)	ok	ok		...	ok	ok
escolhe valor máximo	$\sum_{i=1}^N$ ok	$\sum_{i=1}^N$ ok	$\sum_{i=1}^N$ ok	...	$\sum_{i=1}^N$ ok	$\sum_{i=1}^N$ ok

Tabela 4.2: Característica iterativa

Fonte: [Nakaraha 2006]

Segundo [Nakaraha 2006] pode-se extrair a seguinte hipótese a respeito desta técnica:

Hipótese 1 *Para um diferencial* (α, β) *de (r-1) ciclos,*

$$P(\Delta Y_{r-1} = \beta | \Delta X = \alpha) \approx P(\Delta Y_{r-1} = \beta | \Delta X = \alpha, K_1 = \omega_1, \ldots, K_{r-1} = \omega_{r-1})$$

para todos os conjuntos possíveis de sub chaves $(\omega_1, \omega_1, \ldots, \omega_{r-1})$.

SA diferença de um diferencial tem-se que a mesma é independente do conjunto de sub chaves utilizadas, sendo influenciada, pela diferença no diferencial e pela estrutura interna da cifra [Nakaraha 2006].

4.2 Principais Aplicações da Criptoanálise Diferencial

Para posteriores estudos e comparações feitas ao Papílio, será apresentado a seguir de forma sucinta, o resultado da criptoanálise diferencial de três algoritmos criptográficos, evidenciando apenas as características inerentes a cada cifra com suas respectivas aplicações da criptoanálise diferencial. São eles o DES, FEAL-N(X) e o BlowFish. Porém mais detalhes serão descritos no capítulo 4 quando torna-se necessário a comparação em determinados requisitos junto ao Papílio, explicitando suas características positivas e negativas quanto aos três algoritmos.

4.2.1 Criptoanálise Aplicada ao DES

Os resultados apresentados neste trabalho sobre o DES resultam de Biham e Shamir em [Biham e Shamir 1992]. A criptoanálise aplicada ao DES consegue quebrar a cifra em 16 ciclos completos com uma complexidade menor que 2^{55}, que é a complexidade da busca exaustiva. Segundo os autores a fase de análise computa a chave através de 2^{36} textos cifrados em 2^{37} vezes. Estes 2^{36} textos cifrados de um universo de 2^{47} possíveis textos cifrados, são escolhidos durante a fase de coleção de dados. Um resumo sobre a complexidade da criptoanálise aplicada ao DES está na tabela 4.3, essa complexidade é medida pela quantidade de pares de textos necessários para quebrar a chave em uma quantidade determinada de ciclos. Em [Biham e Shamir 1990] é apresentado, de forma completa, a técnica de criptoanálise aplicada, tendo seu início, a partir do 4º ciclo, a fase de coleta de dados, tentativa de quebra das sub chaves utilizadas até este ciclo, com a utilização dos padrões (probabilidades) encontrados na fase de coleta de dados juntamente com a informação de cada par, se o mesmo era certo ou errado, etc. Nota-se que no

Rounds	Chosen Plaintexts	Analyzed Plaintexts	Complexity of Analysis	Best Previous	
				Time	Space
8	2^{14}	4	2^{9}	2^{16}	2^{24}
9	2^{24}	2	2^{32}	2^{26}	2^{30}
10	2^{24}	2^{14}	2^{15}	2^{35}	–
11	2^{31}	2	2^{32}	2^{36}	–
12	2^{31}	2^{21}	2^{21}	2^{43}	–
13	2^{39}	2	2^{32}	2^{44}	2^{30}
14	2^{39}	2^{29}	2^{29}	2^{51}	–
15	2^{47}	2^{7}	2^{37}	2^{52}	2^{42}
16	2^{47}	2^{36}	2^{37}	2^{58}	–

Tabela 4.3: Resumo da criptoanálise do DES

Fonte: [Biham e Shamir 1992]

momento em que o número de ciclos aumenta o número de pares de textos escolhidos também aumenta. Por exemplo, enquanto a criptoanálise reduzida ao 4º ciclo o número de pares é igual a 2^{14} e a criptoanálise reduzida ao 11º tem o número de pares igual a 2^{31}. No entanto quando refere-se ao número de pares analisados, devido às possíveis probabilidades e características encontradas, já não se pode afirmar a mesma coisa, pois nos ciclos 8º e 9º, por exemplo, o número de pares de textos analisados são respectivamente 4 e 2 porém as suas respectivas com-

plexidades são 2^9 e 2^{32}, para uma mesma quantidade de bits encontrado da chave.

Na técnica de criptoanálise diferencial há uma diminuição forte do número de bits possíveis para uma determinada sub chave, ou seja, ela reduz a busca exaustiva a tentativas de determinados bits, diminuindo assim o universo possível de bits, acarretando em uma complexidade menor. Mais detalhes da aplicação estão apresentados nos trabalhos [Biham e Shamir 1990] e [Biham e Shamir 1992].

4.2.2 Criptoanálise Aplicada ao FEAL

Segundo [Nakaraha 2006][apud [AoOh95],[LaHe94]], os algoritmos dos ataques diferenciais para as cifras FEAL-8(x) foram implementados em linguagem C, para alguns valores de número de quartetos, isso supondo que as chaves tenham tamanho de 64 ou 128 bits com ataques do tipo 1R e 2R. A tabela 4.4 apresenta alguns resultados experimentais para FEAL-8 e para FEAL-8(X).

ataque diferencial-linear				
cifra	FEAL-8		FEAL-8X	
tipo de ataque	1R	2R	1R	2R
# quartetos				
2	-	-	-	-
3	-	-	-	-
4	ok	ok	-	-
5	ok	ok	-	ok
6	ok	ok	ok	ok
7	ok	ok	ok	ok
8	ok	ok	ok	ok
9	ok	ok	ok	ok
10	ok	ok	ok	ok
11	ok	ok	ok	ok

Tabela 4.4: Resultados experimentais para FEAL-8 e para FEAL-8(X).

Fonte: [Nakaraha 2006]

Onde o “ok” indica a possibilidade de determinar as sub chaves completas da cifra e o valor “-” indica que o ataque foi realizado mas não foi possível encontrar algum valor. Na tabela 4.5 está o resumo geral sobre a criptoanálise diferencial aplicada às cifras FEAL-N(X) somente para ataques do tipo 1R e 2R. Segundo [Nakaraha 2006] a tabela 4.5 pode-se veri-

parâmetros de ataques diferenciais para FEAL-N(X)/RP						
	ataque 2R			ataque 1R		
N	prob. da caract.	S/N	número de pares texto escolhidos	prob. da caract.	S/N	número de pares texto escolhidos
8	2^{-30}	2^{21}	2^{32}	2^{-35}	2^{16}	2^{37}
9	2^{-35}	2^{16}	2^{37}	2^{-40}	2^{11}	2^{42}
10	2^{-40}	2^{11}	2^{42}	2^{-45}	2^{4}	2^{47}
11	2^{-45}	2^{4}	2^{47}	2^{-50}	2^{-1}	2^{52}
12	2^{-50}	2^{-1}	2^{52}	2^{-55}	—	2^{57}
13	2^{-55}	—	2^{57}	2^{-60}	—	2^{62}
14	2^{-60}	—	2^{62}	2^{-65}	—	2^{67}
15	2^{-65}	—	2^{67}	2^{-70}	—	2^{72}
16	2^{-70}	—	2^{72}	2^{-75}	—	2^{77}
17	2^{-75}	—	2^{77}	2^{-80}	—	2^{82}
18	2^{-80}	—	2^{82}	2^{-85}	—	2^{87}
19	2^{-85}	—	2^{87}	2^{-90}	—	2^{92}
20	2^{-90}	—	2^{92}	2^{-95}	—	2^{97}

Tabela 4.5: Resumo da criptoanálise do FELA-N(X).

Fonte: [Nakaraha 2006]

ficar que pela característica iterativa utilizada, as cifras FEAL-N(X) apresentam-se imunes a criptoanálise diferencial a partir do 15^{o} ciclo, devido à grande quantidade de número de textos claros escolhidos, i.e. em número de textos cifrados já superior ao custo de busca exaustiva pela chave (complexidade igual 2^{64}).

4.2.3 Criptoanálise Diferencial Aplicada no Blowfish

A criptoanálise diferencial aplicada no BlowFish apresentada em [Oliveira 2003] foi abordada de maneira sucinta pelo autor, o mesmo utiliza-se de textos com tamanho de 16 bits divididos em 4 blocos de 4 bits. A tabela de distribuição das XOR's da entradas e saídas foi feita através de um algoritmo simples que foi implementado pelo próprio autor. Através desta tabela foi possível obter todas as probabilidades das XOR's das entradas e as XOR das saídas dos textos cifrados, logo, conseqüentemente as características de cada par. A tabela 4.6 contém

Subchave Parcial $[K_{5,5} \mathsf{K}\ K_{5,8}, K_{5,13} \mathsf{K}\ K_{1,16}]$	**Prob**	**Subchave Parcial** $[K_{5,5} \mathsf{K}\ K_{5,8}, K_{5,13} \mathsf{K}\ K_{1,16}]$	**Prob**
1C	0,0000	2^{A}	0,0032
1D	0,0000	2B	0,0022
1E	0,0000	2C	0,0000
1F	0,0000	2D	0,0000
20	0,0000	2E	0,0000
21	0,0136	2F	0,0000
22	0,0068	30	0,0004
23	0,0068	31	0,0000
24	0,0244	32	0,0004
25	0,0000	33	0,0004
26	0,0068	34	0,0000
27	0,0068	35	0,0004
28	0,0030	36	0,0000
29	0,0024	37	0,0008

Tabela 4.6: Resultado para o ataque diferencial do BlowFish.

Fonte: [Oliveira 2003]

o resultado sobre a probabilidade das possíveis sub chaves para todos os ciclos do BlowFish. Após esta fase, foram feitas análises sobre cada caixa S, das 4 caixas S pertencentes ao BlowFish (mais detalhes em [Oliveira 2003]). Os ataques foram feitos baseados em dois fatores: os ataques de chaves fracas com a função F conhecida e os ataques de chave aleatória com a função F também conhecida. Em resumo a tabela 4.7 apresenta para cada quantidade de ciclos, o número de textos requeridos para a descoberta das sub chaves utilizadas.

Número de Rodadas (*t*)	**Texto Claro Requerido**
1	3×2^{1}
2	3×2^{2}
3	3×2^{9}
4	3×2^{9}
5	3×2^{16}
6	3×2^{16}
7	3×2^{23}
8	3×2^{23}
9	3×2^{30}
10	3×2^{30}
11	3×2^{37}
12	3×2^{37}
13	3×2^{44}
14	3×2^{44}
15	3×2^{51}
16	3×2^{51}

Tabela 4.7: Resumo da quantidade necessária de textos claros.

Fonte: [Oliveira 2003]

4.2.4 Considerações Sobre a Aplicação da Criptoanálise Diferencial

A criptoanálise apresentada no DES, foi uma das primeiras criptoanálises aplicadas a uma cifra de bloco, e teve um resultado satisfatório, onde pode-se obter a complexidade menor que a busca exaustiva, já na criptoanálise aplicada ao FEAL-N(X), o autor uniu a Criptoanálise Diferencial com a Criptoanálise Linear, pela necessidade de um poder maior através da união das duas, com resultados mais precisos sobre quais eram as possíveis sub chaves de cada ciclo.

Na criptoanálise do BlowFish, o autor decidiu reduzir o tamanho do texto e conseqüentemente o tamanho das sub chaves utilizadas no processo de cifragem, para facilitar sua análise, utilizando somente a Diferencial, conseguindo como resultado a complexidade do BlowFish. Esta técnica é considerada poderosas e eficaz assim como é apresentado em [Nakaraha 2006], [Biham e Shamir 1990], [Dunkelman, Biham e Keller 2007] dentre outros, com resultados expressivos, relevantes e necessários para um funcionamento eficaz na respectivas implementações.

O comparativo dos resultados e análises feitas sobre o DES, BlowFish e FEAL-N(X) será apresentado no capítulo 4 juntamente com os resultados da criptoanálise diferencial aplicada ao Papílio.

5 *Criptoanálise do Papílio*

Tendo como base as bibliografias apresentadas nos capítulos anteriores e as mesmas como requisitos de uma boa base teórica juntamente com um certo conhecimento desejável em criptografia, em suas definições mais básicas, será apresentado neste capítulo a aplicação da criptoanálise diferencial sobre o Papílio do 1º ao 16º ciclo evidenciando características de cada ciclo os pares corretos juntamente com suas probabilidades adquiridas através da Tabela de distribuição.

5.1 Modificações Necessárias para Criptoanálise Diferencial

Inicialmente o Papílio, como foi observado na seção 3, tem 64 bits de tamanho do seu bloco de entrada, logo apenas textos múltiplos de 8 caracteres ou seqüências binárias de 64 bits poderão ser criptografas. Com isso, cada semi bloco de entrada em cada função F terá um tamanho de 32 bits, o que leva a uma combinação de 32 bits de entrada por 32 bits de saída de cada função F como foi visto na figura 3.8.

Contudo, realizam-se séries de modificações na estrutura do programa Viterbi (Papílio Original) apresentado em [Ramos 2002], fpara que possa ser feita a criptoanálise. As modificações foram as seguintes:

1. Mudança na entrada de dados: para garantir o controle total sobre quais textos estariam entrando em processo de criptografia no ato da execução, modificou-se o programa de forma a permitir várias seqüências de bits ao mesmo tempo, lidas a partir de um arquivo, tratando cada linha como um processo separadamente, de forma a emitir o log de cada seqüência binária.

2. Mudança de armazenamento: o log gerado pelo programa, ao invés de ser apresentado em tempo de execução, apenas era gravado em um arquivo texto, como também eliminaram-se informações desnecessárias para futuras análises, melhorando assim seu tempo de execução e ganho no uso de memória da máquina.

