UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIAS E GEOCIÊNCIAS
DEPARTAMENTO DE ELETRÔNICA E SISTEMAS

RAIMUNDO CORRÊA DE OLIVEIRA

CRIPTOANÁLISE DIFERENCIAL

Este trabalho foi apresentado à Pós-Graduação em Engenharia Elétrica do Centro de Tecnologias e Geociências da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

ORIENTADOR: Prof. Ricardo Menezes Campello de Souza

Recife – Setembro de 2003

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIAS E GEOCIÊNCIAS
DEPARTAMENTO DE ELETRÔNICA E SISTEMAS

Pós-Graduação em Engenharia Elétrica

CRIPTOANÁLISE DIFERENCIAL

por

Raimundo Corrêa de Oliveira

Dissertação de Mestrado

Recife – Setembro de 2003

Universidade Federal de Pernambuco

Pós-Graduação em Engenharia Elétrica

PARECER DA COMISSÃO EXAMINADORA DE DEFESA DE DISSERTAÇÃO DE MESTRADO PROFISSIONALIZANTE DE

RAIMUNDO CORRÊA DE OLIVEIRA

TÍTULO

"CRIPTOANÁLISE DIFERENCIAL"

A comissão examinadora composta pelos professores: RICARDO MENEZES CAMPELLO DE SOUZA, DES/UFPE, VALDEMAR CARDOSO DA ROCHA JÚNIOR, DES/UFPE E CARLOS ALEXANDRE BARROS DE MELO, Sistemas Computacionais/UPE, sob a presidência do primeiro, consideram o candidato **RAIMUNDO CORRÊA DE OLIVEIRA** **APROVADO**.

Recife, 30 de setembro de 2003.

RICARDO MENEZES CAMPELLO DE SOUZA

VALDEMAR CARDOSO DA ROCHA JÚNIOR

CARLOS ALEXANDRE BARROS DE MELO

À Rosangela, Benedita,
Nazaré, Sebastiana e Roberta

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, pois sem ELE não seria possível a realização desse trabalho.

À todos os professores e funcionários do DES que de uma forma indireta contribuíram muito solucionando minhas dúvidas e ajudando na vida diária em Recife. Ao professor Rafael Dueire Lins pelo seu esforço para dar continuidade no nosso curso e pelo apoio dispensado em minha estada em Recife. Ao professor Ricardo Menezes Campello de Souza pela sua paciência em ensinar toda a teoria básica que eu precisei para desenvolver este trabalho.

Aos amigos, Jucimar Maia da Silva Jr. e Ricardo da Silva Barboza pelo apoio dado em todo o curso de mestrado e além de suas contribuições para este trabalho.

À minha mãe, Maria de Nazaré Corrêa de Oliveira, pelo apoio dado em toda minha vida e seus conselhos.

À minha Tia, Sebastiana Maria V. de Oliveira, pelo apoio dado durante a minha vida e seus incentivos para termino do mestrado.

À minha Esposa, Rosangela Lima da Costa, por sua dedicação e paciência durante esta jornada de mestrado.

Em especial, à minha Avó, Benedita Corrêa Vinhoti, que me proporcionou toda a base de respeito, honestidade e dedicação para um ser humano, sem os quais não teria enfrentado todas as barreiras da minha vida.

RESUMO

Desde a criação do padrão de cifragem de dados DES (*Data Encryption Standard*) em 1978, cifras iterativas de chave secreta baseadas em redes de Feistel têm sido construídas, visando a aumentar a segurança das informações armazenadas e transmitidas em ambientes de comunicações inseguros. Tais cifras, além de apresentarem, em geral, um alto nível de resistência a ataques criptoanalíticos, permitem implementações eficientes em diferentes plataformas, sendo portanto uma alternativa muito atraente para a proteção de grandes massas de dados. Uma das técnicas mais eficientes para criptoanalisar esse tipo de cifra é a chamada Criptoanálise Diferencial, introduzida em 1990 por Biham e Shamir.

Este trabalho tem como objetivo apresentar um estudo da técnica de Criptoanálise Diferencial, bem como demonstrar uma aplicação da mesma na criptoanálise do cripto-sistema de chave secreta Blowfish, introduzido por Schneier em 1994. Inicialmente o trabalho aborda os principais aspectos matemáticos relacionados com a criptografia de chave secreta e, em seguida, apresenta uma introdução à área de Segurança de Dados, discorrendo sobre aspectos importantes da mesma, tais como criptografia de chave secreta e de chave pública. Além disso, são também descritos os cripto-sistemas de chave secreta DES, RC5, IDEA e Blowfish. A técnica da Criptoanálise Diferencial é então considerada em detalhes, onde são examinados os conceitos mais importantes da mesma, tais como as noções de pares diferenciais e características, entre outros. Uma aplicação da técnica é então apresentada, na avaliação do grau de segurança da cifra de bloco Blowfish.

ABSTRACT

Since the creation of the standard of encryption of data DES (Data Encryption Standard) in 1978, iterative ciphers of based private key in nets of Feistel have been constructed, aiming at to increase the security of the information stored and transmitted in unsafe environments of communications. Such ciphers, besides in general presenting one high level of resistance the criptoanalíticos attacks, allow to efficient implementations in different platforms, being therefore a very attractive alternative for the protection of great masses of data. One of the techniques most efficient to criptoanalisar this type of cipher is the call Differential Cryptoanalysis, introduced in 1990 by Biham and Shamir.

This work has as objective to present a study of the technique of Differential Cryptoanalysis, as well as demonstrating an application of the same one in the cryptoanalysis of the crypto-system of private key Blowfish, introduced for Schneier in 1994. Initially the work approaches the main related mathematical aspects with the cryptography of private key and, after that, presents an introduction to the area of Data locking, discoursing on important aspects of the same one, such as cryptography of private key and public key. Moreover, the crypto-systems of private key DES, RC5, IDEA and Blowfish are also described. The technique of the Differential Cryptoanalysis then is considered in details, where the most important concepts of the same one are examined, such as slight knowledge of the differential and characteristic pairs, among others. An application of the technique then is presented, in the evaluation of the degree of security of the block cipher Blowfish.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

1

INTRODUÇÃO

Com o crescente aumento do uso de redes de computadores para facilitar as comunicações entre pessoas, pois estas estruturas proporcionam serviços úteis para o dia-a-dia da população, tais como e-mail, transação bancária, acesso a banco de dados, etc., a segurança destes serviços está despontando como um problema real [AST97]. Nesse cenário, sistemas proporcionando sigilo, autenticidade e integridade, podem ser projetados com o auxílio da criptografia, mostrando que a mesma é uma ferramenta fundamental para construção de sistemas seguros [DDW89].

1.1 Criptografia

A segurança da comunicação entre dois usuários está baseada no modo como o remetente irá esconder a informação para que um terceiro usuário desautorizado não venha a ter acesso à mesma.

A criptografia estuda os métodos para codificar uma mensagem de modo que só seu destinatário legítimo consiga interpretá-la. Se o objetivo é sigilo, então todo sistema é composto no mínimo por uma função de cifragem e uma função de decifragem. Cifragem é o processo de transformar um texto claro em texto cifrado. Decifragem é o processo reverso da cifragem, transforma texto cifrado em um texto claro. Estes processos são controlados por uma chave.

O propósito original das cifras é o de ocultar informações quando estas forem transmitidas. Porém, informações armazenadas podem ser ocultadas de forma similar, mas a segurança de sistemas de informações é muito mais ampla que simplesmente o sigilo e inclui a

integridade de mensagens e a autenticidade das partes envolvidas na comunicação [DDW89].

Segundo Diffie e Hellman [DWH76], a criptografia é o estudo de sistemas "matemáticos" para resolver dois tipos de problemas de segurança: Privacidade e Autenticação.

A privacidade é a propriedade que um sistema criptográfico fornece para que uma pessoa desautorizada não possa entender e não consiga decifrar a mensagem interceptada.

A autenticação garante ao receptor que o remetente da mensagem seja realmente quem ele disse que era no início da comunicação.

Uma outra propriedade importante que um sistema criptográfico tem que garantir é a integridade das informações que são enviadas. Ou seja, garantir que uma mensagem não foi modificada acidentalmente ou deliberadamente durante o trafego, quer seja, por troca, inserção ou exclusão [GJS92].

O termo texto claro é usado para indicar que o significado da informação pode ser facilmente compreendido sem informação adicional, e o termo texto cifrado indica que, em princípio, seu conteúdo é ilegível (a um intruso).

Um cripto-sistema é o conjunto de textos claro, textos cifrados, transformações e chaves. Este tem que garantir todas a propriedades citadas acima. A cifra é um algoritmo de cifragem que transforma um texto claro em um texto cifrado sob controle de uma chave de cifragem. Analogamente, a transformação inversa, que recupera um texto claro, a partir de um texto cifrado, através de uma chave de decifragem, é o algoritmo de decifragem associado.

A necessidade da existência de uma chave criptográfica é devida à possibilidade das cifras serem de conhecimento público, ou mesmo poderem ser deduzidas ou descobertas por um intruso de modo ilegal. A chave é um parâmetro que deve ser mantido em segredo. Nota-se, é desejável que a segurança de uma cifra se concentre exclusivamente, ou no maior grau possível, no segredo da chave. Com isso, a cifra não deve gerar um texto cifrado que contenha alguma informação sobre a chave ou sobre o texto claro.

De acordo com Diffie e Hellman [DWH76], cifras que têm a características de propagação de erro (ou seja, qualquer mudança mínima no texto cifrado irá gerar uma mensagem não autêntica) devem ser empregadas em cripto-sistemas para que garanta a integridade da mensagem.

Uma característica muito importante para um cripto-sistema, é fazer a cifragem e decifragem em poucas operações, sem deixar informações para que uma criptoanálise possa quebrar a cifra. Sem isso o sistema se tornará inviável para a maioria das aplicações. Podemos classificar de dois modos um cripto-sistema quanto a sua segurança: computacionalmente seguro e incondicionalmente seguro. O primeiro se refere ao fato de que um espião que tenha computação ilimitada possa decifrar a mensagem. O segundo resistirá a todos os ataques possíveis e sua segurança, em hipótese alguma, não será violada.

Atualmente têm-se dois tipos de cripto-sistema:

- **Chave Secreta ou Simétrico**: neste tipo os dois usuários tem que trocar uma chave secreta, que servirá para cifragem e decifragem das informações trocadas, antes de iniciar a comunicação. Como exemplo: IDEA e AES.
- **Chave Pública ou Assimétrico**: neste sistema temos uma chave pública e uma chave privada, e elas são diferentes no remetente e no receptor . Quando um usuário A deseja enviar uma mensagem para um usuário B, este deve usar a chave pública de B e cifrar a mensagem. Na recepção, o usuário B decifra a mensagem com sua chave privada. Como exemplo: RSA e ElGamal.

Os cripto-sistemas têm que ter sua segurança avaliada para que possamos usá-lo sem o perigo de um usuário desautorizado possa descobrir as mensagens que foi cifrada com tal cifra. Para avaliação da segurança de um cripto-sistema devemos empregar técnicas reconhecidas no meio científico e verificar se a cifra foi quebrada ou se resistiu ao ataque. Depois de avaliada e comprovada sua segurança poderemos usá-la e sempre ficarmos atento para ataques futuros.

Um ataque possível consiste em, dado um texto claro M e o texto cifrado correspondente C, testar todas as chaves do cripto-sistema em busca daquela que, quando aplicada a M, produz C. Com este tipo de ataque, denominado ataque por busca exaustiva da chave ou ataque por força bruta, todas as cifras em uso são teoricamente “quebráveis” com ele.

A busca exaustiva, em geral, usa uma quantidade fixa de memória, necessita de poucas amostras de texto cifrado (e/ou claro), mas o número médio de operações aritméticas

tornam este ataque inviável na prática. Por exemplo, se uma determinada chave tiver n bits, a complexidade do ataque será $O(2^n)$, isto é, exponencial no número de bits da chave.

Entretanto, se o limite é à procura de todas as chaves, podemos avaliar uma cifra se ela resiste a um ataque que seja de complexidade menor que o ataque por busca exaustiva. Se uma cifra sucumbir a tal tipo de ataque ela pode ser considerada insegura. O método de criptoanálise diferencial é um tipo de ataque estatístico por texto claro escolhido que pode ser aplicado a cifras de bloco e servirá de parâmetro para assegurar que uma cifra é segura.

1.2 Objetivo e Organização da dissertação

O objetivo deste trabalho é apresentar um estudo da técnica de criptoanálise diferencial, incluindo uma aplicação da mesma no cripto-sistema de chave secreta Blowfish.
O Blowfish é uma cifra de feistel que utiliza S-Boxes dependentes da chave, ficando muito difícil sua análise com o método de Criptoanálise diferencial. Um ataque publicado de Serge Vaudenay mostrou-se eficiente na procura por chaves fracas do Blowfish.
A dissertação está organizada da seguinte forma:
Capitulo 2 – Fundamentos Matemáticos apresenta ferramentas matemáticas empregadas no texto, com ênfase nos assuntos: Grupos e Corpos Finitos.
Capitulo 3 – Introdução a Criptografia apresenta uma breve introdução à área de criptografia. Neste capítulo são descritos alguns dos principais cripto-sistemas de chave secreta atualmente em uso, tais como DES, RC5 e IDEA. Também é descrito o cripto-sistema de chave secreta Blowfish, que será atacado com a criptoanálise diferencial.
Capítulo 4 – Criptoanálise Diferencial descreve a técnica da criptoanálise diferencial, proposta por Eli Biham e Adi Shamir em 1990. Nesse capítulo, a técnica é empregada na criptoanálise de uma cifra, sem fins práticos, para mostrar o emprego da técnica.
Capítulo 5 – Criptoanálise Diferencial do Cripto-Sistema Blowfish apresenta o emprego da técnica na criptoanálise da cifra Blowfish.
Capítulo 6 – Conclusões e sugestões para futuros trabalhos apresenta uma análise crítica, feita sobre os resultados obtidos com o ataque da criptoanálise diferencial. Possíveis trabalhos futuros são propostos.

Apêndice A – Algoritmo do Blowfish Reduzido em linguagem C apresenta a listagem do algoritmo em C que foi implementado para efetuar a experiência de criptoanálise diferencial mostrada nesta dissertação.

Apêndice B – Algoritmo do Blowfish em linguagem C apresenta a listagem do algoritmo em C que foi implementado para efetuar a experiência de criptoanálise diferencial mostrada nesta dissertação.

Apêndice C – Tabela de Freqüência para as Diferenças de Saída dos S-Boxes mostra a tabela de freqüência das diferenças de saída para os S-Boxes do Blowfish reduzido.

Apêndice D – Tabela com valores dos S-Boxes para o Blowfish mostra a tabela com os valores dos S-Boxes do Blowfish.

2

FUNDAMENTOS MATEMÁTICOS

O objetivo deste capítulo é fornecer a base matemática para o entendimento dos cripto-sistemas de chave secreta a serem discutidos nesta dissertação.

2.1 Grupos

Definição 2.1 – Um grupo $< G,* >$ é uma estrutura algébrica composta por um conjunto não vazio G munido de uma operação binária $*: G \times G \to G$, com as seguintes propriedades:

G1. Fechamento: $\forall a,b \in G, a*b \in G$

G2. Associatividade: $(a*b)*c = a*(b*c) = a*b*c,\ \forall a,b,c \in G$

G3. Elemento identidade e: $\exists\, e \in G \quad a*e = e*a = a,\ \forall a \in G$

G4. Elementos inverso: para todo elemento $a \in G \quad \exists\, a' \in G$ tal que $a*a' = a'*a = e$. O elemento a' é denominado o elemento inverso de a e é denotado por a^{-1}.

Quando no grupo $\langle G,* \rangle$ a operação * obedece a lei de comutatividade:

G5. Comutatividade: $a*b = b*a, \forall a,b \in G$

diz-se que o grupo $\langle G,* \rangle$ é um **grupo comutativo** ou **abeliano** (em honra a NIELS HENRIK ABEL, matemático norueguês do século XIX).

Definição 2.2 – A ordem de um grupo $\langle G,* \rangle$ é a cardinalidade do conjunto *G*, denotado por $|G|$. Quando a ordem de *G* é finita, o grupo é dito finito.

Exemplo 2.1 - Os seguintes grupos são abelianos de ordem infinita $|G| = \infty$: $< Z,+ >$ grupo aditivo dos inteiros, $\langle Q,+ \rangle$ grupo aditivo dos racionais, $\langle R,+ \rangle$ grupo aditivo dos reais. Em cada caso, o elemento identidade é sempre o 0 e o inverso de a é $-a$.

Exemplo 2.2 – Os seguintes grupos são grupos abelianos multiplicativos de ordem infinita $|G| = \infty$: $< Q^*,\cdot >$, $< R^*,\cdot >$ e $< C^*,\cdot >$. O elemento identidade é 1, o inverso de a é $\frac{1}{a}$.

Definição 2.3 – Um grupo *G* é cíclico se há um elemento $a \in G$ tal que para cada $b \in G$ há um inteiro i com $b = a^i$. Tal elemento a é chamado um gerador de *G*.

2.2 Subgrupos

Definição 2.4 – Sejam $\langle G,* \rangle$ um grupo e *H* um subconjunto não vazio de *G*. O par $\langle H,* \rangle$ diz-se um subgrupo do grupo $\langle G,* \rangle$, se e somente se, $\langle H,* \rangle$ também é um grupo.

Se H é um subgrupo de G, onde $H \neq G$ e $H \neq \{e\}$, então H é chamado um subgrupo próprio de *G*.

Teorema 2.1 - Para que o grupo $\langle H,* \rangle$ seja um subgrupo do grupo $< \mathrm{G},* >$, este tem que satisfazer as seguintes condições:

(a) H é um subconjunto não vazio de G.

(b) Se $a,b \in H$, então $a * b \in H$ (Fechamento)

(c) Se $a \in H$, então $a^{-1} \in H$. (Elemento Inverso).

Exemplo 2.3 – O grupo dos inteiros com a operação de adição é um subgrupo do grupo dos números reais com adição.

Exemplo 2.4 – Com a operação de multiplicação, $\{1,-1\}$ é um subgrupo do grupo dos números reais diferentes de zero.

Definição 2.5 – A ordem de um elemento $g \in G$ é o menor número natural k tal que $g^k = e$ (se tal k existe). Se tal k não existe, então a ordem de g é $\propto$

Teorema 2.2 (Teorema de Lagrange) - Se o grupo $< H,* >$ é um subgrupo do grupo $< G,* >$, e este é finito, então $|H|$ é um divisor de $|G|$ [LP98].

2.3 Anéis

Definição 2.6 – Um anel é uma estrutura algébrica composta por um conjunto R juntamente com duas operações binárias, + e ., chamadas adição e multiplicação, tais que:

R1. $< R,+ >$ é um grupo abeliano com identidade denotada por 0;

R2. O produto $r \cdot s$ de qualquer dois elementos $r,s \in R$ esteja em R e a multiplicação seja associativa;

R3. Para todo $r,s,t \in R : r \cdot (s+t) = r \cdot s + r \cdot t$ e $(r+s) \cdot t = r \cdot t + s \cdot t$ (distributividade);

O elemento identidade em $< R,+ >$ será denotado por 0 e chamado zero, o inverso aditivo de $r \in R$ será denotado por $-r$. Um exemplo de um anel é $< Z,+,\cdot >$.

Definição 2.7 – Se em $\langle R,+,\cdot \rangle$, a multiplicação é comutativa, então o anel é dito ser comutativo.

2.4 Corpos

Definição 2.8 – Um anel comutativo $\langle F,+,\cdot\rangle$ em que o conjunto de elementos diferentes de zero forma um grupo com relação a multiplicação é chamado um corpo.

Alternativamente, um corpo pode ser definido como um domínio de integridade em que cada elemento diferente de zero tem um inverso em relação à multiplicação, pois o 0 é o elemento neutro de $\langle F,+\rangle$. Os grupos $\langle F,+\rangle$ e $\langle F^*,\cdot\rangle$ são grupos abelianos.

Uma relação entre classes de anéis é dada pela figura 2.1.

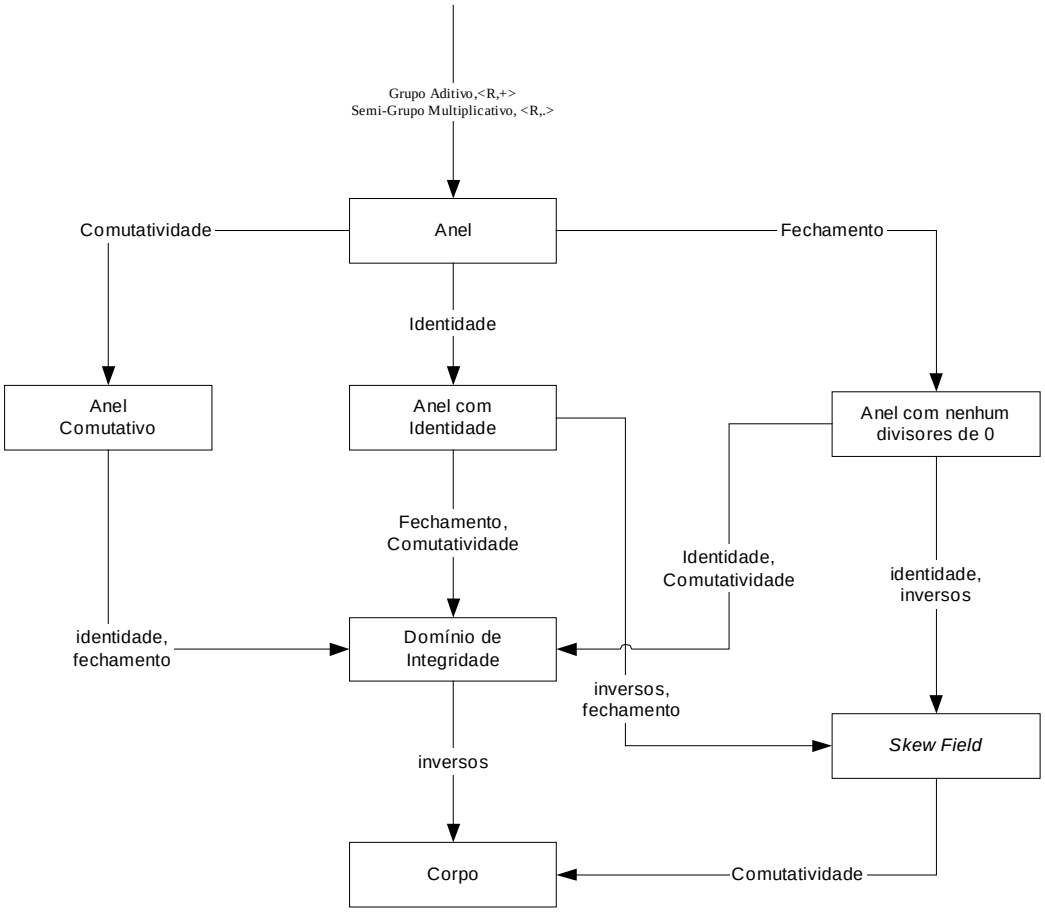

Figura 2.1 - Sistemas Algébricos derivados de Anéis

Definição 2.9 - A característica de F, denotada por *char F*, é o menor número natural *k* tal que $kr = r + \mathbf{L} + r$ (*k* vezes) seja igual a zero para todo $r \in R$. Se $k = char\ F$, todos elementos no grupo $\langle F, + \rangle$ tem uma ordem dividindo *k* [MO96].

Exemplo 2.5 – O conjunto finito $Z_n = \{0,1,\mathbf{K}\, n-1\}$ é um corpo sob as operações usuais de adição e multiplicação modulo *n*, se e somente se *n* é um número primo. Se *n* é primo, então Z_n tem característica *n*.

Definição 2.10 – Um subconjunto *K* de um corpo *F* é um subcorpo de *F* se *K* é um corpo com relação às operações em *F*.

2.5 Corpos Finitos

Definição 2.11 – Um corpo finito é um corpo que contém um número finito de elementos. A ordem de F é o número de elementos em F.

Corpos Finitos foram estudados por Evariste Galois e são conhecidos como campo de Galois. O campo de Galois de ordem *q* é usualmente denotado por *GF(q)*.

Teorema 2.3 – Os inteiros {0, 1, 2, ..., *p*-1}, onde *p* é um primo, formam o corpo *GF(p)* sob adição e multiplicação modulo *p* [WCK95].

É possível mostrar que, se **F** é um corpo finito, então **F** contém p^m elementos para algum primo *p* e inteiro $m \geq 1$. Além disso, para toda potência de primo de ordem p^m, há um único corpo finito de ordem p^m [MO96].

Definição 2.12 – Seja *b* um elemento em *GF(q)*. A ordem de β (*ord*(β)) é o menor inteiro positivo *m* tal que $b^m = 1$ [WCK95].

Teorema 2.4 – Se $t = ord(b)$ para algum $b \in GF(q)$, então $t \mid (q-1)$ [WCK95].

Este teorema determina as possíveis ordens que um elemento de um corpo finito pode assumir. Por exemplo, em *GF(16)* os elementos podem apenas ter ordem $\{1, 3, 5, 15\}$. Dada uma ordem válida, é então possível determinar exatamente quantos elementos no corpo finito tem aquela ordem.

Teorema 2.5 – Sejam α e β elementos em *GF(q)* tal que $b = a^i$. Se a $ord(a) = t$, então $ord(b) = t / mdc(i,t)$ [WCK95].

Teorema 2.6 – Com a estrutura multiplicativa do campo de Galois pode-se fazer as seguintes considerações [WCK95]:

1. Se *t* não divide $(q-1)$, então não há elementos de ordem *t* em *GF(q)*.
2. Se $t \mid (q-1)$, então há $f(t)$ elementos de ordem *t* em *GF(q)*, onde representa a $f(t)$ função de Euler.

Definição 2.13 – Um elemento primitivo em *GF(q)* é um elemento cuja ordem é $(q-1)$.

Corolário 2.1 – Em todo corpo finito *GF(q)* há exatamente $f(q-1)$ elementos primitivos.

Todos os elementos não nulos em *GF(q)* podem ser representados com $(q-1)$ potências consecutivas de um elemento primitivo α.