3. Mudança no número de bits de entrada, ou seja, o tamanho do bloco de entrada: A priore, o Papílio trata apenas 64 bits de texto ou palavras múltiplas de 8 caracteres. Nessa situação a quantidade de texto necessários para gerar a tabela de distribuição será igual a 2^{32} devido ao tamanho do semi bloco de entrada na função F. Neste momento esta situação não pode ser concluída por questões computacionais de armazenamento e leitura de dados resultantes do bloco de 64 bits de entrada. A mudança necessária habilitará o programa a criptografar com os mesmos padrões de criptografia oferecidas pelo Papílio à sentenças binárias que sejam múltiplas de 8, 16 e 32 bits de entrada, conseqüentemente textos múltiplos de 1, 2 ou 4 caracteres.

4. A geração de sub chaves: de acordo com o tamanho de bits da entrada, o tamanho de cada sub chave é alterado de forma a manter o padrão de bits necessários para as operações XOR's contidas no programa.

Essas mudanças permitiram a aplicação da técnica de criptoanálise diferencial, possibilitando assim, construir ou obter informações baseadas no comportamento do algoritmo diante de pares, operações e logs gerados durante o processo de criptoanálise. A primeira questão a ser levantada em todo algoritmo de criptografia que se submete à criptoanálise diferencial é a elaboração das tabelas de distribuição, que será apresentado a seguir.

5.2 Tabela de Distribuição

A tabela de distribuição segundo [Biham e Shamir 1990] é dada por conjuntos de XOR's dos pares de entrada pela XOR das saídas dos respectivos pares de entrada. Na figura 4.1 que apresenta a tabela de distribuição do DES, foi feito uma tabela de distribuição para cada caixa-S, com 64 possíveis XOR's de entrada com 16 possíveis XOR de saída de cada caixa-S pelo fato de cada caixa-S do DES ter como entrada e saída 6 e 4 bits respectivamente. No Papílio isso é diferente, será feita apenas uma tabela de distribuição, devido ao fato do Viterbi Modificado funcionar como apenas uma caixa de substituição, semelhante às do DES. Como pode ser visto na figura 5.1, a linha pontilhada mostra a entrada e a saída da tabela de distribuição, isto acarreta em 2^{32} textos de entrada e 2^{32} textos de saída. Por exemplo, o intervalo de números seqüenciais que poderão ser considerados universo de entrada da Função F é igual a $[0;4294967295]$, ou seja, $[2^0;2^{32}]$, isto representa uma tabela de distribuição de $2^{32}x2^{32}$, ou seja, 2^{64} células, isso corresponde, no Papílio, uma tabela de distribuição para uma única caixa S.

Por se tratar de um número realmente elevado e para mostrar uma projeção da complexidade do Papílio, o processo de construção de características, probabilidades e da tabela de

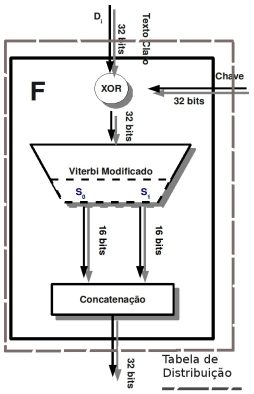

Figura 5.1: Estrutura da função F.
Fonte:[Ramos 2002].

distribuição, modificou-se o Papílio Original a aceitar a criptografia de textos com tamanhos de textos variáveis, sendo estes múltiplos de 8 bits e também o tamanho das sub chaves de modo adaptativo, permitindo a aplicação da Criptoanálise do Papílio para os seguintes casos apresentados a seguir.

5.3 Textos com 8 bits de Tamanho

Para essa situação o Papílio Original aceita textos que sejam múltiplos de 1 caractere, logo aceita qualquer entrada, e conseqüentemente textos binários que sejam múltiplos de 8 bits. Neste caso, o tamanho do bloco será de 8 bits, e portanto, o semi bloco de entrada será a entrada na função com tamanho de 4 bits. A geração de sub chaves, neste caso, foi reduzida ao tamanho do semi bloco, logo o tamanho de cada sub chave será de 4 bits.

Outra informação inerente a este caso, é que a tabela de distribuição pode ser considerada uma matriz quadrada [1], agora tem 2^4 possíveis XOR's de entrada e 2^4 XOR's de saída. Para construir a tabela foi utilizado um arquivo de texto gerado pelo algoritmo Papílio. Uma amostra deste arquivo está no Apêndice A.1, que contém informações sobre todas as entradas e saídas possíveis da função F, saída da função F, após a XOR com a sub chave e a saída do semi bloco do ciclo correspondente. Neste caso, a chave matriz produz as 16 sub chaves com valores iguais à zero. Com isso, a tabela e os números que originaram a tabela de distribuição estão em ordem crescente.

De posse deste arquivo de saída, utiliza-se o código fonte *tabXor.swp.lua* disponível no apêndice A.5 para gerar a tabela de distribuição e a lista de pares que compõem a respectiva tabela. O processo de construção da tabela é dada da seguinte maneira:

[1] Uma matriz é dita quadrada quando seu número de linhas é igual ao seu número de colunas

1. Calcula-se a XOR de dois textos de entradas;
2. Calcula-se a XOR de dois textos de saídas respectivas das entradas escolhidas;
3. Cada cabeçalho de linha representa o valor da XOR de entrada obtida;
4. Cada cabeçalho de coluna representa o valor da XOR de saída obtida;
5. Cada célula representa a quantidade de textos que originaram uma determinada XOR de entrada e XOR de saída.

A tabela referente ao texto de entrada igual a 8 bits de tamanho é mostrada na tabela 5.3. As células que contém como valor "*", denota que para uma determinada XOR de entrada e para uma determinada XOR de saída, não houveram pares a satisfazer aquela condição. Por exemplo, na tabela 5.3 para o caso do valor da XOR de entrada igual a 13, obteve-se 12 textos de entrada que levaram a uma saída XOR igual a 8. Logo, $13 \rightarrow 8$ tem uma probabilidade de $12/16$ de ocorrer. Outro fator é que a somatória dos valores de cada linha é sempre igual ao número de textos possíveis de entrada. Por exemplo, na 4.1 que representa a tabela distribuição, o somatório dos valores de cada célula de uma linha é igual a 64, pois esta é a quantidade de textos possíveis de entrada em cada caixa S. No Papílio de 8 bits e 16 bits, o perfil da tabela de distribuição do DES obedece o mesmo padrão, ou seja, o somatório de cada linha é igual ao número de possíveis valores de entrada.

Entrada XOR / Saída XOR	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	16	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
1	*	4	*	*	*	12	*	*	*	*	*	*	*	*	*	*
2	*	12	*	*	*	4	*	*	*	*	*	*	*	*	*	*
3	*	*	*	*	16	*	*	*	*	*	*	*	*	*	*	*
4	*	*	*	4	*	*	*	12	*	*	*	*	*	*	*	*
5	*	*	12	*	*	*	4	*	*	*	*	*	*	*	*	*
6	*	*	4	*	*	*	12	*	*	*	*	*	*	*	*	*
7	*	*	*	12	*	*	*	4	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*	*	12	*	*	*	4	*
9	*	*	*	*	*	*	*	*	*	*	*	4	*	*	*	12
10	*	*	*	*	*	*	*	*	*	*	*	12	*	*	*	4
11	*	*	*	*	*	*	*	*	*	*	4	*	*	*	12	*
12	*	*	*	*	*	*	*	*	*	4	*	*	*	12	*	*
13	*	*	*	*	*	*	*	*	12	*	*	*	4	*	*	*
14	*	*	*	*	*	*	*	*	4	*	*	*	12	*	*	*
15	*	*	*	*	*	*	*	*	*	12	*	*	*	4	*	*

Tabela 5.1: Tabela de distribuição de bloco igual à 8 bits

A lista de pares que satisfazem essa implicação ($13 \rightarrow 8$) está presente no arquivo gerado pelo programa em Lua, chamado tabXor.swp.lua presente no Apêndice A.5. Estes pares formam a tabela de distribuição para 8 bits de entrada. A lista de todos os pares de todas as implicações estão no Apêndice A.2. Porém, a lista de pares referente a $13 \rightarrow 8$ é apresentada na tabela 5.3. Nesta tabela, são apresentados os 6 pares que têm como entrada XOR o valor 13 e quais suas XOR's de saídas de suas respectivas cifras são iguais a 8. Para cada par tem-se dois textos (Texto 1 e Texto 2), para o par 1 até o par 6.

Pares	Texto 1	Texto 2
Par 1	1	12
Par 2	2	15
Par 3	4	9
Par 4	5	8
Par 5	6	11
Par 6	7	10

Tabela 5.2: Lista de pares pertencentes a $13 \rightarrow 8$

Outro fator a ser colocado é que todos os pares com XOR de entrada igual a zero, obrigatoriamente suas XOR de saída também serão iguais a zero. Isto ocorre porque os textos iguais sofrem influência da mesma sub chave em respectivo ciclo de forma a gerar sempre a mesma saída.

5.4 Textos com 16 bits de Tamanho

Agora o Papílio é alterado para receber e criptografar dados de entrada com tamanhos múltiplos de 2 caracteres ou que tenham o comprimento igual a 16 bits. Assim, seu semi bloco de entrada em cada ciclo será igual a 8 bits ou um caractere.

Como a quantidade é 8 bits, o maior número permitido como entrada na função F é 2^8, ou seja, a quantidade de possíveis valores de entrada na função está no intervalo de $[0;255]$, e como a saída da função também é igual a 8 bits, tem-se como maior valor possível igual a 256 também. As sub chaves, assim como na seção 5.3, serão também iguais a zero, somente quanto a elaboração da tabela de distribuição, voltando ao seus valores normais para posteriormente serem feitas as análises. Na tabela 5.3 a seguir, é vista apenas uma amostra da tabela gerada pela saída do Papílio referente aos blocos de 16 bits de tamanho. O arquivo necessário para elaborar esta tabela, é o de saída do algoritmo Papílio Original por conter uma grande quantidade de linhas, igual a 1024 linhas, não será possível disponibilizar no Apêndice deste trabalho em sua

totalidade. Mas, no Apêndice A.3 é apresentada uma amostra deste arquivo, onde é separado por registro de 4 linhas, onde a 1ª linha é o texto de entrada na função F apenas no primeiro ciclo, a 2ª linha é o semi bloco após a XOR com a sub chave, a 3ª linha é a sua respectiva saída do semi bloco e, por fim, a 4ª linha é o resultado da parte esquerda do semi bloco com a saída da função F.

A tabela de distribuição é construída, assim como na seção 5.3, aplicando sobre esses registros operações XOR, seguindo sempre o processo da criptoanálise diferencial quanto a construção desta tabela, assim como em [Biham e Shamir 1990] obtendo, desta forma as entradas e saídas XOR de cada par de entrada na função F. A tabela de distribuição com 16 bits de tamanho de bloco é apresentada a seguir, contudo apenas em pequena escala na tabela 5.3

X.E./ X.S	0	1	17	16	3	18	2	19	50	35	51	34	33	48	32	49	6	7	23	22	37	36	52
0	256																						
1		128	128																				
2		128	128																				
3				256																			
4					16	16	48	48	16	48	16	48											
5					48	16	48	16	16	16	48	48											
6					16	48	16	48	48	48	16	16											
7					48	48	16	16	48	16	48	16											
8					16	16	48	48	16	48	16	48											
9					48	16	48	16	16	16	48	48											
10					16	48	16	48	48	48	16	16											
11					48	48	16	16	48	16	48	16											
12													128			128							
13														64	192								
14														192	64								
15													128			128							
16																	12			4			
17																		12	4				
18																		4	12				
19																	4			12			
20																					8		
21																						16	
22																							16

Tabela 5.3: Tabela de distribuição de bloco igual à 16 bits

Os cabeçalhos das linhas significam os valores das XOR's de entrada enquanto os cabeçalhos de cada coluna representam os valores das XOR's de saída de cada par de texto de entrada. Nas células em branco, significa que não houve pares para aquelas entradas e saídas XOR's. Por exemplo, existem 16 textos que satisfazem $22 \rightarrow 52$, logo a probabilidade dessa implicação ocorrer é de $16/256$. Os textos que satisfazem todas as XOR's de entradas e XOR's de saídas compõem um arquivo composto por todos os pares de entrada e pares de saída, onde para cada XOR de entrada, independente do valor da XOR de saída, há 128 pares (256 textos) como pares de entrada e 256 textos como pares de saída, tornando assim praticamente impossível de apresentá-lo em sua totalidade, no entanto, uma amostra deste arquivo está no Apêndice A.4 (apresentando somente algumas XOR's de entrada e XOR's de saída). A tabela 5.4 apresenta os 16 textos que satisfazem $22 \rightarrow 52$.

Pares	Texto 1	Texto 2
Par 1	0	22
Par 2	1	23
Par 3	2	20
Par 4	3	21
Par 5	12	26
Par 6	13	27
Par 7	14	24
Par 8	15	25

Tabela 5.4: Lista de pares pertencentes a $22 \rightarrow 52$

Todas as probabilidades servem para montar as características de um grupo de ciclos, que será apresentado na próxima seção para ambas situações de tamanho de bloco de entrada. Todos as características apresentadas, juntamente com pares corretos e as probabilidade, seguirão o fluxo dos textos escolhidos no 1^{o} ciclo até o 16^{o} ciclo.

5.5 Características, Pares Corretos e Probabilidades

Nesta seção será apresentado, para cada ciclo do Papílio, as características de cada ciclo conforme as apresentadas por [Biham e Shamir 1990] na criptoanálise do DES. Todas as características serão compostas por probabilidades de XOR's de entradas implicarem em XOR's de saídas durante um determinado número de ciclos para dois possíveis tamanhos de bloco de entrada juntamente com os pares corretos.

5.5.1 Papílio 1° ao 16º Ciclo

No primeiro ciclo do Papílio pode se observar que textos com XOR's de entrada igual zero sempre acarretará em XOR's de saída igual a zero. A seguir, será apresentado para blocos de tamanho igual a 8 bits e posteriormente para blocos de 16 bits.

Textos com 8 Bits de Tamanho

A figura 5.2 apresenta essa situação, logo a probabilidade sempre será igual a 1.

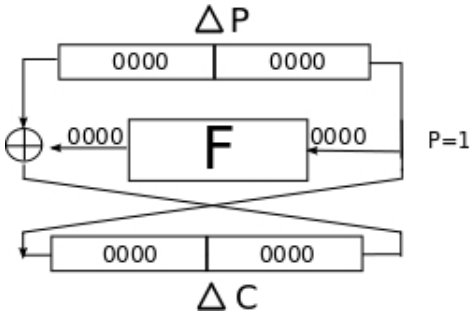

Figura 5.2: Característica do 1º ciclo para $\Delta_P = 00000000_2$

Um segundo exemplo para este ciclo é apresentado na figura 5.3, que apresenta para entrada 00110111 de tamanho igual a 8 bits com uma probabilidade igual a $12/16$ para a saída XOR igual a 0011.