Exemplo 2.6 – A estrutura multiplicativa de *GF(7)* pode ser vista da seguinte maneira:

Como 7 é primo, podemos construir *GF(7)* usando os inteiros {0, 1, 2, 3, 4, 5, 6} e as operações de adição e multiplicação modulo 7. A tabela 2.1 mostra os resultados da multiplicação.

Tabela 2.1 – Operação Multiplicação de *GF(7)*

·	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
1	**0**	**1**	**2**	**3**	**4**	**5**	**6**
2	**0**	**2**	**4**	**6**	**1**	**3**	**5**
3	**0**	**3**	**6**	**2**	**5**	**1**	**4**
4	**0**	**4**	**1**	**5**	**2**	**6**	**3**
5	**0**	**5**	**3**	**1**	**6**	**4**	**2**
6	**0**	**6**	**5**	**4**	**3**	**2**	**1**

Pelo corolário 2.1 há $f(6) = 2$ elementos primitivos em *GF(7)*. Estes são 3 e 5. Considerando as potências consecutivas de 5:

$$5^1 = 5, \quad 5^2 = 4, \quad 5^3 = 6, \quad 5^4 = 2, \quad 5^5 = 3, \quad 5^6 = 1.$$

Todos os elementos do corpo são escritos como potências de 5. Usando o teorema 2.5, as ordens dos elementos de *GF(7)* são as mostradas na tabela 2.2.

Tabela 2.2 - Ordens de elementos em GF(7)

i	5^i	$ord(5^i) = 6 / mdc(i,6)$
0	1	1
1	5	6
2	4	3
3	6	2
4	2	3
5	3	6

Completamos o exemplo ao notar que, como esperado, as ordens dos vários elementos são divisores de 6 e ocorrem com a freqüência especificada, conforme tabela 2.3.

Tabela 2.3 - Elementos não nulos em *GF(7)* e suas ordens

Ordem i	**Elementos em GF(7) com ordem *i***	f(***i***)
1	{1}	1
2	{6}	1
3	{2, 4}	2
4	{ }: 4 não divide 6	-
5	{ }: 5 não divide 6	-
6	{3, 5}	2/6 elementos diferentes de zero

Definição 2.14 – A característica de um campo de Galois GF(q) é o menor inteiro positivo p tal que $p(1) = 0$. Onde $p(1)$ refere-se ao somatório de p 1's.

Teorema 2.7 – A característica de um campo de Galois é sempre um inteiro primo [WCK95].

2.5.1 Construindo Corpos Finitos

Toda a descrição desta seção é baseada em [RM87].

Definição 2.15 – Um domínio de Integridade D é um domínio Euclidiano se há uma função $s : (D - \{0\}) \rightarrow \mathrm{N}$ com a propriedade que dado qualquer $a, b \in D, \quad b \neq 0$, exista $q, r \in D$ satisfazendo

$$a = qb + r$$

onde $r = 0 \quad ou \quad sr < sb$.

Dado um elemento $m \in D$ (domínio euclidiano), não necessariamente primo, e definir uma relação de equivalência "~": $a \sim b$, se e somente se $m \mid (a - b)$. Esta relação de equivalência será representada pela notação de Gauss:

$$a \equiv b \pmod{m} \qquad \text{se} \qquad m \mid (a-b) \tag{2.1}$$

Esta relação de equivalência, como qualquer relação de equivalência, divide o conjunto (no caso D) em um certo número de subconjuntos disjuntos chamados "Classes de Equivalência". Se $a \in D$, denotaremos a única classe de equivalência contendo a por $\bar{a}$.

Agora, vamos definir adição e multiplicação (mod m) nas classes de equivalência. Para adição a definição natural é esta:

$$\bar{a} + \bar{b} = \overline{a+b} \tag{2.2}$$

Similarmente, definimos a multiplicação como segue:

$$\bar{a} \cdot \bar{b} = \overline{(a \cdot b)} \tag{2.3}$$

É fácil ver que o conjunto de classes de equivalência forma um anel com relação a estas aritméticas. A identidade na adição é dada por:

$$\bar{0} = \{x \in D : x \equiv 0 \pmod{m}\};$$

e para multiplicação:

$$\bar{1} = \{x \in D : x \equiv 1 \pmod{m}\}.$$

Este anel é denotado pelo símbolo D mod m. Por exemplo, se Z é o conjunto dos inteiros, então Z mod 4 é um anel contendo 4 elementos.

Estamos interessados em saber se D mod m é um corpo. Isto será provado pela existência do inverso na multiplicação, isto é, se para qualquer $\bar{a} \neq 0$ existe um $\bar{b}$ tal que:

$$\bar{a} \cdot \bar{b} = \bar{1} \tag{2.4}$$

Isto nem sempre é o caso. Em um corpo não deve existir o elemento zero na multiplicação. Z mod 4 contém $\bar{2} \cdot \bar{2} = \bar{0}$.

Teorema 2.8 – Se p é primo, D mod p é um corpo.

Prova: As discussões anteriores dão todas as propriedades necessárias exceto a existência do inverso. Seja $\bar{a}$ um elemento de D mod p, $\bar{a} \neq \bar{0}$; devemos mostrar que existe um elemento $\bar{b}$ tal que assegure (2.4). Agora $\bar{a} \neq 0$ significa que p não divide a. Existem

elementos b e t em D tais que $ab + pt = 1$. Então $ab \equiv 1 \pmod{p}$, e isto é equivalente a (2.4). Então $\bar{b}$ é o inverso de $\bar{a}$.

Exemplo 2.7 – Seja $D = Z$, o conjunto dos inteiros, p = 13. Então D mod p tem 13 elementos, que podemos denotar por $\bar{0}, \bar{1}, \mathbf{K}, \overline{12}$. Então, por exemplo,

$$\bar{5} \cdot \overline{10} = \overline{11}, \qquad \bar{4} \cdot \bar{7} = \bar{2}, \text{ etc.}$$

Vamos encontrar o inverso de $\bar{6}$. Aplicando o algoritmo de Euclides [RM87] para 6 e 13 para encontrar uma combinação linear de 6 e 13 que seja igual a 1. Encontramos que $6 \cdot 11 - 13 \cdot 5 = 1$. Então $\bar{6} \cdot \overline{11} = \bar{1}$, isto é, $(\bar{6})^{-1} = \overline{11}$.

Exemplo 2.8 – Dado $D = Z$, seja p um primo arbitrário. Os corpos finitos tem exatamente p elementos $\{\bar{0}, \bar{1}, \mathbf{K}, \overline{p-1}\}$. Esta construção produz infinitos corpos finitos, desde que haja infinitos primos. Mas há muitos outros corpos finitos.

Exemplo 2.9 – Seja D = $GF(2)[x]$, $p(x) = x^3 + x + 1$. Primeiro note que $p(x)$ é de fato irredutível, por que $p(0) = p(1) = 1$, então $p(x)$ não tem zeros em $GF(2)$; e uma cúbica com nenhum zero em um corpo deve ser irredutível sobre aquele corpo.

Uma vez que $p(x)$ é uma cúbica, podemos ter, como representantes das classes de equivalência (mod p), 8 polinômios da forma $a_2 x^2 + a_1 x + a_0, \quad a_i \in GF(2)$.

Para fazer a multiplicação desses 8 elementos do corpo (vetores), iniciamos pela multiplicação dos polinômios correspondentes:

$$\begin{aligned}(a_2 x^2 + a_1 x + a_0)(b_2 x^2 + b_1 x + b_0) = a_2 b_2 x^4 + (a_2 b_1 + a_1 b_2) x^3 + \\ (a_2 b_0 + a_1 b_1 + a_0 b_2) x^2 + (a_1 b_0 + a_0 b_1) x + a_0 b_0\end{aligned} \qquad (2.5)$$

Este polinômio é de grau 4, e isto não é um elemento do corpo (representa classe de equivalência). Devemos reduzir os termos grau de 3 e 4. Para fazer isto note que

$$x^3 \equiv x+1 \,(\mathrm{mod}\, x^3+x+1)$$

$$x^4 \equiv x^2+x \,(\mathrm{mod}\, x^3+x+1)$$

Então $a_2b_2x^4 \equiv a_2b_2x^2 + a_2b_2x$ e $(a_2b_1 + a_1b_2)x^3 \equiv (a_2b_1 + a_1b_2)x + (a_2b_1 + a_1b_2)$ $(\mathrm{mod}\, x^3+x+1)$. Segue que a regra de multiplicação é $[a_2, a_1, a_0]\cdot[b_2, b_1, b_0] = [c_2, c_1, c_0]$ onde:

$$\begin{aligned} c_2 &= a_2b_2 + a_2b_0 + a_0b_2 \\ c_1 &= a_2b_2 + a_2b_1 + a_1b_2 + a_1b_0 + a_0b_1 \\ c_0 &= a_2b_1 + a_1b_2 + a_0b_0 \end{aligned} \qquad (2.6)$$

Isto não é uma regra muito simples ou clara, mas pelo teorema 2.8, sabemos que produz um vetor de 3 dimensões sobre *GF*(2) em um corpo. Entretanto, há uma maneira mais simples de visualizar este corpo, o qual denotaremos por *GF*(8).

O primeiro passo é calcular as potências de *x* modulo $p(x) = x^3 + x + 1$:

$$\begin{aligned} x^0 &\equiv 1 \\ x^1 &\equiv x \\ x^2 &\equiv x^2 \\ x^3 &\equiv x+1 \\ x^4 &\equiv x^2+x \\ x^5 &\equiv x^3+x^2 \equiv x^2+x+1 \\ x^6 &\equiv x^3+x^2+x \equiv x^2+1 \\ x^7 &\equiv x^3+x \equiv 1, \end{aligned}$$

todos módulo x^3+x+1. Note que a seqüência de potências de $x(\mathrm{mod}\, p(x))$ é periódica, de período 7. Agora retornaremos ao corpo *GF*(8), e denotamos a classe de equivalência $\bar{x}$ por a. Como um vetor de 3 dimensões, $a = [0, 1, 0]$. Usando a tabela das potências de *x* modulo *p(x)*, podemos produzir a seguinte tabela de potências de *a*:

$$a^0 \equiv 1$$
$$a^1 \equiv a$$
$$a^2 \equiv a^2$$
$$a^3 \equiv a + 1$$
$$a^4 \equiv a^2 + a$$
$$a^5 \equiv a^2 + a + 1$$
$$a^6 \equiv a^2 + 1$$
$$a^7 \equiv 1,$$

As 7 primeiras potências de a são todas distintas em $GF(8)$; mas, mesmo que haja somente 7 elementos distintos em $GF(8)$, todos são potências de a. Então podemos usar a como base para "logaritmos", e concordamos que:

$$\log_a b = k \quad \rightarrow \quad a^k = b$$

Na tabela 2.4 temos a tabela de logaritmos e antilogaritmos em GF(8).

Tabela 2.4 – Logaritmos e antilogaritmos em $GF(8)$

b	$\log_a b$	K	a^k
000	*	*	000
001	0	0	001
010	1	1	010
011	3	2	100
100	2	3	011
101	6	4	110
110	4	5	111
111	5	6	101

3

INTRODUÇÃO A CRIPTOGRAFIA

O objetivo deste capítulo é fornecer uma visão geral da criptografia incluindo os conceitos básicos sobre Criptografia de Chave Secreta, Criptografia de Chave Pública, Cifragem de Bloco e bit a bit. Será dada uma descrição de algumas cifras de bloco, tais como: DES, RC5, BlowFish e IDEA.

3.1 Criptografia

Criptologia é o ramo do conhecimento que engloba criptografia e criptoanálise. A criptoanálise diferencial será estudada no capítulo 4. Segundo [DDE82], criptografia é a ciência que estuda a escrita secreta. Segundo Diffie e Hellman [DWH76], a criptografia é o estudo de sistemas "matemáticos" para resolver dois tipos de problemas de segurança: Privacidade e Autenticação.

Esta ciência faz estudos de métodos ou algoritmos que chamamos de cifra. Uma cifra é um método ou algoritmo que tem como entrada um texto claro e uma chave secreta, e gera um texto cifrado (também chamado critptograma). Este processo é conhecido como cifragem. Para obter o texto original de volta, é aplicado o processo de decifragem, que faz o reverso da cifragem, ou seja, usa a mesma chave secreta e o texto cifrado e o transforma no texto claro. A cifragem e a decifragem podem ser controlada por uma ou mais chaves criptográficas. Um criptógrafo é a pessoa que desenvolve cifras. O tipo de esquema descrito acima é denominado criptografia de Chave Secreta, pois a mesma chave é utilizada para cifrar e decifrar a mensagem, conforme mostrado na figura 3.1.

Os métodos de cifragem podem ser funções matemáticas complexas ou simplesmente manipulação de bits.

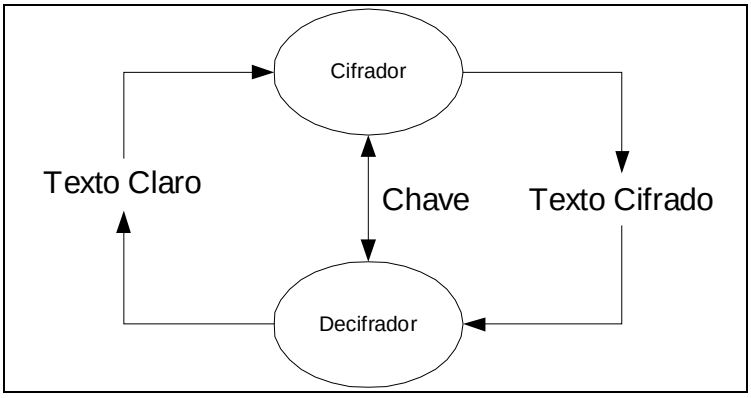

Figura 3.2 - Esquema de Cifragem e Decifragem de chave secreta**.**

Há dois tipos básicos de cifras:

- **Transposição**: rearrajam os bits ou caracteres nos dados. Com um cifra "rail-fence", por exemplo, as letras de uma mensagem de texto claro são escritas em uma forma que parece uma cerca apoiada, e então são removidas pelas linhas, como mostrado na Figura 3.2. A chave da cifra é dada pela profundidade da cerca, que neste exemplo é 3.

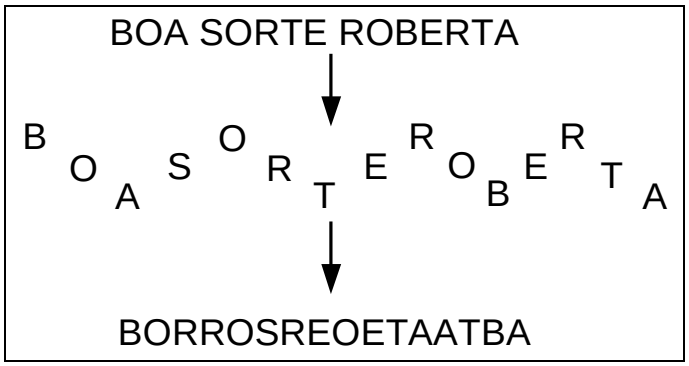

Figura 3.3 - Exemplo de Cifra de Transposição

- **Substituição**: trocam bits, caracteres ou blocos de caracteres por outros. Um tipo simples de cifra de substituição desloca cada letra do alfabeto para frente em K posições

(fazendo o ciclo de Z para A); K é a chave da cifra. A cifra é freqüentemente chamada de cifra de César porque Julio César usou-a com K = 3, como mostrado na Figura 3.3.

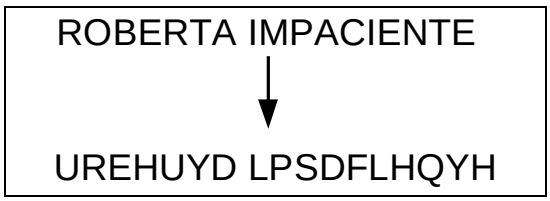

Figura 3.4 - Exemplo de Cifra por Substituição

O Data Encryption Standard (DES), por exemplo, é uma cifra de blocos de 64-bits que usa uma combinação de transposição e substituição.

As cifras podem ser classificadas quanto à segurança em:

- **Incondicionalmente segura**: não importando quanto texto cifrado é interceptado, não há informação suficiente que determine o texto claro unicamente.
- **Computacionalmente segura**: não pode ser violado por análise sistemática com recursos disponíveis, porém com recursos ilimitados o sistema sucumbirá.

A criptografia tem como objetivo garantir:

- **Sigilo**: manter o conteúdo das informações escondido de pessoas desautorizadas. Há numerosas abordagens para fornecer sigilo, desde a proteção física até algoritmos matemáticos que tornam os dados ilegíveis.
- **Integridade**: previne a modificação desautorizada dos dados. O cripto-sistema deve ter a habilidade de detectar qualquer manipulação desautorizada nos dados. Manipulação de dados podem ser inserção, exclusão e substituição.
- **Autenticação**: fornece a identificação das partes que estão se comunicando. Está dividida em duas classes: autenticação de entidade e autenticação da origem dos dados.

Um cripto-sistema tem os seguintes componentes, conforme mostrado na Figura 3.4:

1. Um espaço de texto claro, ***M***.

2. Um espaço de texto cifrado, *C*.
3. Um espaço de chave, *K*.
4. Uma família de transformações de cifragem, E_k: $M \rightarrow C$, onde k ε *K*.
5. Uma família de transformações de decifragem, D_k: $C \rightarrow M$, onde k ε *K*.

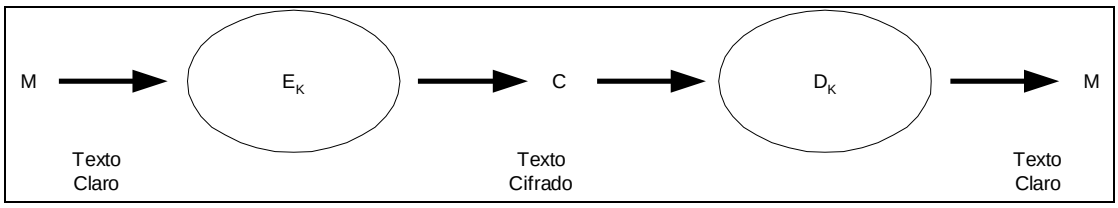

Figura 3.5 - Sistema criptográfico de chave secreta

Um cripto-sistema deve satisfazer três requisitos gerais:

1. As transformações de cifragem e de decifragem devem ser eficientes para todas chaves;
2. A segurança do sistema deve depender apenas do sigilo da chave e não dos algoritmos *E* ou *D*. Tem sido demonstrado na prática que o método de cifragem (ou decifragem) não deve ser secreto, pois, aparentemente é sempre possível descobrir o modo como a cifragem é feita. Esse é o chamado princípio de Kerckhoff.

Um cripto-sistema deve garantir o sigilo e autenticidade das mensagens. Como não conseguimos em geral, na prática, cripto-sistemas incondicionalmente seguros e tratamos de cripto-sistemas computacionalmente seguros, então definimos que, para sigilo, o cripto-sistema deverá atender aos seguintes requisitos:

1. Será computacionalmente inviável para um criptoanalista determinar sistematicamente a transformação de decifragem D_k do texto cifrado interceptado *C*, mesmo se o correspondente texto claro *M* seja conhecido.
2. Será computacionalmente inviável para um criptoanalista determinar sistematicamente o texto claro *M* a partir do texto cifrado *C* interceptado.

Para autenticidade o cripto-sistema deverá atender aos seguintes requisitos:

1. Será computacionalmente inviável para um criptoanalista determinar sistematicamente a transformação de cifragem E_k dado *C*, mesmo se o correspondente texto claro *M* seja conhecido.
2. Será computacionalmente inviável para um criptoanalista encontrar sistematicamente o texto cifrado *C'* tal que $D_k(C')$ seja texto claro válido no conjunto *M*.

Existem dois tipos mais utilizados de cripto-sistema:

- **Chave Secreta ou Simétrico**: neste tipo os dois usuários tem que trocar uma chave secreta, antes de iniciar a comunicação, que servirá para cifragem e decifragem das informações trocadas. Estes esquemas foram os descritos até este momento. Como exemplo: DES, SAFER, BlowFish, IDEA e AES, entre outros. [DDE82]
- **Chave Pública ou Assimétrico**: neste sistema temos uma chave pública e uma chave privada, e elas são diferentes no remetente e no receptor . Como exemplo: RSA e ElGamal. [DDE82]

Definição 3.1 – Em uma Cifra de Bloco, o texto claro é dividido em blocos de tamanho fixo sendo então cifrados usando uma chave, e gerando blocos de texto cifrado de mesmo tamanho.

$$c_i = f(m_i)$$

Exemplos de cifras de bloco são: DES, AES, RC5, Blowfish, SAFER.

Definição 3.2 – Se o comprimento do bloco em uma cifra de bloco é igual a 1, tem-se a chamada cifragem bit a bit.

O sistema de cifragem bit a bit mais conhecido é a cifra de Vernam. Quando a sequencia de chave da cifra é gerada aleatoriamente e não é usada novamente, esta é chamada de *one-time pad*. Esta cifra é incondicionalmente segura. [DDE82].

3.1.1 Modos de Operação

Segundo [FIP81], que regulamentou os modos de operação para o DES, existem quatro modos de operação que podem ser utilizados com qualquer cifra de bloco. Os modos de operação são os seguintes:

- Dicionário Eletrônico (Eletronic Code Book - ECB): este é o modo nativo, onde um bloco é cifrado por vez, independentemente das cifragens de blocos anteriores.
  - Cifragens: $C_i = E_k(M_i)$
  - Decifragens: $M_i = E_k(C_i)$
- Cifragem com encadeamento (Cipher Block Chaining - CBC): a cifragem do bloco depende da cifragem de blocos anteriores.
  - Cifragens: $C_i = E_k(M_i \oplus C_{i-1})$
  - Decifragens: $M_i = D_k(C_i) \oplus C_{i-1}$

  Onde C_0 é um valor inicial escolhido.
- Cifragem com Realimentação (Cipher Feedback - CFB): neste modo apenas um caracter de m bits é cifrado por vez.
  - Cifragens: $C_i = M_i \oplus MSB_m(E_k(X_i))$

    $X_{i+1} = LSB_{n-m}(X_i) \,||\, C_i$
  - Decifragens: $M_i = C_i \oplus MSB_m(E_k(X_i))$

    $X_{i+1} = LSB_{n-m}(X_i) \,||\, C_i$

onde X_1 é um valor inicial escolhido, || representa a concatenação de blocos, MSB_s e LSB_s representam o s-ésimo bit mais e menos significativo respectivamente ou equivalentemente os bits mais a direita e mais a esquerda. Aqui m é qualquer número entre 1 e o tamanho do bloco da cifra. Se o texto claro consiste de caracteres $m = 7$ ou $m = 8$ é usualmente o parâmetro bem escolhido.

- Realimentação de Saída (Output Feedback - OFB): os bits da seqüencia não são dependentes dos textos claro anteriores, isto é, somente os bits da seqüência são realimentados, não o texto cifrado como no modo CFB.

– Cifragens: $C_i = M_i \oplus MSB_m(E_k(X_i))$

$$X_{i+1} = LSB_{n-m}(X_i) \,||\, MSB_m(E_k(X_i))$$

– Decifragens: $M_i = C_i \oplus MSB_m(E_k(X_i))$

$$X_{i+1} = LSB_{n-m}(X_i) \,||\, MSB_m(E_k(X_i))$$

onde X_1 é um valor inicial escolhido.

3.1.2 Criptografia de Chave Pública

Neste tipo de criptografia, cada usuário tem uma chave pública e uma privada, e os dois usuários podem se comunicar conhecendo apenas a chave pública do outro. Cada usuário deve fazer a publicação, em um diretório público, de sua chave pública. A chave privada deve permanecer em segredo.

Se um usuário **A** deseja enviar um texto claro ***M***, com sigilo, a um outro usuário **B**, então:

- **A** usa a transformação de cifragem pública **(E_B)** de **B** e cifra a mensagem.
- Com sua transformação privada **(D_B)**, **B** decifra a mensagem e obtém o texto original (***M***), conforme figura 3.5.

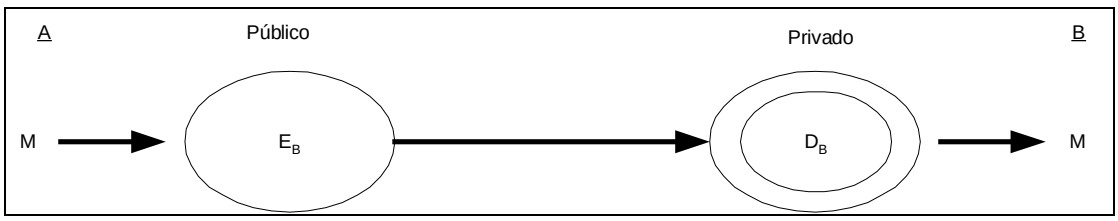

Figura 3.6 - Sigilo em um sistema de chave pública

Para obter autenticidade, as etapas devem ocorrer da seguinte maneira:

- ***M*** deve ser cifrada pela transformação privada (***D_A***) de **A**;
- Ao receber ***M***, **B** usa a transformação pública ***E_A*** de **A** para obter:

$$E_A(C) = E_A(D_A(M)) = M$$

A autenticidade é obtida porque somente **A** conhece sua transformação privada, porém qualquer usuário pode obter o texto original pela transformação pública de A, não obtendo com isso o sigilo, conforme mostrado na Figura 3.6.