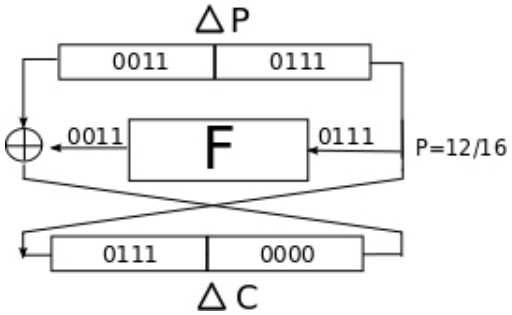

Figura 5.3: Característica do 1º ciclo para $\Delta_P = 00110111_2$

Observa-se que, para cada característica, diferentemente como utilizada para gerar a tabela de distribuição e as probabilidades, a característica é influenciada pelo semi bloco esquerdo da XOR de entrada, causando assim comportamentos alternados e que podem ser previstos durante os 16 ciclos do Papílio.

No exemplo para o 3º ciclo, a probabilidade da característica é obtida através do produto da probabilidade de cada ciclo, logo o produto será igual à $(12/16 * 1 * 12/16)$. Como mostra

a figura 5.4, a probabilidade resultante é igual a $144/256$ da característica da XOR de entrada igual a 00110111 para uma XOR de saída é igual 01110011.

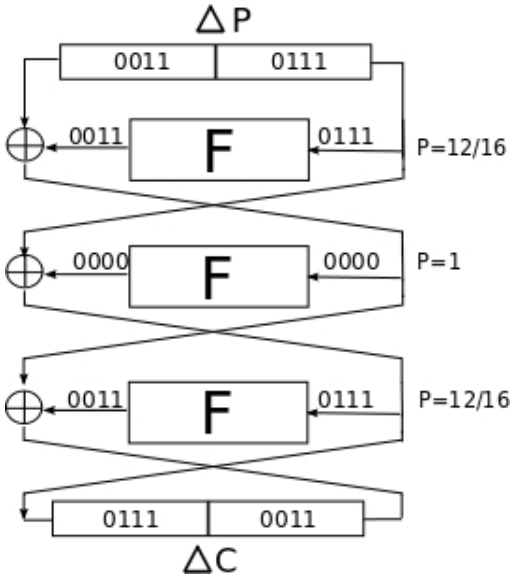

Figura 5.4: Característica do 3º ciclo para $\Delta_P = 00110111_2$

Note que esta característica tem a peculiaridade de sua XOR resultante dos valores de cada semi bloco (EC e DC) trocados. Isso demonstra que o algoritmo obedece um padrão lógico, o que permite concatenar características assim como é feito no DES. A figura 5.5 apresenta uma característica completa nos 16 ciclos presentes no Papílio para a mesma entrada XOR e para a XOR de saída igual a 00000111. O produto de todas as probabilidades durante os 16 ciclos de $\Delta_P = 00110111_2 \rightarrow \Delta_C = 00000111_2$ é dado pelo produto de $(12/16 * 2/16 * 2/16 * 2/16 * 2/16 * 2/16 * 2/16)$ que é igual a 35831808/268435456, ou seja, igual a $0,133483887$. Com essas características, e juntamente com a tabela de distribuição, é possível descobrir quais pares não pertencem a estas características, ganhando assim, tempo e precisão nas escolhas de prováveis sub chaves ou mesmo nas comparações futuras após a descoberta de cada sub chave.

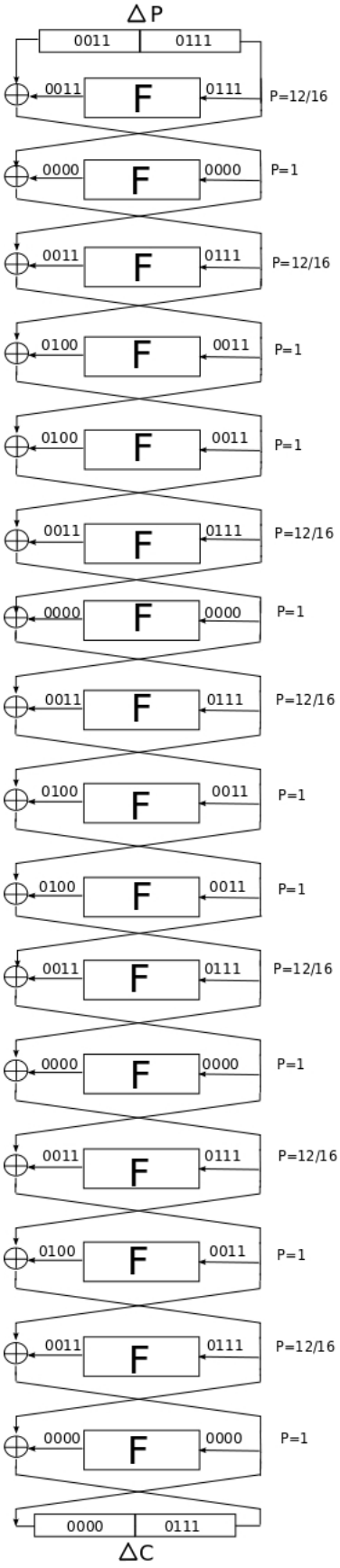

Figura 5.5: Característica do 16º ciclo para $\Delta_P = 00110111_2 \rightarrow \Delta_C = 00000111_2$

Textos com 16 Bits de Tamanho

A figura 5.6 apresenta essa situação, logo a probabilidade sempre será igual a 1.

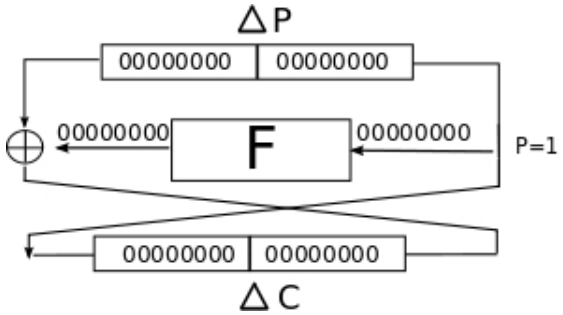

Figura 5.6: Característica do 1º ciclo para $\Delta_P = 0000000000000000_2$

Um segundo exemplo para este ciclo é ilustrado na figura 5.7, na qual é utilizada a entrada de 16 bits: 0000111101111100. Na figura 5.7 pode se observar que a saída da função F é 00001111,com uma probabilidade de $8/256$.

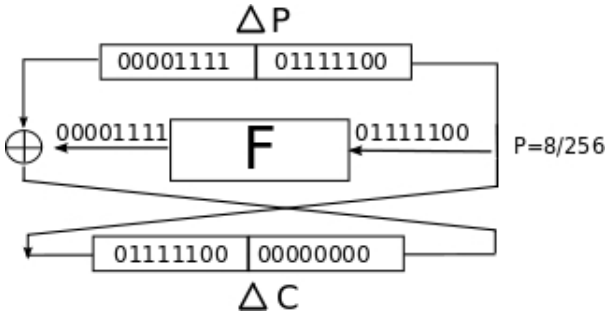

Figura 5.7: Característica do 1º ciclo para $\Delta_P = 0000111101111100_2$

Observa-se também, que para cada característica, independente de como é utilizada para gerar a tabela de distribuição e as probabilidades, a característica é influenciada pelo semi bloco esquerdo da XOR de entrada, causando assim comportamentos alternados e que podem ser previstos durante os 16 ciclos do Papílio. As figuras 5.8 e 5.9 apresentam as seqüências da característica escolhida até o 2º e 6º ciclo respectivamente, e através do comportamento da característica é possível analisar alguns fatores. O primeiro fator é sobre a saída do 2º ciclo, onde apresenta o semi bloco esquerdo do Δ_C igual a 0, indicando que os 8 textos pertencentes a $224 \rightarrow 15$ ao sofrerem a operação XOR com o semi bloco esquerdo de Δ_C sempre resultará em 124 que será usado com o semi bloco direito de entrada do 3º ciclo. Assim a probabilidade dessa característica para o 2º ciclo é igual a $8/256$.

Para a figura 5.9, que representa a característica reduzida ao 6º ciclo, tem-se que as probabilidade e XOR's de entradas e XOR's de saídas se alternam durante os ciclos. Isso significa que este par, que foi escolhido aleatoriamente para demonstrar esta característica, não

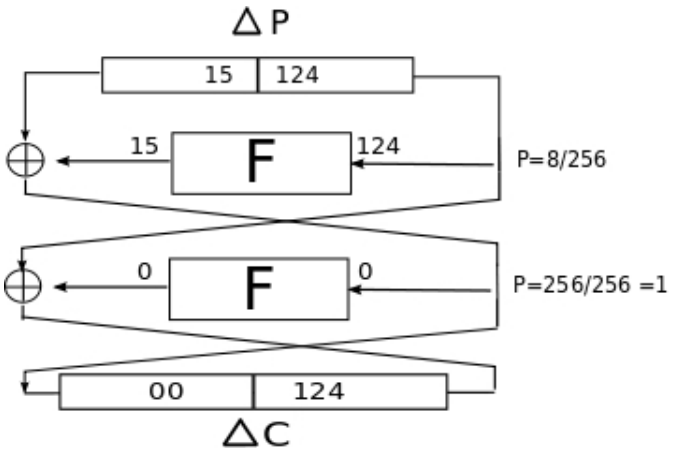

Figura 5.8: Característica do 2º ciclo para $\Delta_P = 0000111101111100_2$

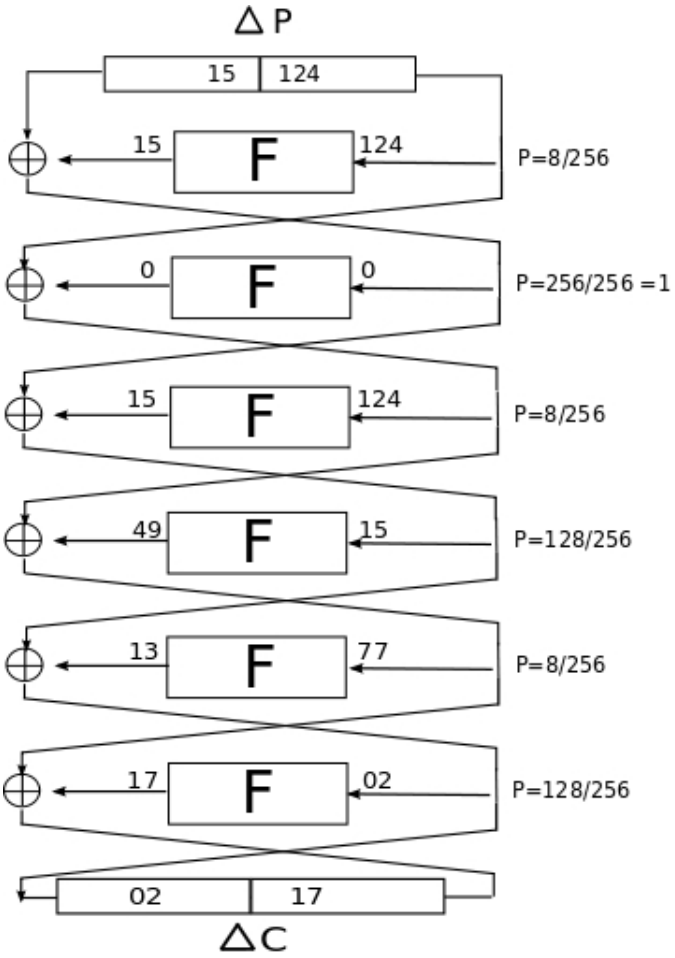

Figura 5.9: Característica do 6º ciclo para $\Delta_P = 0000111101111100_2$

obedece padrões conhecidos, o que o torna uma característica fraca. No entanto, a busca da melhor característica é possível através da escolha de outros pares e através das escolhas de suas probabilidades, atingindo todas as probabilidades possíveis. Para a característica reduzida ao 6^o ciclo, a probabilidade dessa característica é igual ao produto de $8/256 * 1 * 8/256 * 128/256 * 8/256 * 128/256$, ou seja, igual a $8388608/281474976710656$ ou igual a $3x10^{-9}$. Uma observação importante é o fato de que mesmo que uma característica tenha uma probabilidade alta de conter, não significa necessariamente que ela seja uma característica forte, contudo é uma boa característica, porém só será considerada uma característica forte quando a mesma tiver além de probabilidade alta, uma seqüência de padrões que auxiliem na busca de pares corretos que serão usados na busca pelas sub chaves. A figura 5.10 apresenta a característica completa nos 16 ciclos para o $\Delta_P = 0000111101111100_2$. A probabilidade, conforme a figura 5.10, é $1/4003199668773774222222222x10^{-9}$. Essa probabilidade, ao longo dos 16 ciclos do Papílio, é útil para elaborar tabelas de distribuição tratando todo o algoritmo como uma Função F, guardando e utilizando os pares que satisfazem aquela probabilidade de forma a garantir a quebra das sub chaves.

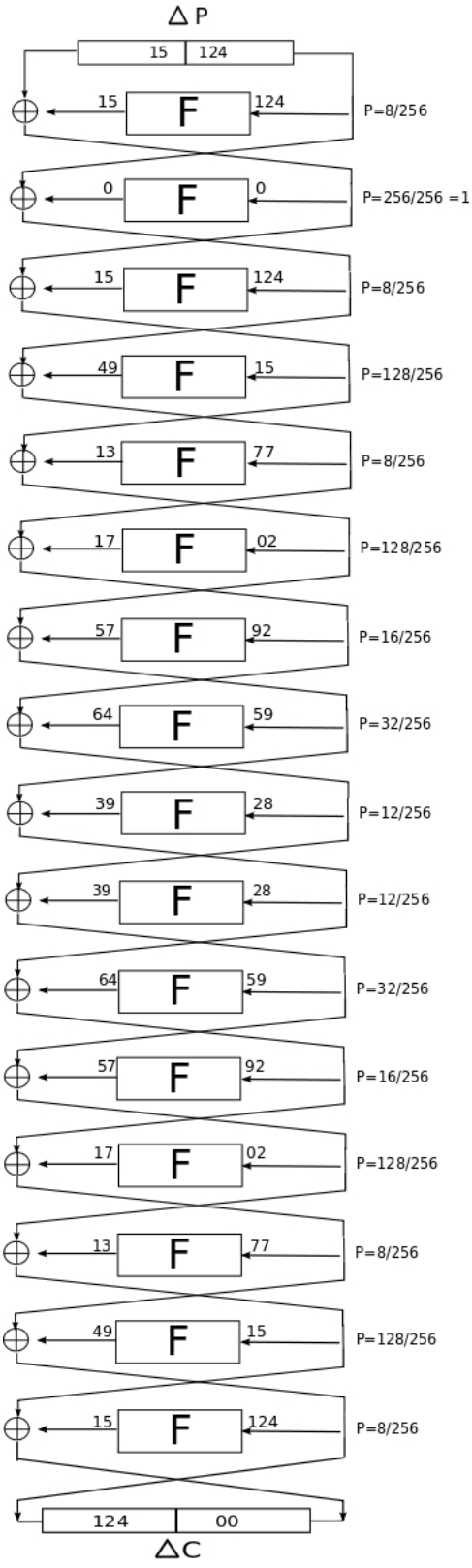

Figura 5.10: Característica do 16º ciclo para $\Delta_P = 0000111101111100_2$

5.6 Quebra de Sub Chaves do Papílio

A possibilidade de descobrir qual é a sub chave utilizada pelo algoritmo em um determinado ciclo é obtida através das características, pares corretos e das probabilidades resultantes do processo visto na seção anterior. Supondo que se queira quebrar a cifra do Papílio sem utilizar a técnica de criptoanálise diferencial, o outro método de quebra, é a busca exaustiva, nesta técnica onde é necessário testar todos os possíveis valores para cada sub chave e comparar seus valores resultantes para verificar se foi descoberta ou não a sub chave do algoritmo.

Neste caso, para as duas situações de tamanho de bloco (8 e 16 bits) e respectivos semi blocos (4 e 8 bits), os intervalos $[0..15]$ e $[0..255]$ dos valores das sub chaves possíveis de entrada na função F, são conjuntos de possíveis valores dos quais as sub chaves dos ciclos podem assumir. Como o tamanho do bloco do Papílio Original é igual a 64 bits e conseqüentemente o tamanho do semi bloco é igual 32, logo o tamanho da sub chave também é igual 32, o que acarreta no intervalo de valores para cada sub chave igual a $[0..4294967295]$. Como esses intervalos são apenas para cada ciclo (cada sub chave) isso torna a quebra da sub chave ainda mais complexa, pelo fato de acertar o valor de 16 sub chaves de forma seqüencial ordenada. Por exemplo, para o conjunto de sub chaves $[SK_1, SK_2, ..., SK_{15}, SK_{16}]$, como cada sub chave assumirá valores que obrigatoriamente estarão em ordem pré estabelecida, e pelo princípio fundamental da contagem[2], isso acarretará em uma combinação de sub chaves, logo sua complexidade pode ser obtida pela fórmula:

$$C_K = V^{16},$$

onde C_K representa a complexidade das sub chaves e V representa o valor máximo permitido pela sub chave.

Por exemplo, para o tamanho do semi bloco igual a 8 bits, a complexidade de todas as sub chaves é igual a $C_K = 8^{16} = 281474976710656$, isto porque o tamanho de cada sub chave é 8 bits. Já para o tamanho do semi bloco igual a 16 bits, a complexidade de todas as sub chaves é igual a $C_K = 16^{16} = 18446744073709551616$, isto porque o tamanho de cada sub chave é 16 bits.

Para os dois casos apresentados neste trabalho, a busca por força bruta (exaustiva) é aplicável, pelo fato do Papílio 8 e 16 bits ter um universo pequeno de possíveis sub chaves comparado ao Papílio 32 e 64 bits, ou seja, para as condições normais do Papílio que cada sub chave tem o

[2] Se determinado acontecimento ocorre em n etapas diferentes, e se a primeira etapa pode ocorrer de K_1 maneiras diferentes, a segunda de K_2 maneiras diferentes, e assim sucessivamente, então o número total T de maneiras de ocorrer o acontecimento é dado por: $T = K_1.K_2.K_3.....K_n$.

tamanho igual a 32 bits, essa complexidade é igual $C_K = 1208925819614629174706176$. Com a Criptoanálise Diferencial é possível reduzir o intervalo de valores possíveis para cada sub chave, reduzindo em até 2 possibilidades de sub chaves para cada ciclo. O processo de busca e de descoberta de sub chaves, que é aplicado em textos com 8 bits e 16 bits de tamanho, é descrito abaixo:

- 1º Passo: Primeiramente supõe-se dois valores de entrada, que tenham tamanhos compatíveis às duas possibilidades de tamanho de texto (8 e 16 bits).
- 2º Passo: Calcula-se as suas respectivas saídas do algoritmo. Neste caso, como a intenção é descobrir a sub chave para cada ciclo isoladamente, essa saída criptografa equivale à saída da função F (Viterbi Modificado).
- 3º Passo: Calcula-se a XOR dos valores de entrada e a XOR dos valores de suas respectivas cifras.
- 4º Passo: Em posse da tabela de distribuição, desenvolvida na seção 5.2, e da lista de pares originada pelo programa *tab.XOR.lua* disponível no Apêndice A.5, localiza-se quais são os valores que implicam nos valores de XOR de entrada e XOR de saída calculada no passo anterior.
- 5º Passo: Aplica-se nestes pares a operação XOR com um dos valores de entrada. Os resultados são armazenados em uma tabela e cada um desses valores resultantes dessa operação é considerado um provável valor da sub chave.
- 6º Passo: Os 5 primeiros passos devem ser repetidos até que algum valor se repita. Para que isso ocorra, é necessário informar a cada início dois valores novos de entrada.

Estes passos foram implementados no programa *Analisador.lua*, disponível no Apêndice A.6. Ele gera como saída os valores de sub chave mais repetidos, indicando o intervalo de valores de sub chaves. Lembrando que a técnica de criptoanálise diferencial é utilizada para diminuir o esforço necessário na quebra de uma chave, ou seja, a mesma utilizada com o métodos da tentativa tem um grau de melhora considerado em relação ao método de busca exaustiva realizado separadamente. A seguir são apresentadas duas tabelas que informam quais os valores encontrados como possíveis sub chaves para cada ciclo.

5.6.1 Textos com 8 Bits de Tamanho

A tabela 5.5 apresenta os valores de possíveis sub chaves para cada ciclo. O número de possíveis valores para cada sub chave reduziu de 16 para 2, uma redução de $87,5\%$ em relação ao método de força bruta. No entanto para chegar a esses valores uma quantidade de pares e séries de cálculos foram realizados. A complexidade do algoritmo nessas condições é calculada pelo número de operações necessárias para encontrar dois valores pela a quantidade de textos utilizados nessa varredura de informações. A questão de apenas 2 números como possíveis sub chaves está vinculada a mudança de apenas dois bits menos significativos, por exemplo, na tabela 5.5 no ciclo 11 os valores $[08][11]$ diferenciam apenas dois bits, visto que em binário, os números $[08][11]$ são iguais a $[0100][0111]$, ou seja, os valores dos dois bits menos significativos são alterados de forma a complementar um ao outro. Logo o número de bits encontrado em cada ciclo é igual $N_{Bits} = Tamanho\_Bloco - 2$. Esta característica do Papílio é satisfeita em todos os ciclos, tanto para tabela 5.5 como para a tabela 5.6 .

Observe como muitas das sub chaves encontradas se repetem como nos casos do ciclo 1, ciclo 6 e ciclo 8, que sugerem como possíveis sub chaves os valores 05 e 06. Isso decorre porque o intervalo é considerado pequeno e possui apenas 16 valores possíveis. A partir deste ponto, para definir de fato qual é a sub chave utilizada, é necessário apenas informar ao algoritmo os dois valores e comparar suas saídas, então a sub chave que levar o valor correspondente será a certa.

5.6.2 Textos com 16 Bits de Tamanho

No caso de blocos com 16 bits de tamanho, o resultado é ainda mais surpreendente. Pois a quantidade de sub chaves possíveis se reduz de 256 valores para apenas 2 valores. Neste caso, a redução foi de aproximadamente $99,22\%$ da quantidade de possíveis sub chaves, em relação à quantidade original necessária para blocos com este tamanho de entrada. Observe que ainda há algumas repetições, porém em pequena escala, visto que o intervalo de sub chaves é consideravelmente superior ao bloco de 8 bits. Fazendo-se o mesmo trabalho aplicado como na subseção anterior, apenas é necessário aplicar as duas sub chaves ao algoritmo e ao ciclo correspondente e comparar os resultados e a sub chave que corresponder ao valor certo será a sub chave correta.

A complexidade da criptoanálise diferencial sobre o Papílio de 8 e 16 bits é calculada através dos textos escolhidos (textos que formam a tabela de distribuição) pelo algoritmo Analisador.lua. O código fonte do Analisador.lua está disponível no Apêndice A.6. Após a escolha

TAMANHO DE BLOCO IGUAL A 8 BITS	
Número do Ciclo	Valores Propostos
Ciclo 1	[09] [10]
Ciclo 2	[09] [10]
Ciclo 3	[04] [07]
Ciclo 4	[05] [06]
Ciclo 5	[09] [10]
Ciclo 6	[05] [06]
Ciclo 7	[00] [03]
Ciclo 8	[05] [06]
Ciclo 9	[13] [14]
Ciclo 10	[01] [02]
Ciclo 11	[08] [11]
Ciclo 12	[08] [11]
Ciclo 13	[13] [14]
Ciclo 14	[01] [02]
Ciclo 15	[08] [11]
Ciclo 16	[05] [06]

Tabela 5.5: Tabela de valores de sub chave com 8 bits de tamanho

dos textos, é aplicado sobre eles as operações necessárias para encontrar os valores das sub chaves mais prováveis, cada texto que foi utilizado por uma operação é denotado como texto Analisado. Como há de fato um trabalho para encontrar os textos escolhidos e este esforço computacional, é de extrema importância para o início da criptoanálise diferencial, a complexidade será equivalente ao número de textos utilizados para que serão posteriormente analisados. Os resultados e informações mais detalhadas sobre os dois tipos de blocos de entrada, seus textos e as quantidades necessárias para cada par de sub chave encontrada será vista na seção seguinte.

5.7 Resultados

Nesta seção serão apresentados os resultados significativos do Papílio restritamente às duas opções de tamanho de entrada de texto (8 e 16 bits). As quantidades de operações necessárias, de pares utilizados assim como a complexidade requerida para cada sub chave de cada ciclo de 8 e 16 bits de bloco serão exibidas nas tabelas 5.7 e 5.8, respectivamente.

As tabelas 5.7 e 5.8 representam, em detalhes, quantos textos foram escolhidose quais destes textos foram analisados pelo Analisador.lua e quais as sub chaves propostas por ele. Os valores são resultantes do programa Analisador.lua.

A coluna T. Escolhidos (Textos Escolhidos) exibe o número de textos necessários para

TAMANHO DE BLOCO IGUAL A 16 BITS	
Numero do Ciclo	Valores Propostos
Ciclo 1	[160] [163]
Ciclo 2	[029] [030]
Ciclo 3	[003] [000]
Ciclo 4	[096] [099]
Ciclo 5	[168] [171]
Ciclo 6	[016] [019]
Ciclo 7	[192] [195]
Ciclo 8	[032] [035]
Ciclo 9	[161] [162]
Ciclo 10	[018] [017]
Ciclo 11	[193] [194]
Ciclo 12	[032] [035]
Ciclo 13	[054] [053]
Ciclo 14	[081] [082]
Ciclo 15	[133] [134]
Ciclo 16	[125] [126]

Tabela 5.6: Tabela de valores de sub chave de 16 bits de tamanho

RESUMO DA CRIPTOANÁLISE COM TAMANHO DO BLOCO IGUAL A 8 BITS				
Ciclo	**Val. Propostos**	**T. Escolh.**	**T. Analis.**	**Complexidade**
1º	[09] [10]	124	88	124
2º	[09] [10]	160	62	160
3º	[04] [07]	328	96	328
4º	[05] [06]	364	64	364
5º	[09] [10]	180	48	180
6º	[05] [06]	1076	156	1076
7º	[00] [03]	584	216	584
8º	[05] [06]	234	44	234
9º	[13] [14]	272	12	272
10º	[01] [02]	930	460	930
11º	[08] [11]	474	120	474
12º	[08] [11]	1020	112	1020
13º	[13] [14]	216	108	216
14º	[01] [02]	234	52	234
15º	[08] [11]	346	64	246
16º	[05] [06]	216	116	216

Tabela 5.7: Resumo da criptoanálise diferencial aplicada ao Papílio com 8 bits de entrada

RESUMO DA CRIPTOANÁLISE COM TAMANHO DO BLOCO IGUAL A 16 BITS				
Ciclo	**Val. Propostos**	**T. Escolh.**	**T. Analis.**	**Complexidade**
1º	[160] [163]	1548	224	1548
2º	[029] [030]	2322	108	2322
3º	[003] [000]	19608	124	19608
4º	[096] [099]	4128	36	4128
5º	[168] [171]	6708	384	6708
6º	[016] [019]	42054	640	42054
7º	[192] [195]	13416	336	13416
8º	[032] [035]	19608	568	19608
9º	[161] [162]	26316	316	26316
10º	[018] [017]	2580	116	2580
11º	[193] [194]	7224	100	7224
12º	[032] [035]	7740	84	7740
13º	[054] [053]	3870	56	3870
14º	[081] [082]	4644	160	4644
15º	[133] [134]	4644	20	4644
16º	[125] [126]	5676	264	5676

Tabela 5.8: Resumo da criptoanálise diferencial aplicada ao Papílio com 16 Bits de Entrada

construção da tabela de distribuição para 8 e 16 bits, para cada entrada XOR e saída XOR necessários, para encontrar as duas prováveis sub chaves. Conseqüentemente a coluna T. Analis. (Textos Analisados) faz referência aos textos utilizados pelo programa Analisador.lua, para todos os ciclos do algoritmo, onde, a partir dessas análises, resultaram nas duas sub chaves possíveis que foram utilizadas pelo algoritmo Papílio durante a criptografia dos dados.