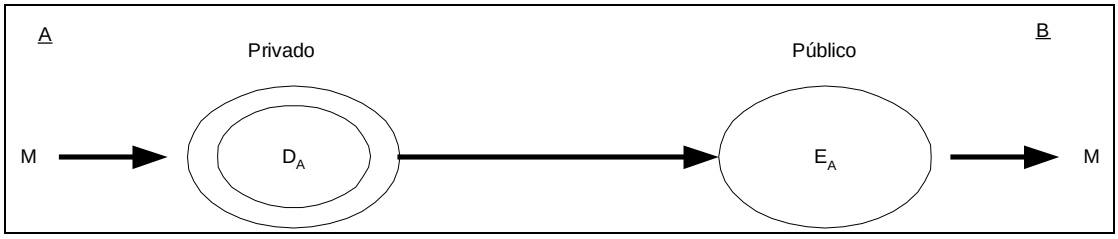

Figura 3.7 - Autenticidade em sistema de chave pública

Para conseguir sigilo e autenticidade, aplicamos o esquema mostrado na Figura 3.7, onde há uma combinação dos dois esquemas anteriores:

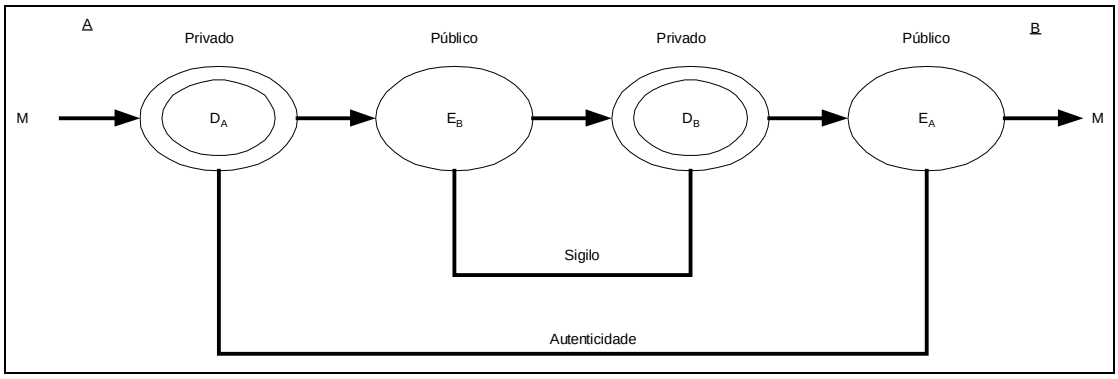

Figura 3.8 - Sigilo e autenticidade em sistema de chave pública

3.2 O Cripto-Sistema DES

No início dos anos 70, havia uma necessidade crescente de proteção das informações que trafegavam nas redes de computadores. As transações bancárias já podiam ser feitas em redes e isso pedia que, se um intruso obtivesse as mensagens enviadas, ele poderia

modificá-las e com isso alterar os dados, fraudando as operações feitas. Devido a essas necessidades, vários fabricantes começaram a produzir dispositivos criptográficos e mantinham em segredo como a criptografia era feita. Além disso, se tinha uma forma padronizada para comunicações entre dispositivos de fabricantes diferentes. Essencialmente, a padronização poderia ser de quatro tipos:

- Básico: Usado para especificar funções genéricas (serviços, métodos, resultados) requeridas para um certo conjunto de objetivos comuns.
- Interoperabilidade: especifica funções e formatos tal que os dados transmitidos por um computador possam ser recebidos por outro.
- Interface: especifica não apenas as funções e formatos de dados cruzando a interface, mas também inclui especificações físicas, elétricas e lógicas suficientes para mudar uma implementação em um lado da interface por outro.
- Implementação: especifica não apenas a interface, funções e formatos, mas também a estrutura e os métodos de implementação.

A padronização era uma necessidade evidente e poderia evitar que não houvesse comunicação entre computadores que tinham dispositivos criptográficos de fabricantes diferentes. Nesse período a NBS (*National Bureau of Standards*) iniciou um processo de coleta de algoritmos criptográficos que serviriam para ser analisados e se aprovados constituiriam o padrão para todo o governo americano. O algoritmo deveria obedecer aos seguintes critérios:

- Fornecer um alto nível de segurança;
- Completamente especificado e de fácil entendimento;
- A segurança deve residir na chave;
- Disponibilizado para todos os usuários;
- Adaptável para uso em diversas aplicações;
- A sua implementação em dispositivos eletrônicos deveria ser economicamente viável;
- Eficiente;

- Possibilidade de validação;
- Exportável;

A primeira chamada por algoritmos foi desastrosa, pois nenhum candidato se apresentou. Logo após isso, a NBS publicou nova chamada para algoritmos candidatos. Nesse momento, dentre alguns algoritmos recebidos, o algoritmo desenvolvido pelo departamento de pesquisa da IBM chamou bastante atenção. Este algoritmo era baseado na cifra chamada LUCIFER, que foi desenvolvido pelo Dr. Horst Feistel (posteriormente a estrutura principal da cifra tornou-se conhecida como rede Feistel e foi aplicada em vários outros algoritmos). Depois de a IBM e a NSA (*National Security Agency*) terem avaliado e trabalhado no re-projeto da cifra, esta se tornou padrão efetivo em Julho de 1977 e começou a ser chamada de DES (*Data Encryption Standard*). A IBM abriu mão de todos os direitos sobre algoritmo, porém os critérios de projeto não foram totalmente divulgados para o público em geral.

Com a disponibilidade ao público da cifra, o mundo acadêmico começou a trabalhar na cripto-análise do algoritmo para demonstrar possíveis falhas. Um dos grandes críticos do algoritmo foi Martin Hellman, que na época previu que a cifra seria quebrada em aproximadamente 10 anos [MH79], pois o tamanho efetivo da chave sendo de 56 bits e o crescente avanço tecnológico implicaria na quebra da cifra por busca exaustiva, em um tempo razoavelmente aceitável. Além disso, havia questionamentos sobre a possível existência de *trapdoors*, pois cogitavam que a NSA conhecia meios de fazer a decifragem das informações.

Mesmo com todas estas críticas o DES começou a ser implementado e foi decidido que a cada 5 anos este seria revisado, para que fosse validada a sua continuidade. O DES foi revisado nos anos de 1983, 1988, 1999, sendo usado nos departamentos do governo americano. Posteriormente o setor privado também adotou o algoritmo como um padrão.

3.2.1 Descrição

Esta seção é baseada no [FIP46]. O DES é uma cifra de Bloco de chave simétrica de 64 bits, que cifra blocos de texto claro de 64 bits em blocos de texto cifrado de 64 bits. Um bloco a ser cifrado é submetido a uma permutação inicial ***IP***, então passa 16 vezes por uma

função de transformação dependente da chave e, finalmente, a uma permutação que é o inverso da permutação inicial, $\boldsymbol{IP^{-1}}$. As permutações ***IP*** e $\boldsymbol{IP^{-1}}$ não tem efeito criptográfico. A computação dependente da chave é simplesmente definida em termos de uma função ***f***, chamada função da cifra, e uma função ***KS***, chamada programação da chave (*key schedule*).

3.2.2 Processo de Cifragem

A figura 3.8 mostra o processo de cifragem do DES, conforme o [FIP46]. Os 64 bits do bloco de entrada são primeiro submetidos a uma permutação inicial conforme mostrado na Tabela 3.1. Esta permutação não tem efeito criptográfico e sua implementação em software é difícil, porém é muito mais fácil em hardware. A permutação ocorre da seguinte forma: o primeiro bit é o bit 58 da entrada, o segundo é o 50, e assim por diante.

Tabela 3.1- Permutação Inicial (*IP*).

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

Após passar por 16 etapas de uma função dependente da chave (a função ***f***), o bloco é submetido a uma permutação final $\boldsymbol{IP^{-1}}$ da forma mostrada na Tabela 3.2. Tal como a permutação inicial, esta também não tem efeito criptográfico. A permutação ocorre da seguinte maneira: o primeiro bit é o bit 40 da entrada, o segundo é o 8, e assim por diante.
A computação que usa o bloco de entrada permutado como sua entrada e produz o bloco de pré-saída consiste de 16 iterações de um cálculo que é descrito abaixo em termos da função da cifra ***f***, que opera em dois blocos, um de 32 bits e um de 48 bits, e produz um bloco de 32 bits.

O bloco de entrada de 64 bits à uma interação é composto por um bloco de 32 bits ***L*** seguido por um 32 bits ***R***. O bloco de entrada é ***LR***.

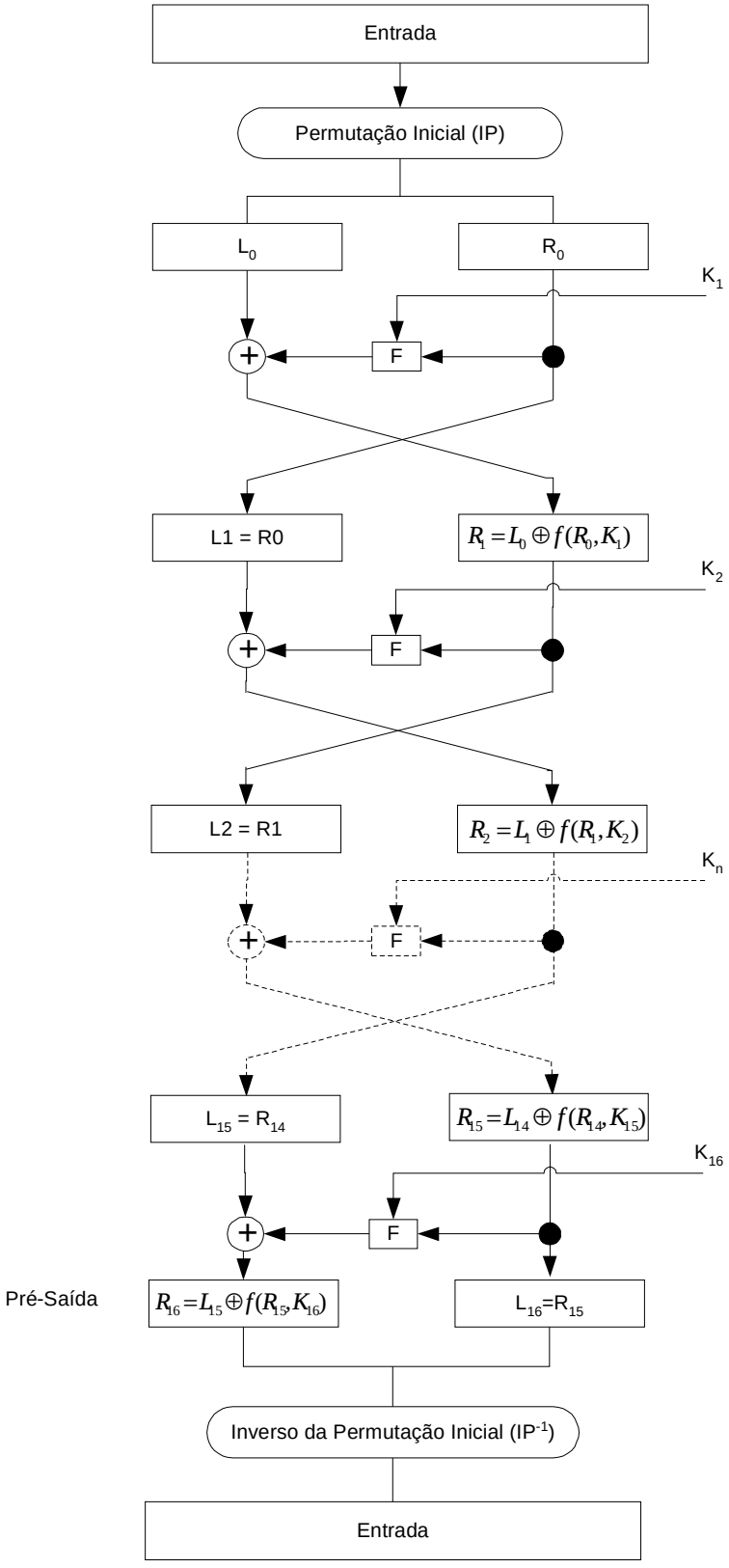

Figura 3.9 – DES

Tabela 3.2 - Permutação final (IP^{-1})

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

Seja ***K*** um bloco de 48 bits escolhidos da chave de 64 bits. Então a saída ***L'R'*** de uma iteração com entrada ***LR*** é definida por:

$$L' = R \qquad (3.1)$$
$$R' = L \oplus f(R,K)$$

onde $\oplus$ denota adição bit-a-bit módulo 2.

A entrada da primeira interação do cálculo é o bloco de entrada permutado. Se ***L'R'*** é a saída da 16ª interação então ***R'L'*** é o bloco pré-saída. Em cada iteração um bloco diferente ***K***, da chave, é escolhido da chave de 64 bits chamada ***KEY***.

Seja ***KS*** uma função que tem um inteiro *n* na faixa de 1 a 16 e o bloco ***KEY*** de 64 bits como entrada e produz como saída um bloco $\boldsymbol{K_n}$ de 48 bits, que é uma seleção permutada de bits de ***KEY***, isto é:

$$K_n = KS(n, KEY) \qquad (3.2)$$

sendo $\boldsymbol{K_n}$ determinado pelos bits em 48 diferentes posições dos bits de ***KEY***. ***KS*** é chamado programação da chave (*key schedule*) porque o bloco ***K*** usado nas iterações de (3.1) são os blocos $\boldsymbol{K_n}$ determinados por (3.2).

Para uma representação algébrica das 16 iterações executadas pelo DES podemos escrever a seguinte equação:

$$L_n = R_{n-1} \qquad (3.3)$$
$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

onde *n* está na faixa de 1 a 16 e $\boldsymbol{K_n}$ representa as 16 subchaves que o algoritmo necessita na saída da programação da chave (*key schedule*). O bloco de pré-saída é o $\boldsymbol{R_{16}L_{16}}$.

A estrutura da operação de cifragem é exatamente espelhada pela operação de decifragem, a principal deferença é que o key schedule em vez de gerar as subchaves na ordem $K_1, K_2, K_3, \mathbf{K}, K_{16}$ gera na ordem $K_{16}, K_{15}, K_{14}, \mathbf{K}, K_1$.