A coluna T. Analisados também é equivalente ao números de operações, que foram necessárias para chegar encontrar os pares de sub chaves. Exemplos dos pares que foram utilizados para cada quebra, estão nos Apêndices A.4 e A.2. Por último, a complexidade pode ser medida de diversas formas, mas a escolhida significa o número de textos utilizados para conseguir as prováveis sub chaves, ou seja, que foram utilizados na construção de cada tabela de distribuição, pelo fato, da necessidade de um esforço computacional para construção e armazenamentos de todos os pares para cada entrada e saída XOR da tabela. Estas observações são verificadas nas tabelas 5.7 e 5.8.

Tanto para tabela 5.7 como para a tabela 5.8, a coluna de V. Propostos (Valores Propostos) indica quais foram as sub chaves do algoritmo indicadas resultantes para aquele ciclo, visto que nas duas tabelas sempre é o caso de dois valores de sub chave:

5.8 Projeção da Criptoanálise Diferencial ao Papílio Original com 64 Bits de Bloco

Como observado na seção 5.2, a tabela de distribuição para o Papílio 64 bits, é 2^{32} (apenas a metade do bloco entra em cada função F), a qual é praticamente impossível de ser armazenada, isto ocorre pelo fato da representação de cada número em binário, ter exatamente 32 bits por cadeia, como no programa tabXor.swp.lua, cada bit é um caractere, ou seja, cada bits 0 ou 1, requer 8 bits de espaço. Énecessário uma grande quantidade de espaço para armazenar a tabela de distribuição com todos os pares correspondentes para cada entrada e saída XOR, ou seja, a tabela de distribuição, excluindo a lista de pares, para o Papílio 64 bits terá $2^{32}.2^{3}$ de tamanho em bits.

Mesmo para o Papílio de 32 bits de tamanho, a tabela seria de $2^{16}.2^{3}$, o que para os recursos computacionais disponíveis, ainda é inviável. Os recursos compuacionais da máquina utilizada estão descritas na seção 5.8. Pela impossibilidade de executar a criptoanálise sobre o modo de 64 bits de bloco de entrada do Papílio, uma outra vertente é a projeção dos blocos 8 e 16 bits para 32 e 64 bits, ou seja, pode-se fazer uma projeção baseados nos testes, cálculos, operações, escolha de pares, rodadas e complexidade feitos para o Papílio com blocos de 8 e 16 bits para determinar a complexidade de se quebrar o Papílio original. A tabela 5.9 apresenta, de acordo com os resultados obtidos para 8 e 16 bits de entrada, a razão obtida entre os dados das tabelas 5.7 e 5.8, projeção de possíveis valores para o Papílio original.

Ciclo	T. Esc. 8B	T. Esc. 16B	Razão. 16b/8b	T. Analis. 8B	T. Analis. 16B	Razão. 16b/8b	Complexidade
1º	124	1548	**12.48**	88	224	**2.55**	**12.48**
2º	160	2322	**14.51**	62	108	**1.74**	**14.51**
3º	328	19608	**59.78**	96	124	**1.29**	**58.78**
4º	364	4128	**11.34**	64	36	**0.56**	**11.34**
5º	180	6708	**37.27**	48	384	**8**	**37.27**
6º	1076	42054	**39.08**	156	640	**4.1**	**39.08**
7º	584	13416	**22.97**	216	336	**1.56**	**22.97**
8º	234	19608	**83.79**	44	568	**12.91**	**83.79**
9º	272	26316	**96.75**	12	316	**26.33**	**96.75**
10º	930	2580	**62.77**	460	116	**0.25**	**62.77**
11º	474	7224	**15.24**	120	100	**0.83**	**15.24**
12º	1020	7740	**7.59**	112	84	**0.75**	**7.59**
13º	216	3870	**17.92**	108	56	**0.52**	**17.92**
14º	234	4644	**19.85**	52	160	**3.08**	**19.85**
15º	346	4644	**13.42**	64	20	**0.31**	**13.42**
16º	216	5676	**26.28**	116	264	**2.28**	**26.28**

Tabela 5.9: Razão entre a tabela 5.7 e a tabela 5.8

Os valores positivos demonstram que, para o caso de 16 bits, houve um aumento nos números de pares escolhidos em relação à situação de 8 bits de bloco de entrada. Por exemplo, para o 10^{o} ciclo houve um aumento de 62.77 vezes no números de textos escolhidos para encontrar duas sub chaves. No entanto, houve uma queda no números de textos (T. Analis.) necessários, porém para tamanhos de entradas diferentes, onde esses valores não se tratam das mesmas sub chaves ou muito menos o tamanho delas são iguais, a quantidade de textos escolhidos refere-se somente aos cálculos necessários para obter tais sub chaves. Observa-se que sempre há um aumento no número de textos escolhidos, isso pelo fato do tamanho da sub chave de 8 bits passar a ter 16 bits, implicando no aumento da quantidade de valores para cada XOR de entrada e XOR de saída.

Também observa-se na tabela, que há de fato variações imprevistas como por exemplo, no 15^{o} ciclo que houve um acentuado aumento no número de textos escolhidos e uma drástica queda na quantidade de operações feitas (textos analisados) de 64 para 20 pares.

Na coluna de complexidade na tabela 5.9, demonstra-se a razão de crescimento da complexidade de 8 bits para 16 bits de entrada.

Isso demonstra que as informações são em certos casos (ciclos) impossíveis de serem projetadas, pois as mesmas dependem da tabela de distribuição e de suas probabilidades que afetam diretamente o curso das operações e a quantidade de pares em cada entrada e saída XOR da tabela de distribuição. Porém, a única métrica que é possível prever no Papílio original é a sua complexidade que certamente acarretará na utilização da razão do crescimento entre dos textos utilizados no mesmo ciclo (Papílio 8 bits), respectivamente ao mesmo ciclo para o Papílio 16 bits, para construir para cada ciclo, uma projeção de textos utilizados para o Papílio 32 e 64 bits de bloco de entrada.

O fato de não ter sido possível obter os reais valores resultantes da criptoanálise diferencial ao Papílio, se deu por não haver recursos computacionais de processamento e principalmente de armazenamento, visto que cada arquivo gerado, seja a tabela de distribuição ou a lista de pares resultante desta tabela, para tamanho de bits padrões do Papílio, não seria suportado nas máquinas disponíveis para a realização do trabalho. A máquina com a maior capacidade de recursos, estava disponível com a seguinte configuração: 8 processadores, 16 GB de memória ram, 500 GB de HD. A quantidade de recursos necessários para garantir a criptoanálise completa sobre o Papílio Original seria em torno de 32 GB de memória Ram, sendo assim as variáveis do programa *tabXOR.swp.lua* poderiam armazenar grandes quantidades de informações, para realização dos cálculos, evitando o caso de escrever / liberar informações em tempo de execução o que geraria um tempo ainda maior para realização dos cálculos ne-

cessários. Quanto ao processamento e ao número de processadores disponíveis, podem ser desconsiderados, pelo fato das operações serem seriais, ou seja, a instrução posterior depende do resultado da instrução anterior, contudo quanto maior for o ”clock”do processador, mais rápidos as instruções serão processadas. Já no quesito de armazenamento, trabalha-se com seqüências binárias, que são interpretadas pelo algoritmo como cada bit ”1”ou ”0”como sendo um caractere e com cada caractere tendo valor de 8 bits. O armazenamento para o Papílio 64 bits, que tem 32 bits de semi bloco, tem como possíveis valores de sub chaves o intervalo $[2^0; 2^{32}]$ e como cada elemento é representado na forma de 32 bits, com por exemplo, $2^0 = 00000000000000000000000000000000$, é necessário um alto poder de armazenamento para representar o intervalo utilizado.

Contudo, com o aperfeiçoamento do algoritmo, dos modos de armazenamentos (registros) e quais as informações que realmente devem ser armazenadas, possa-se ter os valores como resultados esperado para o Papílio.

PROJEÇÃO PARA 64 BITS DE TAMANHO DE ENTRADA					
Ciclo	**T. Esc. 8B**	**T. Esc. 16B**	**Razão. 16b/8b**	**Prev. 32 b**	**Prev. 64 b**
1º	124	1548	**12.48**	19325	241251
2º	160	2322	**14.51**	33698	489043
3º	328	19608	**59.78**	1172176	70073241
4º	364	4128	**11.34**	746814	530904
5º	180	6708	**37.27**	249985	9316100
6º	1076	42054	**39.08**	1643624	64238795
7º	584	13416	**22.97**	308200	7080166
8º	234	19608	**83.79**	1643050	137679151
9º	272	26316	**96.75**	2546073	246332563
10º	930	2580	**2.77**	7157	19856
11º	474	7224	**15.24**	110097	1677940
12º	1020	7740	**7.59**	58733	445679
13º	216	3870	**17.92**	69338	1242297
14º	234	4644	**19.85**	92166	1829131
15º	346	4644	**13.42**	62332	836613
16º	216	5676	**26.28**	149153	3919401

Tabela 5.10: Projeção para o Papílio 32 e 64 bits.

Entretanto, os testes e resultados apresentados servem ao propósito deste trabalho, visto que a estrutura do algoritmo não muda se for alterado o tamanho do bloco de entrada (bloco de 8 e 16 bits), sendo assim possível traçar o perfil do algoritmo. Isto demonstra que o algoritmo é passível da criptoanálise diferencial, ou seja, seu funcionamento permite analisar e obter resultados significativos mesmo sem ser aplicado no seu comportamento natural, conseguindo

assim, ter informações sobre os ciclos, os comportamentos das sub chaves, as saídas da função F e as características de seu comportamento. Quanto à questão do número de textos escolhidos é possível fazer uma projeção. Neste fator, como pode ser visto na tabela 5.9 há sempre um aumento no número dos pares, se for considerada a razão apresentada nesta tabela como sendo uma constante, ou mesmo uma função linear.

O fator de linearidade aplicada na projeção traz ótimos resultados para os blocos 32 e 64 bits, isto baseado na vertente de enquadrar a projeção como uma função não linear, ou seja, pode ser exponencial. Em outras palavras, apesar de abordar de forma linear a projeção dos resultados e mesmo assim obter um considerável resultado, é possível melhorar ou mesmo provar que esses resultados podem ser ainda melhor. Isto é possível pelo fato do tamanho de entrada crescer de forma exponencial, ou seja, para 4 bits de bloco de entrada, o intervalo é $[0..15]$, para 8 bits de bloco, o intervalo é $[0..255]$ e conseqüentemente para 16 bits de bloco, intervalo é $[0..65535]$ textos possíveis de entrada. Portanto, como tem-se um aspecto exponencial sobre o crescimento da entrada, este aspecto influencia nos resultados e conseqüentemente nas projeções para 32 e 64 bits de bloco de entrada. A quantidade prevista de textos utilizados para cada ciclo (linear), baseadas na projeção de $8 \rightarrow 16$, e implicando em $16 \rightarrow 32$ e posteriormente em $32 \rightarrow 64$, onde 64 bits é o padrão aceito pelo Papílio original, pode ser vista na tabela 5.10. Um fator importante na projeção é que o número de textos necessários para encontrar a sub chave é maior do que a força bruta em alguns ciclos. Por exemplo, para o Papílio de 8 bits, é necessário para a força bruta para o 1^{o} ciclo de forma independente[3], o total de 2^4 tentativas e para cada tentativa um par de textos (1 texto claro e 1 texto cifrado), logo um total de $2^4.2 = 32$ textos, o que torna todo ciclo com mais de 32 textos necessários para encontrar uma sub chave, mais complexo do que a força bruta. O mesmo raciocínio é aplicado para ciclos pertencentes ao Papílio 16, 32 e 64 bits de tamanho de entrada, para respectivos limites inferiores de textos escolhidos, os limites são iguais a $2^9, 2^{17}$ e 2^{33} textos utilizados pela força bruta para cada ciclo.

[3]Onde a sub chave de cada ciclo independente das outras sub chaves ou de uma sub chave geradora

6 Conclusão

A nível de comparação entre os demais algoritmos citados neste trabalho, serão apresentadas as principais diferenças entre eles, admitindo como ponto de referência os fatores de complexidade, número de textos utilizados e sub chaves encontradas. Estas comparações estão descritas neste capítulo, são elas:

- Primeiramente, o Papílio tem uma grande vantagem em relação a sua função F. Comparado às demais cifras permite que, mesmo alterando o tamanho da entrada do algoritmo (bloco de entrada), a função F adapte-se a qualquer entrada, isso pelo fato do Viterbi Modificado, ser acima de tudo, uma máquina de estado, não tendo, ao contrário do DES, o valor fixo para entrada e saída das caixas S. Isto permite que o algoritmo em si, seja adaptado mais rapidamente à situação em que ele deve atuar, ou seja, ele pode ser aplicado tanto na criptografia de grandes quantidades de dados como em pequenas quantidade, onde a rapidez na criptografia dos dados é realmente necessária.

- Ao projetar o Papílio para 1024 bits de entrada e com tamanho da chave também igual a 512, é praticamente impossível aplicar a criptoanálise diferencial sobre o mesmo. Isto é observado pelo fato do intervalo possível de valores passa de $[0..18446744073709551615]$ para $[0..2^{512}]$, ou seja, exigindo ainda mais poder de armazenamento e processamento dos computadores atuais, dificultando ainda mais a quebra das sub chaves.

A tabela 6.1 apresenta o resumo das principais métricas que servem de comparação entre às cifras apresentadas neste trabalho. A primeira delas é a quantidade de textos escolhidos para análise. Neste fator, como pode ser visto na tabela 5.9 o Papílio 64 necessita de uma grande quantidade de textos em cada ciclo para realizar a análise. Vale lembrar que o número de textos escolhidos obtém-se através de métodos independentes, não acarretando obrigatoriamente o aumento proporcional de textos ao número de ciclos. Na tabela 6.1, compara-se apenas os números referentes ao Papílio 8 e 16 bits, isso porque apenas nestes dois tipos de entradas, forma aplicadas de fato a técnica da criptoanálise diferencial. Se fosse possível adpatar as cifras DES, FEAL e BlowFish para os mesmos tamanhos de entradas do Papílio 8 e 16 bits, seria de

fato possível a comparação.

Quanto ao número de textos escolhidos (Papílio 8 e 16 bits), todas as cifras requerem mais textos do que ele. Quanto a quantidade de textos analisados, pode-se observar um considerável ganho de complexidade no Papílio, mesmo utilizando entradas menores do que 50% das entradas de FEAL e DES. Por exemplo, no 14º ciclo, o Papílio 16 bits exige que pelo menos 160 textos sejam analisados, enquanto o DES exige pelo menos 8 textos para encontrar as sub chaves utilizadas naquele ciclo.

Em relação aos dados reais (8 e 16 bits de entrada) o Papílio se torna muito fraco em relação aos demais algoritmos. A relação do Papílio com as demais cifras citadas neste trabalho, é diferenciada nos métodos utilizados e cálculos utilizados nas outras cifras. Por exemplo, o resultado da criptoanálise diferencial do BlowFish foi aplicado sobre apenas uma projeção de entradas de bits do BlowFish, enquanto no DES e na cifra FEAL, não foi necessário fazer tal adaptação. Porém, com os dados da projeção, em relação ao números de textos analisados o Papílio leva vantagem em relação as cifras em alguns ciclos e desvantagens em relação a outros. Por exemplo, observando a tabela 5.10, no 16º ciclo o Papílio leva cerca de 56.769 textos para descobrir as sub chaves, enquanto no DES por exemplo, precisa de $2^6 = 64$ textos para descobrir sub chaves de tamanho maior. Em relação a outras cifras, tanto nos números de textos escolhidos como na quantidade necessária de textos para serem analisados, o Papílio leva desvantagens, isso por que as outras cifras, apesar de serem tratadas com mais facilidade que o Papílio, possuem mais de uma caixa S em suas funções F, neste caso, esse fator é diretamente proporcional à complexidade, visto que quanto maior o número de caixas S, maior será o número de tabelas de distribuição, maior será o número de possibilidades das características e assim aumentando sempre a quantidade de textos a serem utilizados para a análise.

Sobre a complexidade do Papílio em relação às outras complexidades, ela é calculada de forma, já definida anteriormente, e diferente das demais, porém, para textos de entrada quase 50% menores que as entradas de textos das demais cifras, o Papílio demonstra ter uma complexidade alta para o tamanho de entradas testados, o que qualifica o Papílio, mesmo com um bloco de entrada pequeno, como uma cifra forte.

Por fim, apresenta-se algumas mudanças necessárias ao Papílio, para que o mesmo se torne ainda melhor em relação as demais cifras. São elas:

- Alterar o tamanho do bloco de entrada de 64 bits para 1024 ou superior, possibilitando que uma grande quantidade de texto seja criptografada mais rapidamente, como também aumentar exponencialmente o tamanho da tabela de distribuição utilizada pela técnica de

criptoanálise diferencial, dificultando a quebra das sub chaves.

- Alterar o tamanho da chave matriz de 128 bits para o mesmo tamanho do semi bloco, ou seja, com esse tamanho a capacidade de gerar sub chave de tamanho variáveis de 0 a até a metade do bloco de entrada e será dada de forma menos previsível, dificultando mais a quebra do algoritmo.

- Modificar a função F, de modo a alterar o estado de cada ciclo do Viterbi Modificado e inserir essa informação no texto cifrado. Com isso aumenta a complexidade de elaboração das tabelas de distribuição. Além de modificar a função F, alterar a quantidade da tabela do Viterbi dentro de cada função.

- Aumentar o número de ciclos, de forma variada, garantindo assim que o atacante não saberá em quantos ciclos o texto cifrado foi codificado.

Essas modificações já foram implementadas na versão mais recente do Papílio, chamado Papílio Versátil [Neto 2009], que será alvo da criptoanálise diferencial e da criptoanálise linear, a qual é mais complexa do que a aplicada neste trabalho. No caso, o número de ciclos varia de acordo com o texto de entrada.

Portanto, o Papílio Original é uma cifra forte, mas com ressalvas em relação ao tamanho do bloco de entrada e o tamanho das sub chaves utilizadas por ele. Afinal a chave é o principal mecanismo de segurança de uma cifra. Contudo, o Papílio Original é quebrável, e sua criptoanálise diferencial mostrou isso, descobrindo, ou mesmo reduzindo o intervalo de possíveis sub chaves para que seja possível sua quebra.

	Papílio						DES			BLOWFISH			FEAL		
CICLO	T. Esc.		T. Ana.		Comp.		T. Esc.	T. Ana.	Comp.	T. Esc.	T. Ana.	Comp.	T. Esc.	T. Ana.	Comp.
	8 B	**16 B**	**8 B**	**16 B**	**8 B**	**16 B**	**56 Bits**			**64 Bits**			**128 Bits**		
8	234	19608	44	568	234	19608	2^{21}	2^{20}	2^{9}	$3x2^{23}$	–	–	2^{32}	–	–
9	272	26316	12	316	272	26316	2^{26}	16	2^{32}	$3x2^{30}$	–	–	2^{37}	–	–
10	930	2580	460	116	930	2580	2^{35}	8	2^{15}	$3x2^{30}$	–	–	2^{42}	–	–
11	474	7224	120	100	474	7224	2^{36}	2^{12}	2^{32}	$3x2^{37}$	–	–	2^{47}	–	–
12	1020	7740	112	84	1020	7740	2^{43}	8	2^{21}	$3x2^{37}$	–	–	2^{52}	–	–
13	216	3870	108	56	216	3870	2^{44}	2^{20}	2^{32}	$3x2^{44}$	–	–	2^{62}	–	–
14	234	4644	52	160	234	4644	2^{51}	8	2^{29}	$3x2^{44}$	–	–	2^{67}	–	–
15	346	4644	64	20	346	4644	2^{52}	2^{28}	2^{37}	$3x2^{51}$	–	–	2^{72}	–	–
16	216	5676	116	264	216	5676	2^{58}	2^{6}	2^{37}	$3x2^{51}$	–	–	2^{77}	–	–

Tabela 6.1: Comparação do Papílio 8 e 16 bits, quanto a complexidade, quantidade de textos escolhidos e textos utilizados, com as cifras DES, BlowFish e FEAL.

Referências Bibliográficas

[Biham e Shamir 1990]BIHAM, E.; SHAMIR, A. Differential cryptanalysis of des-like cryptosystems. *CRYPTO 1990 and Journal of Cryptology*, Vol. 4, n. No. 1, p. 3–72, 1990.

[Biham e Shamir 1992]BIHAM, E.; SHAMIR, A. Differential cryptanalysis of the full 16-rounds des. *Proceedings of CRYPTO '92*, v. 740, 1992.

[Corporation, Adler e Gailly 2004]CORPORATION, P.; ADLER, M.; GAILLY, J.-L. *An Introduction to Cryptography*. Second. [S.l.: s.n.], 2004.

[Dunkelman, Biham e Keller 2007]DUNKELMAN, O.; BIHAM, E.; KELLER, N. A new attack on 6-round idea. *Springer-Verlag*, p. 211–224, 2007.

[Konheim 2007]KONHEIM, A. G. *Computer and Security and Cryptography*. First. Canada: [s.n.], 2007. (Wiley-Interscience). ISBN 978-0-471-94783-7.

[Lai e Massey 1990]LAI, X.; MASSEY, J. L. A proposal for a new block encryption standard. *EUROCRYPT*, p. 389–404, 1990.

[Nakaraha 2006]NAKARAHA, J. *Cirptoanálise Diferencial-Linear aplicada às Cifras FEAL-N e FEAL-NX*. Dissertação (Mestrado) — Universidade de São Paulo-USP, São Paulo, 2006.

[Neto 2009]NETO, J. B. de S. L. *Papílio Versátil: Um algoritmo criptográfico - Trabalho de conclusão de Curso*. Natal- RN, Dezember 2009.

[Oliveira 2003]OLIVEIRA, R. C. *Criptoanálise Diferencial*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife, 2003.

[Ramos 2002]RAMOS, K. D. N. *Papílio: Proposta de um Algoritmo de Criptografia Baseado no Algoritmo de Viterbi e Codificação Convolucional*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Natal, 2002.

[Rojas, Custódio e Berkenbrock 2004]ROJAS, M. A. T.; CUSTÓDIO, R. F.; BERKENBROCK, G. Segurança em cifradores de bloco simétricos. *I Workshop de Ciências da Computação e Sistemas da Informação da Região Sul - Anais do WORKCOMP-SUL*, 2004.

[Ryan e Nudd 1993]RYAN, M.; NUDD, G. R. The viterbi algorithm. *Department of Computer. Science, University of Warwick, Coventry, CV4 7AL*, February 1993.

[Smid. e Branstad 1998]SMID., M. E.; BRANSTAD, D. K. *The Data Encryption Standard Past and Future*. [S.l.], 1998. 43–64 p.

[Stallings 2005]STALLINGS, W. *Cryptography and Network Security Principles and Practices*. 4. ed. [S.l.: s.n.], 2005. (Prentice Hall).

APÊNDICE A – Códigos Fontes

A.1 Texto Base para Tabela com 8 Bits

```
00000000
0000
0000
0000
00000001
0001
0001
0001
00000010
0010
0101
0101
00000011
0011
0100
0100
00000100
0100
0011
0011
00000101
0101
0110
0110
00000110
0110
0010
0010
00000111
0111
0111
0111
00001000
1000
1110
1110
00001001
1001
1011
1011
00001010
```

```
42 1010
43 1111
44 1111
45 00001011
46 1011
47 1010
48 1010
49 00001100
50 1100
51 1001
52 1001
53 00001101
54 1101
55 1100
56 1100
57 00001110
58 1110
59 1000
60 1000
61 00001111
62 1111
63 1101
64 1101
65 00010000
66 0000
67 0000
68 0001
69 .
70 .
71 .
72 01111001
73 1001
74 1011
75 1100
76 01111010
77 1010
78 1111
79 1000
80 01111011
81 1011
82 1010
83 1101
84 01111100
85 1100
86 1001
87 1110
88 01111101
89 1101
90 1100
91 1011
92 01111110
93 1110
94 1000
95 1111
96 01111111
97 1111
98 1101
99 1010
```