O cálculo da função da cifra ***f*** é mostrado na figura 3.9. Sendo ***E*** denotando uma função que usa um bloco de 32 bits como entrada e produz um bloco de 48 bits na saída. Seja ***E*** tal que os 48 bits de sua saída, escritos com 8 blocos de 6 bits cada, sejam obtidos pela seleção dos bits em sua entrada de acordo com a tabela 3.3. Sendo que os primeiros três bits de *E(R)* são dos bits na posição 32 1 e 2 de R enquanto os últimos 2 bits de *E(R)* são os bits nas posições 32 e 1.

Tabela 3.3 - Permutação de expansão (*E)* de metade do bloco (expande de 32 para 48 bits)

32	1	2	3	4	5	4	5	6	7	8	9
8	9	l0	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

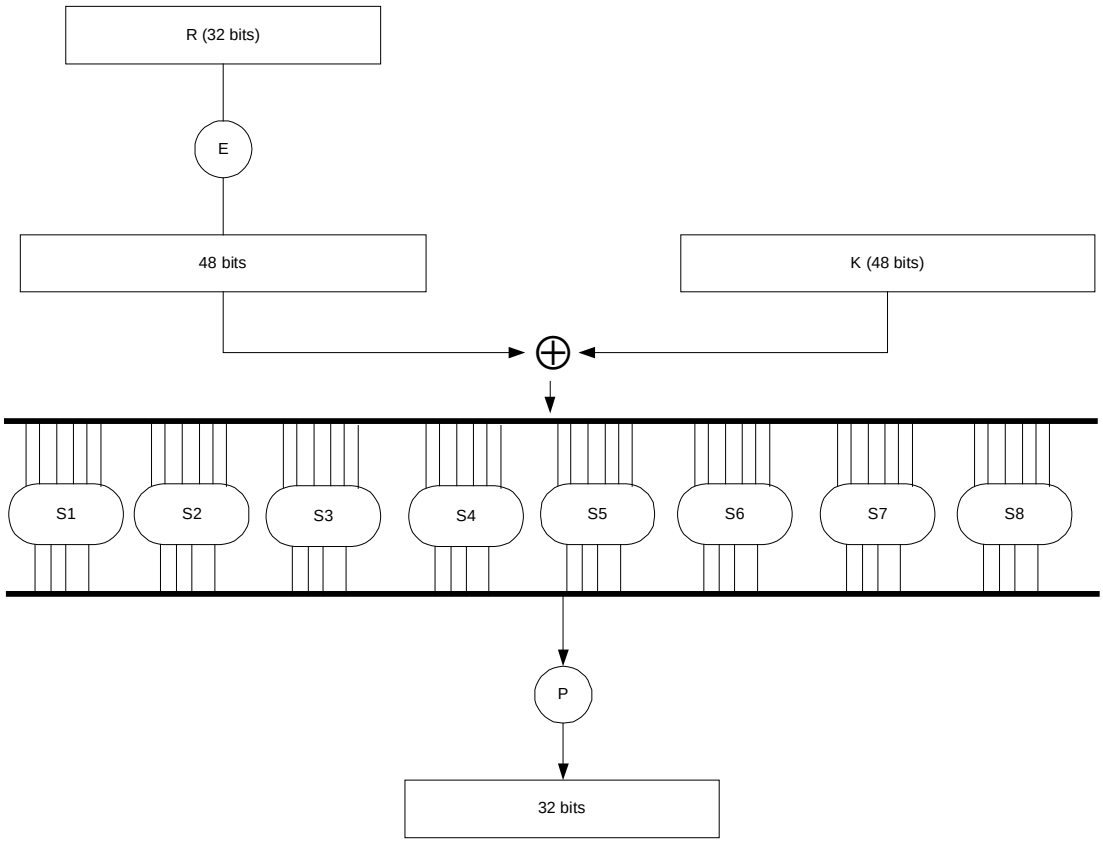

Figura 3.10- Função da cifra f

Cada uma das funções de seleção $S_1, S_2, \mathbf{K}, S_8$ tem um bloco de 6 bits com entrada e produz um bloco de 4 bits como saída. Estas funções são representadas pelas tabelas 3.4 a 3.11, respectivamente S1 a S8.

Tabela 3.4 - S1 - Box

14	4	13	1	2	15	11	8	3	l0	6	12	5	9	0	7
0	15	7	4	14	2	13	1	l0	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	l0	5	0
15	12	8	2	4	9	1	7	5	11	3	14	l0	0	6	13

Tabela 3.5 - S2 – Box

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tabela 3.6 – S3 – Box

l0	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tabela 3.7 - S4 – Box

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tabela 3.8 – S5 – Box

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	l0	4	5	3

Tabela 3.9 - S6 - Box

12	l	l0	15	9	2	6	8	0	13	3	4	14	7	5	Ll
l0	15	4	2	7	12	9	5	6	l	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	l	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tabela 3.10 - S7 - Box

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	L
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tabela 3.11 - S8 - Box

13	2	8	4	6	15	11	1	l0	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

A permutação P produz uma saída de 32 bits de uma entrada de 32 bits pela permutação dos bits do bloco de entrada. Tal função é definida pela tabela 3.12.

Tabela 3.12 - Permutação (P-Box) (sobre 32 bits)

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

Podemos representar uma etapa do DES pela figura 3.10, onde mostramos todas as atividades que são realizadas pelo algoritmo.

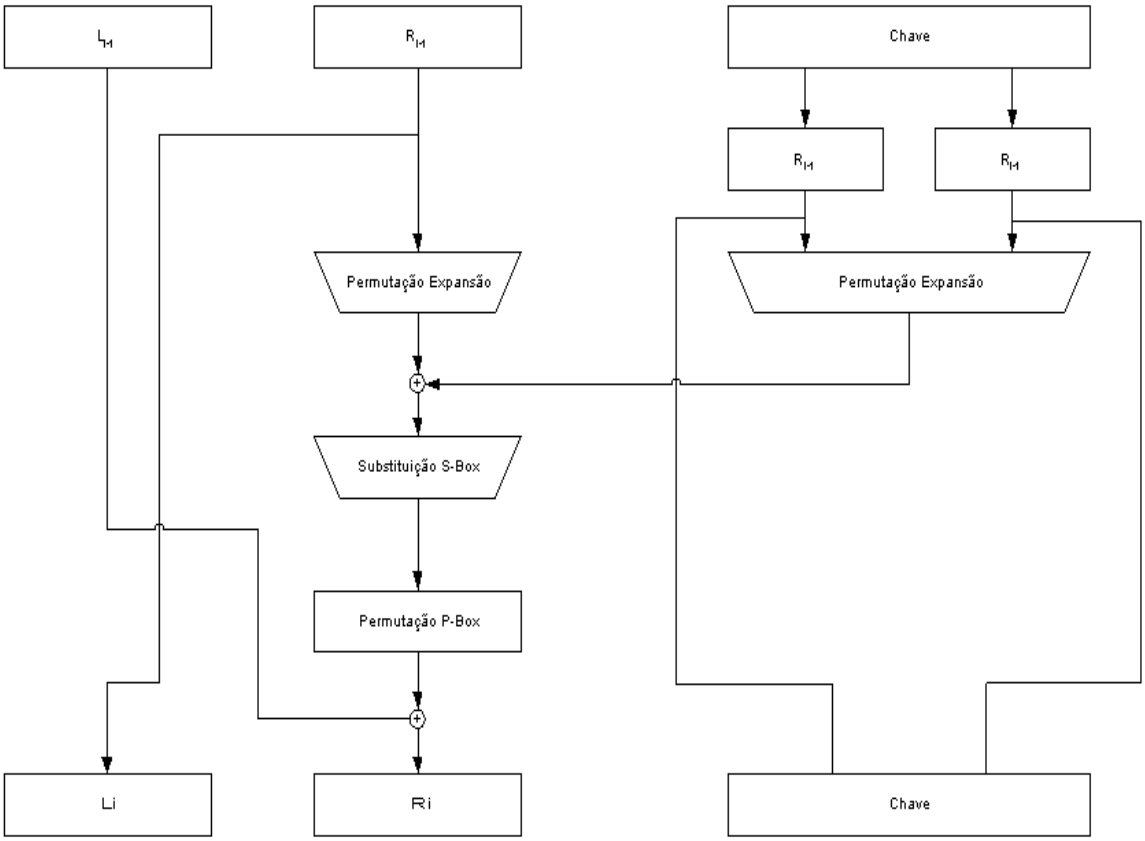

Figura 3.11- Uma Etapa do DES

O cálculo de KS é ilustrado pela figura 3.11. Um bit em cada 8 bits da chave pode ser usado para detecção de erro na geração, distribuição e armazenamento da chave. Os bits 8, 16, 24, ..., 64 são para assegurar que cada byte tenha paridade impar.

A permutação (PC-1) é determinada pela tabela 3.13. A tabela foi dividida em duas partes, com a primeira parte determinando como os bits C() são escolhidos, e a segunda parte

determinando como os bits de D() são escolhidos. Os bits da chave são numerados de 1 até 64. Os bits de C() são respectivamente os bits 57, 49, 41, ..., 44 e 36 da chave, com os bits de D() sendo os bits 63, 55, 47, ..., 12 e 4 da chave.

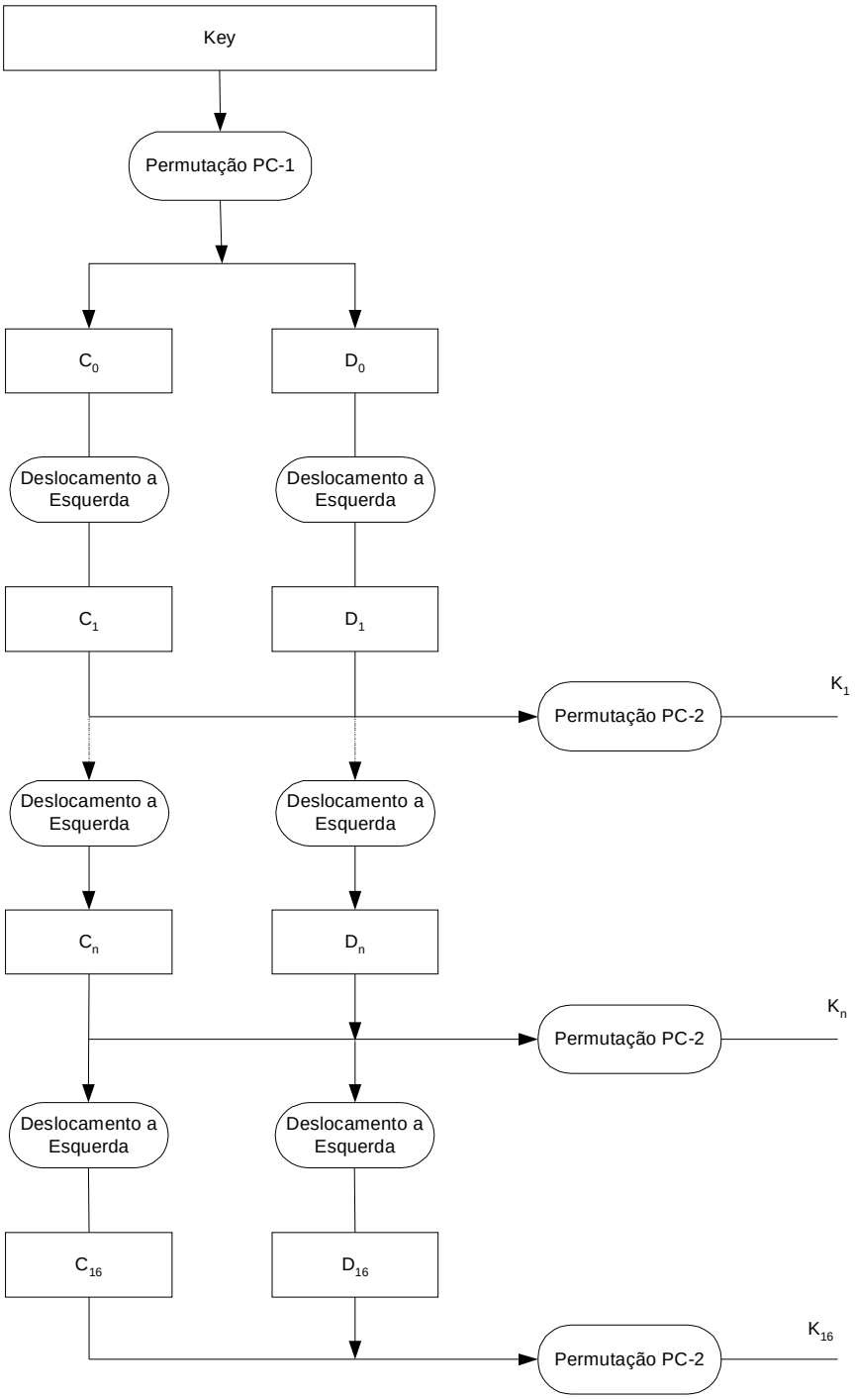

Figura 3.12 - Geração das Subchaves

Tabela 3.13 - Permutação de compressão da chave (PC-1)

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

Com C() e D() definidos, podemos agora definir como os blocos Cn e Dn são obtidos dos blocos C_{n-1} e D_{n-1}, respectivamente, n = 1, 2, 3, ..., 16. Isto é feito conforme a tabela 3.14 de deslocamento conforme a rodada em que se encontra o algoritmo.

A permutação de compressão da subchave é determinada conforme a tabela 3.15. Com isso, o primeiro bit de K_n é o 14º bit de C_nD_n, o segundo é o 17º, e assim por diante, sendo o 47º bit é o 29º e 48º bit é o 32º.

Tabela 3.14 - Deslocamentos da chave por rodada (cifragem: para esquerda; decifagrem: para direita)

Rodada	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Desloc.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabela 3.15 - Permutação de compressão da subchave (PC-2)

14	17	11	24	1	5	3	28	15	6	21	l0
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

3.2.3 O efeito do algoritmo nos dados

O efeito da mudança de um bit no texto claro seria, idealmente, o de mudar o valor de cada bit no texto cifrado com propabilidade de 1/2. Davies e Price em [DDW89], mostraram os efeitos dessas mudanças e, conforme as tabelas 3.16 e 3.17, podemos verificar o efeito da mudança de um bit no bloco de texto claro na entrada do algoritmo e na chave de cifragem, respectivamente. Para confirmar a suposição, eles calcularam a distância de Hamming, que mostra a quantidade de bits mudados entre duas palavras.

Tabela 3.16 - Análise da dependência do texto claro para o texto cifrado

Chave = 0 1 2 3 4 5 6 7 8 9 A B C D E F

Núm.	Texto Claro	Texto cifrado	Distância de Hamming
1	A B C D E F A B C D E F A B C D	C D E 8 7 2 D 4 A 4 7 1 3 4 6 F	---
2	8 B C D E F A B C D E F A B C D	C D 3 D 0 A A 4 C 4 0 2 4 B 4 A	29
3	A 9 C D E F A B C D E F A B C D	8 0 1 F 8 A 2 9 6 8 B C 4 4 7 3	38
4	A B D D E F A B C D E F A B C D	5 D 9 8 C 4 7 D D D B A 6 F 3 0	36
5	A B C F E F A B C D E F A B C D	9 9 8 9 5 6 2 A 8 4 F 4 0 1 C 9	26
6	A B C D 6 F A B C D E F A B C D	6 7 C 2 6 9 F 2 5 4 2 7 9 1 F 9	30
7	A B C D E B A B C D E F A B C D	F 8 C 9 8 F 7 9 A D C 0 6 E A 4	33
8	A B C D E F D B C D E F A B C D	8 7 D 3 2 4 0 A B B F 4 4 0 7 4	34
9	A B C D E F A 9 C D E F A B C D	D B 9 9 8 B 6 7 0 4 6 C D C E 7	30
10	A B C D E F A B E D E F A B C D	2 F 6 E 5 4 7 0 E 4 E 3 5 1 A C	25
11	A B C D E F A B C C E F A B C D	B 5 3 E 4 2 D E 3 0 F 9 7 A D 0	29
12	A B C D E F A B C D 6 F A B C D	4 F 4 0 6 7 7 2 6 B 3 5 B 0 1 4	28
13	A B C D E F A B C D E 7 A B C D	A B 1 5 5 2 8 9 6 6 0 C 6 0 B 2	35
14	A B C D E F A B C D E F 2 B C D	5 B D A 9 3 F 7 D 4 2 7 B 8 D 2	30
15	A B C D E F A B C D E F A F C D	9 8 5 3 C 5 1 1 E D 5 6 8 8 7 E	34
16	A B C D E F A B C D E F A B D D	7 0 A A 2 4 0 7 9 5 9 F 0 4 B 1	34
17	A B C D E F A B C D E F A B C 5	8 9 2 B E C 4 7 C 9 7 1 2 B E 3	26
			Média: 31,06

Tabela 3.17 - Análise da dependência da chave para o texto cifrado

Texto claro = A B C D E F A B C D E F A B C D

Núm.	Chave	Texto cifrado	Distância de Hamming
1	0 1 2 3 4 5 6 7 8 9 A B C D E F	C D E 8 7 2 D 4 A 4 7 1 3 4 6 F	---
2	8 0 2 3 4 5 6 7 8 9 A B C D E F	1 B 7 3 F E 8 B C 0 B 8 8 6 0 6	35
3	0 2 2 3 4 5 6 7 8 9 A B C D E F	0 F 9 2 F 6 0 D 2 F D 4 D 8 B 7	32
4	0 1 6 2 4 5 6 7 8 9 A B C D E F	3 1 A F D 8 C 5 4 F B F 4 B C D	37
5	0 1 2 6 4 5 6 7 8 9 A B C D E F	C 7 9 F 5 9 6 3 D 4 6 5 A 7 E E	29
6	0 1 2 3 0 4 6 7 8 9 A B C D E F	3 6 5 2 9 C C 1 0 7 1 7 A 3 8 9	39
7	0 1 2 3 4 6 6 7 8 9 A B C D E F	7 F 3 5 F 7 E 6 C E C 5 7 E E 3	30
8	0 1 2 3 4 5 4 6 8 9 A B C D E F	C 9 F 3 F D 9 2 6 0 C 6 8 1 8 A	27
9	0 1 2 3 4 5 6 4 8 9 A B C D E F	E 6 9 2 8 3 2 2 E E 8 B 9 A 6 9	36
10	0 1 2 3 4 5 6 7 A 8 A B C D E F	0 9 9 7 A 5 A F 6 E 4 E 1 4 6 0	37
11	0 1 2 3 4 5 6 7 8 A A B C D E F	C 7 B 4 1 C 4 F 3 8 D 9 A F 7 A	31
12	0 1 2 3 4 5 6 7 8 9 E A C D E F	4 B 4 A A 2 0 A B 2 1 4 4 D D D	30
13	0 1 2 3 4 5 6 7 8 9 A 8 C D E F	1 2 9 9 7 E E 8 0 0 1 B D 2 7 C	32
14	0 1 2 3 4 5 6 7 8 9 A B 4 C E F	4 2 D 1 7 B 7 D F 5 3 4 3 B 7 9	28
15	0 1 2 3 4 5 6 7 8 9 A B C E E F	8 7 F 6 4 9 3 C 4 C 8 9 8 3 2 7	33
16	0 1 2 3 4 5 6 7 8 9 A B C D 6 E	F C 3 0 F 6 F 7 6 B D 4 2 5 9 2	31
17	0 1 2 3 4 5 6 7 8 9 A B C D E C	1 C B D E C 4 B 7 9 B C A 7 A 1	39
			Média: 32,88

3.2.4 Regularidades conhecidas do DES

3.2.4.1 Complementação

Segundo [DDW89], uma característica notável no DES é a propriedade de complementação. Se o complemento de um bloco de texto claro é cifrado com o complemento de uma chave, então o resultado da cifragem do DES com estes valores é o complemento do texto cifrado original.

Então se

$$y = E_k(x),$$

então

$$\overline{y} = E_{\overline{k}}(\overline{x})$$

Este efeito é devido a presença de duas operações Ou-Exclusivo, um dos quais precede os S-Box no fluxo lógico do algoritmo e o outro após a permutação P.

A propriedade de complementação, que Davies e Price [DDW89] observaram, não representa uma séria fraqueza na segurança do algoritmo, pois não há redução significativa no tempo de uma procura exaustiva usando pares de texto claro - texto cifrado.

3.2.4.2 As chaves fracas

Uma regularidade adicional na operação do DES é devido a maneira em que os bits são selecionados da chave de cifragem para formar as subchaves individuais, K1 a K16. Claramente, se a chave tiver todos os bits 1's ou todos 0's, então a saída do PC2 em cada ciclo será o mesmo para todas as rodadas. O efeito do uso dessas chaves será o mesmo para cifragem e decifragem. As chaves fracas do DES são mostradas na tabela 3.18. Assim, se um texto claro for cifrado com uma chave fraca e a seguir cifrado novamente com a mesma chave, obtêm-se no final o texto claro original.

Tabela 3.18 - Chaves fracas do DES (em hexadecimal; note-se que oito dos bits são na realidade bits de paridade)

Chave (com paridade)	Chave real
01 01 01 01 01 01 01 01	00000000 00000000
FE FE FE FE FE FE FE FE	FFFFFFFF FFFFFFFF
1F 1F 1F 1F 0E 0E 0E 0E	00000000 FFFFFFFF
E0 E0 E0 E0 F1 F1 F1 F1	FFFFFFFF 00000000

3.2.4.3 As chaves semi-fracas

Há chaves semi-fracas que ocorrem em pares. Isto que dizer que em uma cifragem com uma chave do par, seguido pela cifragem com a outra chave do par, o texto claro original é restaurado. Essa propriedade é devido a simetria do padrão de desclocamento mostrado pelas tabelas 3.14 e 3.15 (deslocamento da chave e permutação de compressão da subchave PC-2, respectivamente). A tabela 3.19 mostra os pares de chaves semi-fracas encontradas por [DDW89].

Tabela 3.19 - Chaves semi-fracas do DES

Primeira chave	Segunda chave
01 FE 01 FE 01 FE 01 FE	FE 01 FE 01 FE 01 FE 01
1F E0 1F E0 0E F1 E0 F1	E0 1F E0 1F F1 0E F1 0E
01 E0 01 E0 01 F1 01 F1	E0 01 E0 01 F1 01 F1 01
1F FE 1F FE 0E FE 0E FE	FE 1F FE 1F FE 0E FE 0E
01 1F 01 1F 01 0E 01 0E	1F 01 1F 01 0E 01 0E 01
E0 FE E0 FE F1 FE F1 FE	FE E0 FE E0 FE F1 FE F1

3.2.5 Possíveis critérios do projeto dos S-Boxes

Segundo a IBM, o projeto das caixas S levou em conta as seguintes considerações:

- Não deve existir uma função linear das entradas para as saídas

- Se duas entradas diferem em exatamente um bit, suas saídas devem diferir ao menos em dois bits
- Se duas entradas diferem em dois bits intermediários (que selecionam a coluna), suas saídas devem diferir ao menos em dois bits
- Se duas entradas diferem nos dois primeiros bits e são iguais nos dois últimos bits, suas saídas não devem ser iguais
- Os 4 bits de saída de uma caixa S devem ser distribuídos de forma que dois afetem bits intermediários e dois afetem bits das extremidades da caixa seguinte
- Os 4 bits de saída devem afetar seis caixas diferentes; não pode haver uma caixa repetida

3.3 O Cripto-Sistema RC5

O RC5 foi projetado com os seguintes objetivos:

- Ser uma cifra de bloco simétrica. A mesma chave criptográfica é usada para cifragem e decifragem. O tamanho do texto claro e do texto cifrado é uma seqüência de bits de tamanho fixo (blocos).
- Ser aceitável para *hardware* e *software*. Isto significa que RC5 usaria apenas operações computacionalmente primárias, comumente encontradas em microprocessadores típicos.
- Ser rápido. Isto implica que RC5 seja orientado à palavra: as operações básicas seriam operadores que funcionam na palavra de dados completa de uma vez.
- Ser adaptável a processadores de diferentes tamanhos de palavras. Por exemplo, como processadores de 64-bit se tornam disponíveis, seria possível para o RC5 explorar o tamanho maior de palavra. Entretanto, o número de *w* bits em uma palavra é um parâmetro do RC5; diferentes escolhas deste parâmetro resultam em diferentes algoritmos RC5.

- Ser iterativo em estrutura, com número variável de rodadas. O usuário manipularia explicitamente a troca entre maior velocidade e maior segurança. O número de rodadas *r* é um segundo parâmetro do RC5.
- Ter uma chave criptográfica de tamanho variável. O usuário escolhe o nível de segurança apropriado para sua aplicação, ou com requerido por considerações externas tal como restrições de exportação. O tamanho de chave b (em bytes) é o terceiro parâmetro do RC5.
- Ser simples. Seria fácil implementar. Mais importante, uma estrutura mais simples é talvez mais fácil de analisar e avaliar, tal que a força criptográfica do RC5 seja rapidamente determinada.
- Ter necessidade de pouca memória, tal que seja facilmente implementado em *smart cards* ou outros dispositivos com restrições de memória.
- Fornecer alta segurança quando valores de parâmetros sejam adequadamente escolhidos.
- Destacar-se pelo uso de rotações dependentes dos dados.

O RC5 insere uma nova primitiva criptográfica: rotações dependentes de dados, em que uma palavra de resultados intermediários é ciclicamente deslocada por uma quantidade determinada pelos bits de baixa ordem de outro resultado intermediário.

3.3.1 Parâmetros do RC5

O RC5 é orientado à palavra: todas operações básicas têm palavras de entrada e saída de *w* bits. O RC5 é uma cifra de bloco com tamanho de bloco de entrada (texto claro) e bloco de saída (texto cifrado) de 2 palavras. A escolha nominal para w é 32 bits, para que o RC5 tenha tamanho de bloco de 64 bits para texto claro e texto cifrado. O RC5 é bem definido para qualquer w > 0, embora para simplicidade é proposto que apenas os valores 16, 32 e 64 sejam permissíveis.