A.2 Lista de Pares com 8 Bits

```
1  [0][0]={1={pS1=0, pE1=0, pE2=0, pS2=0}, 2={pS1=1, pE1=1, pE2=1, pS2=1}, 3={
       pS1=5, pE1=2, pE2=2, pS2=5}, 4={pS1=4, pE1=3, pE2=3, pS2=4}, 5={pS1=3,
       pE1=4, pE2=4, pS2=3}, 6={pS1=6, pE1=5, pE2=5, pS2=6}, 7={pS1=2, pE1=6,
       pE2=6, pS2=2}, 8={pS1=7, pE1=7, pE2=7, pS2=7}, 9={pS1=14, pE1=8, pE2=8,
       pS2=14}, 10={pS1=11, pE1=9, pE2=9, pS2=11}, 11={pS1=15, pE1=10, pE2=10,
       pS2=15}, 12={pS1=10, pE1=11, pE2=11, pS2=10}, 13={pS1=9, pE1=12, pE2=12,
        pS2=9}, 14={pS1=12, pE1=13, pE2=13, pS2=12}, 15={pS1=8, pE1=14, pE2=14,
        pS2=8}, 16={pS1=13, pE1=15, pE2=15, pS2=13}}
2  [1][1]={1={pS1=0, pE1=0, pE2=1, pS2=1}, 2={pS1=5, pE1=2, pE2=3, pS2
       =4}}[1][5]={1={pS1=3, pE1=4, pE2=5, pS2=6}, 2={pS1=2, pE1=6, pE2=7, pS2
       =7}, 3={pS1=14, pE1=8, pE2=9, pS2=11}, 4={pS1=15, pE1=10, pE2=11, pS2
       =10}, 5={pS1=9, pE1=12, pE2=13, pS2=12}, 6={pS1=8, pE1=14, pE2=15, pS2
       =13}}
3  [2][1]={1={pS1=3, pE1=4, pE2=6, pS2=2}, 2={pS1=6, pE1=5, pE2=7, pS2=7}, 3={
       pS1=14, pE1=8, pE2=10, pS2=15}, 4={pS1=11, pE1=9, pE2=11, pS2=10}, 5={
       pS1=9, pE1=12, pE2=14, pS2=8}, 6={pS1=12, pE1=13, pE2=15, pS2
       =13}}[2][5]={1={pS1=0, pE1=0, pE2=2, pS2=5}, 2={pS1=1, pE1=1, pE2=3, pS2
       =4}}
4  [3][4]={1={pS1=0, pE1=0, pE2=3, pS2=4}, 2={pS1=1, pE1=1, pE2=2, pS2=5}, 3={
       pS1=3, pE1=4, pE2=7, pS2=7}, 4={pS1=6, pE1=5, pE2=6, pS2=2}, 5={pS1=14,
       pE1=8, pE2=11, pS2=10}, 6={pS1=11, pE1=9, pE2=10, pS2=15}, 7={pS1=9, pE1
       =12, pE2=15, pS2=13}, 8={pS1=12, pE1=13, pE2=14, pS2=8}}
5  [4][3]={1={pS1=0, pE1=0, pE2=4, pS2=3}, 2={pS1=4, pE1=3, pE2=7, pS2
       =7}}[4][7]={1={pS1=1, pE1=1, pE2=5, pS2=6}, 2={pS1=5, pE1=2, pE2=6, pS2
       =2}, 3={pS1=14, pE1=8, pE2=12, pS2=9}, 4={pS1=11, pE1=9, pE2=13, pS2
       =12}, 5={pS1=15, pE1=10, pE2=14, pS2=8}, 6={pS1=10, pE1=11, pE2=15, pS2
       =13}}
6  [5][6]={1={pS1=0, pE1=0, pE2=5, pS2=6}, 2={pS1=4, pE1=3, pE2=6, pS2
       =2}}[5][2]={1={pS1=1, pE1=1, pE2=4, pS2=3}, 2={pS1=5, pE1=2, pE2=7, pS2
       =7}, 3={pS1=14, pE1=8, pE2=13, pS2=12}, 4={pS1=11, pE1=9, pE2=12, pS2
       =9}, 5={pS1=15, pE1=10, pE2=15, pS2=13}, 6={pS1=10, pE1=11, pE2=14, pS2
       =8}}
7  [6][6]={1={pS1=1, pE1=1, pE2=7, pS2=7}, 2={pS1=5, pE1=2, pE2=4, pS2=3}, 3={
       pS1=14, pE1=8, pE2=14, pS2=8}, 4={pS1=11, pE1=9, pE2=15, pS2=13}, 5={pS1
       =15, pE1=10, pE2=12, pS2=9}, 6={pS1=10, pE1=11, pE2=13, pS2
       =12}}[6][2]={1={pS1=0, pE1=0, pE2=6, pS2=2}, 2={pS1=4, pE1=3, pE2=5, pS2
       =6}}
8  [7][3]={1={pS1=1, pE1=1, pE2=6, pS2=2}, 2={pS1=5, pE1=2, pE2=5, pS2=6}, 3={
       pS1=14, pE1=8, pE2=15, pS2=13}, 4={pS1=11, pE1=9, pE2=14, pS2=8}, 5={pS1
       =15, pE1=10, pE2=13, pS2=12}, 6={pS1=10, pE1=11, pE2=12, pS2
       =9}}[7][7]={1={pS1=0, pE1=0, pE2=7, pS2=7}, 2={pS1=4, pE1=3, pE2=4, pS2
       =3}}
9  [8][14]={1={pS1=0, pE1=0, pE2=8, pS2=14}, 2={pS1=4, pE1=3, pE2=11, pS2
       =10}}[8][10]={1={pS1=1, pE1=1, pE2=9, pS2=11}, 2={pS1=5, pE1=2, pE2=10,
       pS2=15}, 3={pS1=3, pE1=4, pE2=12, pS2=9}, 4={pS1=6, pE1=5, pE2=13, pS2
       =12}, 5={pS1=2, pE1=6, pE2=14, pS2=8}, 6={pS1=7, pE1=7, pE2=15, pS2=13}}
10 [9][11]={1={pS1=0, pE1=0, pE2=9, pS2=11}, 2={pS1=4, pE1=3, pE2=10, pS2
       =15}}[9][15]={1={pS1=1, pE1=1, pE2=8, pS2=14}, 2={pS1=5, pE1=2, pE2=11,
       pS2=10}, 3={pS1=3, pE1=4, pE2=13, pS2=12}, 4={pS1=6, pE1=5, pE2=12, pS2
       =9}, 5={pS1=2, pE1=6, pE2=15, pS2=13}, 6={pS1=7, pE1=7, pE2=14, pS2=8}}
11 [10][11]={1={pS1=1, pE1=1, pE2=11, pS2=10}, 2={pS1=5, pE1=2, pE2=8, pS2
       =14}, 3={pS1=3, pE1=4, pE2=14, pS2=8}, 4={pS1=6, pE1=5, pE2=15, pS2=13},
        5={pS1=2, pE1=6, pE2=12, pS2=9}, 6={pS1=7, pE1=7, pE2=13, pS2
       =12}}[10][15]={1={pS1=0, pE1=0, pE2=10, pS2=15}, 2={pS1=4, pE1=3, pE2=9,
        pS2=11}}
```

```
12 [11][14]={1={pS1=1, pE1=1, pE2=10, pS2=15}, 2={pS1=5, pE1=2, pE2=9, pS2
       =11}, 3={pS1=3, pE1=4, pE2=15, pS2=13}, 4={pS1=6, pE1=5, pE2=14, pS2=8},
        5={pS1=2, pE1=6, pE2=13, pS2=12}, 6={pS1=7, pE1=7, pE2=12, pS2
       =9}}[11][10]={1={pS1=0, pE1=0, pE2=11, pS2=10}, 2={pS1=4, pE1=3, pE2=8,
       pS2=14}}
13 [12][9]={1={pS1=0, pE1=0, pE2=12, pS2=9}, 2={pS1=4, pE1=3, pE2=15, pS2
       =13}}[12][13]={1={pS1=1, pE1=1, pE2=13, pS2=12}, 2={pS1=5, pE1=2, pE2
       =14, pS2=8}, 3={pS1=3, pE1=4, pE2=8, pS2=14}, 4={pS1=6, pE1=5, pE2=9,
       pS2=11}, 5={pS1=2, pE1=6, pE2=10, pS2=15}, 6={pS1=7, pE1=7, pE2=11, pS2
       =10}}
14 [13][12]={1={pS1=0, pE1=0, pE2=13, pS2=12}, 2={pS1=4, pE1=3, pE2=14, pS2
       =8}}[13][8]={1={pS1=1, pE1=1, pE2=12, pS2=9}, 2={pS1=5, pE1=2, pE2=15,
       pS2=13}, 3={pS1=3, pE1=4, pE2=9, pS2=11}, 4={pS1=6, pE1=5, pE2=8, pS2
       =14}, 5={pS1=2, pE1=6, pE2=11, pS2=10}, 6={pS1=7, pE1=7, pE2=10, pS2
       =15}}
15 [14][12]={1={pS1=1, pE1=1, pE2=15, pS2=13}, 2={pS1=5, pE1=2, pE2=12, pS2
       =9}, 3={pS1=3, pE1=4, pE2=10, pS2=15}, 4={pS1=6, pE1=5, pE2=11, pS2=10},
        5={pS1=2, pE1=6, pE2=8, pS2=14}, 6={pS1=7, pE1=7, pE2=9, pS2
       =11}}[14][8]={1={pS1=0, pE1=0, pE2=14, pS2=8}, 2={pS1=4, pE1=3, pE2=13,
       pS2=12}}
16 [15][9]={1={pS1=1, pE1=1, pE2=14, pS2=8}, 2={pS1=5, pE1=2, pE2=13, pS2=12},
        3={pS1=3, pE1=4, pE2=11, pS2=10}, 4={pS1=6, pE1=5, pE2=10, pS2=15}, 5={
       pS1=2, pE1=6, pE2=9, pS2=11}, 6={pS1=7, pE1=7, pE2=8, pS2
       =14}}[15][13]={1={pS1=0, pE1=0, pE2=15, pS2=13}, 2={pS1=4, pE1=3, pE2
       =12, pS2=9}}
```

A.3 Texto Base para Tabela com 16 Bits

```
1  0000000000000000
2  00000000
3  00000000
4  00000000
5  0000000000000001
6  00000001
7  00000001
8  00000001
9  0000000000000010
10 00000010
11 00010001
12 00010001
13 0000000000000011
14 00000011
15 00010000
16 00010000
17 0000000000000100
18 00000100
19 00000011
20 00000011
21 0000000000000101
22 00000101
23 00010010
24 00010010
25 0000000000000110
26 00000110
27 00000010
28 00000010
```

```
29 0000000000000111
30 00000111
31 00010011
32 00010011
33 0000000000001000
34 00001000
35 00110010
36 00110010
37 0000000000001001
38 00001001
39 00100011
40 00100011
41 .
42 .
43 .
44 0000000011110011
45 11110011
46 11010110
47 11010110
48 0000000011110100
49 11110100
50 11010100
51 11010100
52 0000000011110101
53 11110101
54 11000101
55 11000101
56 0000000011110110
57 11110110
58 11010101
59 11010101
60 0000000011110111
61 11110111
62 11000100
63 11000100
64 0000000011111000
65 11111000
66 11100101
67 11100101
68 0000000011111001
69 11111001
70 11100100
71 11100100
72 0000000011111010
73 11111010
74 11110100
75 11110100
76 0000000011111011
77 11111011
78 11110101
79 11110101
80 0000000011111100
81 11111100
82 11100111
83 11100111
84 0000000011111101
85 11111101
86 11100110
```

```
11100110
0000000011111110
11111110
11110110
11110110
0000000011111111
11111111
11110111
11110111
```

A.4 Lista de Pares com 16 Bits

```
0][0]={1={pS1=0, pE1=0, pE2=0, pS2=0}, 2={pS1=1, pE1=1, pE2=1, pS2=1}, 3={
    pS1=17, pE1=2, pE2=2, pS2=17}, 4={pS1=16, pE1=3, pE2=3, pS2=16}, 5={pS1
    =3, pE1=4, pE2=4, pS2=3}, 6={pS1=18, pE1=5, pE2=5, pS2=18}, 7={pS1=2,
    pE1=6, pE2=6, pS2=2}, 8={pS1=19, pE1=7, pE2=7, pS2=19}, 9={pS1=50, pE1
    =8, pE2=8, pS2=50}, 10={pS1=35, pE1=9, pE2=9, pS2=35}, 11={pS1=51, pE1
    =10, pE2=10, pS2=51}, 12={pS1=34, pE1=11, pE2=11, pS2=34}, 13={pS1=33,
    pE1=12, pE2=12, pS2=33}, 14={pS1=48, pE1=13, pE2=13, pS2=48}, 15={pS1
    =32, pE1=14, pE2=14, pS2=32}, 16={pS1=49, pE1=15, pE2=15, pS2=49}, 17={
    pS1=6, pE1=16, pE2=16, pS2=6}, 18={pS1=7, pE1=17, pE2=17, pS2=7}, 19={
    pS1=23, pE1=18, pE2=18, pS2=23}, 20={pS1=22, pE1=19, pE2=19, pS2=22},
    21={pS1=37, pE1=20, pE2=20, pS2=37}, 22={pS1=36, pE1=21, pE2=21, pS2
    =36}, 23={pS1=52, pE1=22, pE2=22, pS2=52}, 24={pS1=53, pE1=23, pE2=23,
    pS2=53}, ...
.
.
.
[66][13]={1={pS1=3, pE1=4, pE2=70, pS2=14}, 2={pS1=18, pE1=5, pE2=71, pS2
    =31}, 3={pS1=2, pE1=6, pE2=68, pS2=15}, 4={pS1=19, pE1=7, pE2=69, pS2
    =30}, 5={pS1=50, pE1=8, pE2=74, pS2=63}, 6={pS1=35, pE1=9, pE2=75, pS2
    =46}, 7={pS1=51, pE1=10, pE2=72, pS2=62}, 8={pS1=34, pE1=11, pE2=73, pS2
    =47}, 9={pS1=33, pE1=12, pE2=78, pS2=44}, 10={pS1=48, pE1=13, pE2=79,
    pS2=61}, 11={pS1=32, pE1=14, pE2=76, pS2=45}, 12={pS1=49, pE1=15, pE2
    =77, pS2=60}}[66][29]={1={pS1=0, pE1=0, pE2=66, pS2=29}, 2={pS1=1, pE1
    =1, pE2=67, pS2=28}, 3={pS1=17, pE1=2, pE2=64, pS2=12}, 4={pS1=16, pE1
    =3, pE2=65, pS2=13}, 5={pS1=66, pE1=48, pE2=114, pS2=95}, 6={pS1=67, pE1
    =49, pE2=115, pS2=94}, 7={pS1=83, pE1=50, pE2=112, pS2=78}, 8={pS1=82,
    pE1=51, pE2=113, pS2=79}, 9={pS1=99, pE1=60, pE2=126, pS2=126}, 10={pS1
    =98, pE1=61, pE2=127, pS2=127}, 11={pS1=114, pE1=62, pE2=124, pS2=111},
    12={pS1=115, pE1=63, pE2=125, pS2=110}}[66][60]={1={pS1=97, pE1=52, pE2
    =118, pS2=93}, 2={pS1=113, pE1=55, pE2=117, pS2=77}, 3={pS1=65, pE1=57,
    pE2=123, pS2=125},
.
.
.
[255][148]={1={pS1=92, pE1=116, pE2=139, pS2=200}, 2={pS1=77, pE1=117, pE2
    =138, pS2=217}, 3={pS1=93, pE1=118, pE2=137, pS2=201}, 4={pS1=76, pE1
    =119, pE2=136, pS2=216}, 5={pS1=109, pE1=120, pE2=135, pS2=249}, 6={pS1
    =108, pE1=121, pE2=134, pS2=248}, 7={pS1=124, pE1=122, pE2=133, pS2
    =232}, 8={pS1=125, pE1=123, pE2=132, pS2=233}}[255][135]={1={pS1=37, pE1
    =20, pE2=235, pS2=162}, 2={pS1=53, pE1=23, pE2=232, pS2=178}, 3={pS1=20,
     pE1=24, pE2=231, pS2=147}, 4={pS1=5, pE1=25, pE2=230, pS2=130}, 5={pS1
    =21, pE1=26, pE2=229, pS2=146}, 6={pS1=4, pE1=27, pE2=228, pS2=131}, 7={
    pS1=11, pE1=100, pE2=155, pS2=140}, 8={pS1=26, pE1=101, pE2=154, pS2
    =157}, 9={pS1=10, pE1=102, pE2=153, pS2=141}, 10={pS1=27, pE1=103, pE2
```

```
=152, pS2=156}, 11={pS1=58, pE1=104, pE2=151, pS2=189}, 12={pS1=42, pE1
=107, pE2=148, pS2=173}}[255][150]={1={pS1=6, pE1=16, pE2=239, pS2=144},
 2={pS1=7, pE1=17, pE2=238, pS2=145}, 3={pS1=23, pE1=18, pE2=237, pS2
=129}, 4={pS1=22, pE1=19, pE2=236, pS2=128}, 5={pS1=39, pE1=28, pE2=227,
 pS2=177}, 6={pS1=55, pE1=31, pE2=224, pS2=161}, 7={pS1=101, pE1=32, pE2
=223, pS2=243}, 8={pS1=117, pE1=35, pE2=220, pS2=227}, 9={pS1=68, pE1
=44, pE2=211, pS2=210}, 10={pS1=69, pE1=45, pE2=210, pS2=211}, 11={pS1
=85, pE1=46, pE2=209, pS2=195}, 12={pS1=84, pE1=47, pE2=208, pS2=194},
13={pS1=74, pE1=80, pE2=175, pS2=220}, 14={pS1=75, pE1=81, pE2=174, pS2
=221}, 15={pS1=91, pE1=82, pE2=173, pS2=205}, 16={pS1=90, pE1=83, pE2
=172, pS2=204}, 17={pS1=107, pE1=92, pE2=163, pS2=253}, 18={pS1=123, pE1
=95, pE2=160, pS2=237}, 19={pS1=41, pE1=96, pE2=159, pS2=191}, 20={pS1
=57, pE1=99, pE2=156, pS2=175}, 21={pS1=8, pE1=108, pE2=147, pS2=158},
22={pS1=9, pE1=109, pE2=146, pS2
```