O número r de rodadas é o segundo parâmetro do RC5. Escolhendo-se um número grande de rodadas, provê-se um aumento no nível de segurança. O RC5 usa uma “tabela de

expansão da chave", S, que é derivada da chave secreta do usuário. O tamanho t de tabela S também depende do número de rodadas: S tem t = 2(r + 1) palavras. Escolhendo um grande número de rodadas implica numa necessidade maior de memória. Podemos ver uma descrição dos parâmetros na tabela 3.20.

Tabela 3.20 – Parâmetros do RC5

Parâmetro	Descrição
w	Tamanho da palavra, em bits; cada palavra contém $u = (w/8)$ bytes. O valor nominal é de 32 bits; valores permitidos são 16, 32 e 64. RC5 cifra blocos de duas palavras: blocos de texto claro e texto cifrado têm, cada um, tamanho de *2w* bits.
r	Número de rodadas. A tabela da chave expandida tem t = 2(r + 1) palavras. Valores permitidos são 0, 1, ..., 255.
b	Número de bytes da chave secreta *K*. Valores permitidos são 0, 1, ..., 255.
K	A chave secreta de *b* byte: *K*[0], *K*[1], ..., *K*[*b*-1]

O algoritmo é indicado da seguinte forma RC5-*w*/*r*/*b*. Por exemplo, RC5-32/16/10 tem palavras de 32 bits, 16 rodadas, uma chave secreta de 10 byte e uma tabela de expansão da chave de 2(16 + 1) = 34 palavras.

3.3.2 Operações Primárias do RC5

O *lg(x)* representa logaritmo de x na base 2.

O RC5 usa somente três operações primárias:

1. Adição de palavras em complemento de dois, representado por "+". Isto é adição módulo 2^w. A operação inversa, subtração, é representada por "-".

2. O Ou-exclusivo de palavras é representado por $\oplus$.
3. A rotação a esquerda de palavras: a rotação cíclica a esquerda da palavra *x* por *y* bits é representado por $x \lll y$. O *y* é interpretado em módulo *w*, tal que quando *w* é uma potência de 2, somente *lg(w)* bits de *y* de baixa ordem são usado para determinar o quantidade de rotações. A operação inversa, rotação a direita, é representada por $x \ggg y$.

No RC5 não há tabelas de substituição não-linear ou outros operadores não-lineares. A força do RC5 depende fortemente nas propriedades criptográficas das rotações dependentes de dados [RR].

3.3.3 Algoritmo

O algoritmo do RC5 consiste de 3 componentes: um algoritmo de expansão da chave, um algoritmo de cifragem e um algoritmo de decifragem.

O texto claro de entrada do RC5 consiste de palavras de 2*w* bits, que representamos por A e B. O RC5 usa uma tabela de chave expandida, S[0...t-1], consistindo de t = 2(r + 1) palavras de w bits. O algoritmo de expansão da chave inicializa S com a chave secreta do usuário pelo parâmetro K.

3.3.3.1 Cifragem

Os blocos de entrada são dados em dois registradores A e B de 2*w* bits. A expansão da chave já foi feita e portanto o vetor S[0... t-1] foi corretamente preenchido. Então o código para o algoritmo de cifragem é o seguinte:

$A = A + S[0];$

$B = B + S[1];$

for i = 1 *to r do*

$A = ((A \oplus B) <<< B) + S[2*i];$

$B = ((B \oplus A) <<< A) + S[2*i+1];$

A saída está nos registradores A e B.

3.3.3.2 Decifragem

A rotina de decifragem é facilmente derivada da rotina de cifragem:

for $i = r$ *downto* 1 *do*

$B = ((B - S[2*i+1]) >>> A) \oplus A;$

$A = ((A - S[2*i]) >>> B) \oplus B;$

$B = B - S[1];$

$A = A - S[0];$

3.3.3.3 Expansão da Chave

A rotina de expansão da chave expande a chave secreta do usuário K para preencher o vetor S, tal que S pareça um vetor de palavras binárias aleatórias de t = 2(r + 1) determinado por K. O algoritmo de expansão da chave usa duas "constantes mágicas", e consiste de partes simples.

A primeira é a definição das "constantes mágicas", onde o algoritmo usa duas palavras Pw e Qw. Elas são definidas pela seguinte forma:

$$P_w = Odd((e-2)2^w) \quad (1)$$

$$Q_w = Odd((f-1)2^w \quad (2)$$

onde

$e = 2.718281828459...$ (base de logaritmo natural)

$f = 1.618033988749...$ (relação áurea)

e onde $Odd(x)$ é o inteiro ímpar mais próximo de x. Para w = 16, 32 e 64, estas constantes são dadas abaixo em binário e em hexadecimal.

$$P16 = 1011011111100001 = b7e1$$
$$Q16 = 1001111000110111 = 9e37$$

$$P32 = 10110111111000010101000101100011 = b7e15163$$
$$Q32 = 10011110001101110111100110111001 = 9e3779b9$$

$$P64 = 1011011111100001010100010110000101000101011101101001010100110101 1$$
$$= b7e151628aed2a6b$$
$$Q64 = 1001111000110111011110011011100101111111010010100111110000010101$$
$$= 9e3779b97f4a7c15$$

A segunda parte é a conversão da chave secreta de bytes para palavras. O primeiro passo do algoritmo de expansão da chave é copiar a chave secreta K[0...b-1] em um vetor L[0...c-1] de $c = \left\lceil b/u \right\rceil$ palavras, onde $u = w/8$ é o número de bytes / palavra. Esta operação é feita da maneira natural, usando u bytes consecutivos da chave K para preencher cada palavra em L, byte de baixa ordem a byte de alta ordem. Qualquer posição não preenchida é zerada.
A terceira parte é a inicialização do vetor S. O segundo passo do algoritmo de expansão da chave é inicializar o vetor S com um valor fixo (independente da chave), usando um progressão aritmética modulo 2w determinado pela "constante mágica" Pw e Qw. Desde que Qw seja ímpar, a progressão aritmética tem período 2w.

$S[0] = P_w$;
for $i = 1$ ***to*** $t - 1$ ***do***
    $S[i] = S[i-1] + Q_w$;

O terceiro passo do algoritmo é a mistura de S e L com a chave secreta do usuário, isto é feito em três passos. Mais precisamente, devido aos diferentes tamanhos de S e L, o vetor maior será processado três vezes, e o outro pode ser manipulado mais vezes.

$i = j = 0$;

$A = B = 0$;

do 3*Max(t,c) **times**:

$A = S[i] = (S[i] + A + B) <<< 3$;

$B = L[j] = (L[j] + A + B) <<< (A + B)$;

$i = (i + 1)\ mod(t)$;

$j = (j + 1)\ mod(c)$;

3.4 O Cripto-Sistema BlowFish

O Blowfish é uma rede de Feistel, iterando uma função de cifragem 16 vezes. O comprimento do bloco de texto claro é de 64 bits, e a chave pode ser qualquer comprimento até 448 bits. Embora haja uma complexa fase de inicialização requerida antes que qualquer cifragem seja feita, esta é muito eficiente em grandes microprocessadores. Foi projetada por Bruce Schneier em 1993, sendo disponibilizada para uso do público em geral [BS94].

Trata-se de um cripto-sistema somente aceitável em aplicações onde a chave não muda frequentemente, como um enlace de comunicação ou em cifrador de arquivo de dados. É significativamente mais rápido que o DES quando implementado em microprocessadores de 32 bits com grande cache de dados, tais como Pentium e PowerPC.

O algoritmo consiste de duas partes: uma para expansão da chave e uma para cifragem de dados. A expansão da chave converte uma chave de no máximo 448 bits em diversas subchaves totalizando 4168 bytes.

3.4.1 Processo de Cifragem

O Blowfish é uma rede de Feistel consistindo de 16 rodadas, conforme a fig. 3.12. A entrada (texto claro) é uma palavra de 64 bits. O algoritmo funciona da seguinte maneira: o bloco de texto claro é dividido em duas partes de 32 bits (L_0, R_0). Cada rodada é definida recursivamente conforme a equação 3.4.

$$R_i = K_i \oplus L_{i-1} \qquad (3.4a)$$

$$L_i = R_{i-1} \oplus F(R_i) \qquad (3.4b)$$

onde i inicia com 1 e vai até 16. Podemos notar que na última rodada não há troca entre as partes esquerda (*L*) e direita (*R*). A função não-linear F efetua o processo de confusão de dados. Observe que em cada rodada o bloco de 32 bits *L* sempre é combinado com uma subchave K_i antes de servir de entrada para a função F; esta operação tem o intuito de causar confusão no texto gerado. O texto cifrado final pode ser representado da seguinte forma $C = (R_t \oplus K_{t+2}, L_t \oplus K_{t+1})$, onde t = 16 representa o número total de rodadas.

O algoritmo de inicialização do BlowFish gera 18 subchaves, das quais 16 são utilizadas nas 16 rodadas da cifra e duas na combinação final, gerando dessa forma o texto cifrado. A decifragem é exatamente igual a cifragem, exceto que as subchaves K1, K2, ... , K18 são usados na ordem reversa.

A função *F* , conforme figura 3.13, é definida conforme a equação 3.5.

$$F(L) = ((S_1(a) + S_2(b) \bmod 2^{32}) \oplus S_3(c)) + S_4(d) \bmod 2^{32} \qquad (3.5)$$

O bloco de entrada (*L*) é dividido em 4 blocos de 8 bits cada. Cada bloco é representado como (*a*, *b*, *c*, *d*) é aplicado em S-Boxes dependentes da chave, cujos valores são obtidos conforme programa de inicialização. Podemos dizer que cada S-Box é de 8x32 bits, ou seja, tem entrada de 8 bits e saída de 32 bits.

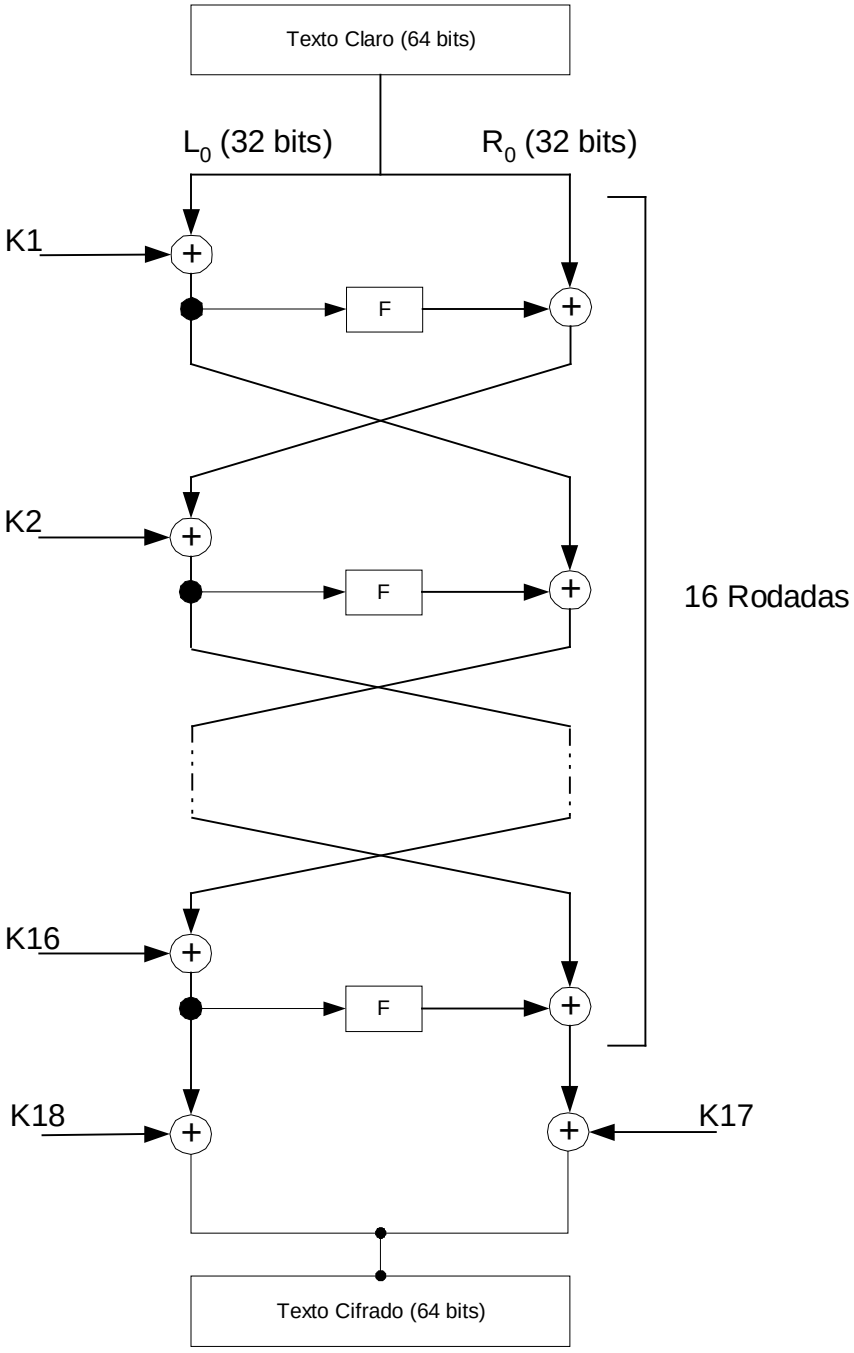

Figura 3.13 - Algoritmo Blowfish