A.5 Codigo Fonte: tabXor.swp.lua

```
1  local Debug = require('dbg')
2  require("BinDecHex")
3
4  local BUFSIZE = 2^13
5  local fXors, err = io.open(arg[1], "r")
6  local iCount, jCount = 0, 0
7  local fLine
8  local tabDist = {}
9  local tLines, tCols = {}, {}
10
11 print(os.time(), os.clock())
12 print("GC Count:", collectgarbage("count"))
13 while true do
14   local contador=0
15   local regLine = {}
16   local wBrk = false
17   for i=1,4 do
18     fLine = fXors:read()
19     if not fLine then wBrk = true break end regLine[#regLine+1] = fLine
20     end
21     if wBrk then break end
22     local fSub, err = io.open(arg[1], "r")
23     if err then
24       error("Erro de leitura", err)
25     end
26     jCount = 0
27
28     while true do
29       local regSub = {}
30       wBrk = false
31       for i=1,4 do
32         fLine = fSub:read()
33         if not fLine then wBrk = true break end regSub[#regSub+1] = fLine
34       end
35       if wBrk then break end
36       if jCount >= iCount then
37         local entraBin1, entraBin2, saidaBin1, saidaBin2 = Bin2Dec(regLine
               [2]), Bin2Dec(regSub[2]), Bin2Dec(regLine[3]), Bin2Dec(regSub
               [3])
```

```
38          local xorEntrada = bxor(entraBin1, entraBin2)
39          local xorSaida = bxor(saidaBin1, saidaBin2)
40          contador=contador+1
41          if not tabDist[xorEntrada] then
42            local iL=#tLines+1
43            tabDist[xorEntrada] = {}
44            tLines[iL] = xorEntrada
45          end
46
47        if not tabDist[xorEntrada][xorSaida] then
48          local criaCol = true
49          for _i=1,#tCols do
50            if tCols[_i] == xorSaida then
51            criaCol = false
52            break
53          end
54        end
55        if criaCol then
56          local iC =  #tCols+1
57          tCols[iC] = xorSaida
58        end
59        tabDist[xorEntrada][xorSaida] = {value=0,listPares={}}
60        end
61        local reg = tabDist[xorEntrada][xorSaida]
62        reg.value = reg.value + 1
63        reg.listPares[#reg.listPares+1] = {pEntrada1=entraBin1,pEntrada2=
              entraBin2, pSaida1=saidaBin1, pSaida2=saidaBin2}
64     end
65     jCount = jCount + 1
66  end
67  if (contador % 100) == 0 then print(contador) end
68  fSub:close()
69  iCount = iCount + 1
70
71  local fTab, err = io.open(arg[2], "wr")
72  if err then
73    error("Erro de escrita no arquivo", err)
74  end
75  local fLista, err = io.open("ListaPares.txt", "wr")
76  if err then
77    error("Erro de escrita no arquivo", err)
78  end
79
80  local lines = string.rep("X", 5)
81  fTab:write(lines)
82  local maximo=0
83  local atual
84  local num1
85  for i, vi in ipairs(tCols) do
86    fTab:write("\t" .. string.format("%5d",vi))
87    num1=vi
88      for i,vi in ipairs(tCols) do
89      atual=math.max(num1,vi)
90      end
91    if atual > maximo then
92       maximo=atual
93    end
94
```

```
95  end
96  fTab:write("\n")
97
98  local contadorGL,contadorGC=1,1
99
100 for i, vi in ipairs(tLines) do
101   contadorGL=contadorGL+1
102   fTab:write(string.format("%5d",vi))
103     for j, vj in pairs(tCols) do
104       if tabDist[vi][vj] and tabDist[vi][vj].value > 0 then
105         fTab:write("\t" .. string.format("%5d",tabDist[vi][vj].value))
106         fLista:write(string.format("[%s][%s]=%s",vi,vj,
107          Debug.tostring(tabDist[vi][vj].listPares)))
108       else
109         fTab:write("\t" .. string.rep(" ", 5))
110       end
111     end
112 fTab:write("\n")
113 fLista:write("\n")
114
115 if (i % 100) == 0 then
116     fTab:flush()
117     fLista:flush()
118   end
119 end
120 fTab:close()
121 end
122 fXors:close()
```

A.6 Código Fonte para Analisar as Possíveis Sub Chaves: Lua Analisador.lua

```
1  local Debug = require('dbg')
2  require("BinDecHex")
3
4  local BUFSIZE = 2^13
5  local Ger_Pap, err1 = io.open(arg[1], "r")
6  local Ger_Pap1, err1 = io.open(arg[1], "r")
7  local Lis_Par, err2 = io.open(arg[2], "r")
8  local regline_Ger_Pap = {}
9  local regline_Ger_Pap1 = {}
10 local regline_Lis_Par = {}
11 local regline_Par_Sug = {}
12 local regline_Lis_Par_Xor_Sai = {}
13 local regline_Lis_Sub_Cha_Pro = {}
14 local regline_Sub_Chaves_Mais_Provaveis = {}
15 local Saida_Num1,Saida_Num2=0,0
16 local cont=0
17 local cont1=0
18 local GerPLineC
19 local GerPLineC1
20 local GerPLineDC
21 local GerPLineDC
22 local GerPLineVi
23 local GerPLineEC
24 local Lista_Pares_Xor_Saida_Contador = {}
```

```
25 local cont2=0
26 local reg_subchaves = {}
27 local subchave_escolhida=0
28 local aux,indiceMAX=0,0
29 local xor_subchave=0
30 local ciclo=0
31 fLine = Ger_Pap:read()
32
33 while ciclo<500 do
34   ciclo=ciclo+1
35   local GerPLineC=0
36   local GerPLineC1=0
37   local GerPLineDC=0
38   local GerPLineDC=0
39   local GerPLineVi=0
40   local GerPLineEC=0
41   local Xor_Saida_Num_Sug=0
42
43   local Ger_Pap, err1 = io.open(arg[1], "r")
44
45   local Ger_Pap1, err1 = io.open(arg[1], "r")
46
47   local Lis_Par, err2 = io.open(arg[2], "r")
48
49   local numero_ciclo = arg[3]
50   print(numero_ciclo)
51   local Num_Cilco={"Apos XOR de resultado com EC: ",}
52   ciclo=ciclo+1
53   NumBits=string.len(string.match(fLine, "%d+"))
54   print("Numero de Bits de Entrada eh:",NumBits)
55   local NumSug1 =  math.random (0,256)
56   print("Randomico 1",NumSug1)
57   local NumSug2 = math.random (0,256)
58   print("Randomico 2",NumSug2)
59   local XorSug=bxor(tonumber(NumSug1),tonumber(NumSug2))
60   print("XOR ENTRADA",XorSug)
61
62   while true do
63     lstLine = Lis_Par:read()
64     if not lstLine then break end
65     if string.find(lstLine,"["..XorSug.."][",1,true) then
66       regline_Lis_Par[#regline_Lis_Par+1]=lstLine
67       for s in string.gmatch(lstLine,".-}}") do
68         cont=cont+1
69         regline_Lis_Par_Xor_Sai[#regline_Lis_Par_Xor_Sai+1]=s
70       end
71     end
72   end
73
74   if numero_ciclo ~='0' then --para rounds 1,2,3,4,5...15
75      for i=0,0 do
76         while true do
77         GerPLineC  = Ger_Pap:read()
78         GerPLineC1 = Ger_Pap1:read()
79          if not GerPLineC then break end
80            if string.find(GerPLineC,"---> Ciclo "..(numero_ciclo-1),1,true)
                 then
81              GerPLineDC = Ger_Pap:read()    --Apos XOR DC
```

```
82              GerPLineDC = Ger_Pap:read()    --Apos Viterbi
83              GerPLineEC = Ger_Pap:read()    --Apos XOR de resultado com EC
84              if string.find(GerPLineEC,"Apos XOR de resultado com EC: "..
85                ..Dec2Bin(NumSug1,NumBits/2),1,true) then
86                print("num_sug em 8 bits:",Dec2Bin(NumSug1,NumBits/2))
87                print(GerPLineEC)
88                GerPLineDC = Ger_Pap:read()
89                GerPLineDC = Ger_Pap:read()
90                GerPLineVi = Ger_Pap:read() --  lendo a saida do..
91                ..viterbi respectiva do num_sug1
92                print(GerPLineVi)
93                Saida_Num1=string.match(GerPLineVi,"%d+")
94                print("bits saida1",string.match(GerPLineVi,"%d+"))
95                print("Binario Saida1",Bin2Dec(Saida_Num1))
96              end
97          end
98          if not GerPLineC1 then break end
99          if string.find(GerPLineC1,"--> Ciclo "..(numero_ciclo-1),1,true)
                then
100           GerPLineDC1 = Ger_Pap1:read()    --Apos XOR DC
101           GerPLineDC1 = Ger_Pap1:read()    --Apos Viterbi
102           GerPLineEC1 = Ger_Pap1:read()    --Apos XOR de resultado com EC
103           if string.find(GerPLineEC,"Apos XOR de resultado com EC:"..
                Dec2Bin(NumSug2,NumBits/2),1,true) then
104             print("num_sug em 8 bits:",Dec2Bin(NumSug2,NumBits/2))
105             print(GerPLineEC1)
106             GerPLineDC1 = Ger_Pap1:read()
107             GerPLineDC1 = Ger_Pap1:read()
108             GerPLineVi1 = Ger_Pap1:read() --  lendo a saida do viterbi
                  respectiva do num_sug1
109             print(GerPLineVi1)
110             Saida_Num2=string.match(GerPLineVi1,"%d+")
111             print("bits saida2",string.match(GerPLineVi1,"%d+"))
112             print("Binario Saida2",Bin2Dec(Saida_Num2))
113           end
114         end
115       end
116       Xor_Saida_Num_Sug=bxor(tonumber(Bin2Dec(Saida_Num1)),tonumber(Bin2Dec
            (Saida_Num2)))
117       print("XOR ENTRADA",XorSug)
118       print("XOR SAIDA",Xor_Saida_Num_Sug)
119   end
120 end
121
122 if numero_ciclo=='0' then-- fim do if se ciclo menor que zero
123   for i=0,0 do
124   print("antes do while do for do i=0,0")
125   while true do
126    GerPLineC  = Ger_Pap:read()
127    GerPLineC1 = Ger_Pap1:read()
128    if not GerPLineC then break end
129    if string.find(GerPLineC,"Conjunto de 16 bits para codificacao:"..
         Dec2Bin(NumSug1,NumBits),1,true) then
130      GerPLineDC = Ger_Pap:read()
131      GerPLineDC = Ger_Pap:read()
132      GerPLineVi = Ger_Pap:read()
133      GerPLineEC = Ger_Pap:read()
134      print(GerPLineVi)
```

```
135       Saida_Num1=string.match(GerPLineVi,"%d+")
136       print("bits saida1",string.match(GerPLineVi,"%d+"))
137       print("Binario Saida1",Bin2Dec(Saida_Num1))
138     end
139     if not GerPLineC1 then break end
140     if string.find(GerPLineC1,"Conjunto de 16 bits para codificacao: "..
            Dec2Bin(NumSug2,NumBits),1,true) then
141       GerPLineDC1 = Ger_Pap1:read()
142       GerPLineDC1 = Ger_Pap1:read()
143       GerPLineVi1 = Ger_Pap1:read()
144       GerPLineEC1 = Ger_Pap1:read()
145       print(GerPLineVi1)
146       Saida_Num2=string.match(GerPLineVi1,"%d+")
147       print("bits saida2",string.match(GerPLineVi1,"%d+"))
148       print("Binario Saida2",Bin2Dec(Saida_Num2))
149     end
150   end
151     Xor_Saida_Num_Sug=bxor(tonumber(Bin2Dec(Saida_Num1)),tonumber(Bin2Dec(
          Saida_Num2)))
152     print("XOR ENTRADA",XorSug)
153     print("XOR SAIDA",Xor_Saida_Num_Sug)
154 end
155 end
156
157 for k,v in ipairs(regline_Lis_Par_Xor_Sai) do
158   if string.find(v,"["..XorSug.."]["..Xor_Saida_Num_Sug.."]",1,true) then
159     print("XOR ENTRADA dentro",XorSug)
160     print("XOR SAIDA dentro",Xor_Saida_Num_Sug)
161     print(v)
162     for k2,v2 in string.gmatch(v,"(pE%d)=(%d+)") do
163       print(k2, v2)
164       xor_subchave = bxor(tonumber(v2),tonumber(NumSug1))
165       print("XOR DA SUBCHAVE",xor_subchave)
166       cont2=cont2+1
167       if Lista_Pares_Xor_Saida_Contador[xor_subchave] then
168         Lista_Pares_Xor_Saida_Contador[xor_subchave]=..
169         ..Lista_Pares_Xor_Saida_Contador[xor_subchave]+1
170       else
171         Lista_Pares_Xor_Saida_Contador[xor_subchave]=1
172       end
173     end
174   end
175 end
176
177 for k,v in pairs(Lista_Pares_Xor_Saida_Contador) do
178     if v>aux then
179     indiceMAX=v
180     aux=indiceMAX
181     end
182 end
183 print("MAX",indiceMAX)
184 local l={}
185 for k,v in pairs(Lista_Pares_Xor_Saida_Contador) do
186 regline_Lis_Sub_Cha_Pro={}
187     if v>=indiceMAX then
188     l[#l+1] = k
189     print(k,v,l[#l])
190     print("sub chave mais provavel",k)
```

```
191     end
192   end
193   print("Numeros de Possiveis Subchaves.",#l)
194   for k1,v1 in ipairs(l) do
195
196   print("valor da chave mas prov",v1)
197   end
198
199   Ger_Pap:close()
200   Ger_Pap1:close()
201   Lis_Par:close()
202
203   print("**********************************\n")
204   print("Nº",ciclo)
205   print("Numero de Op",cont2)
206   if #l==2 then break end
207   end
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo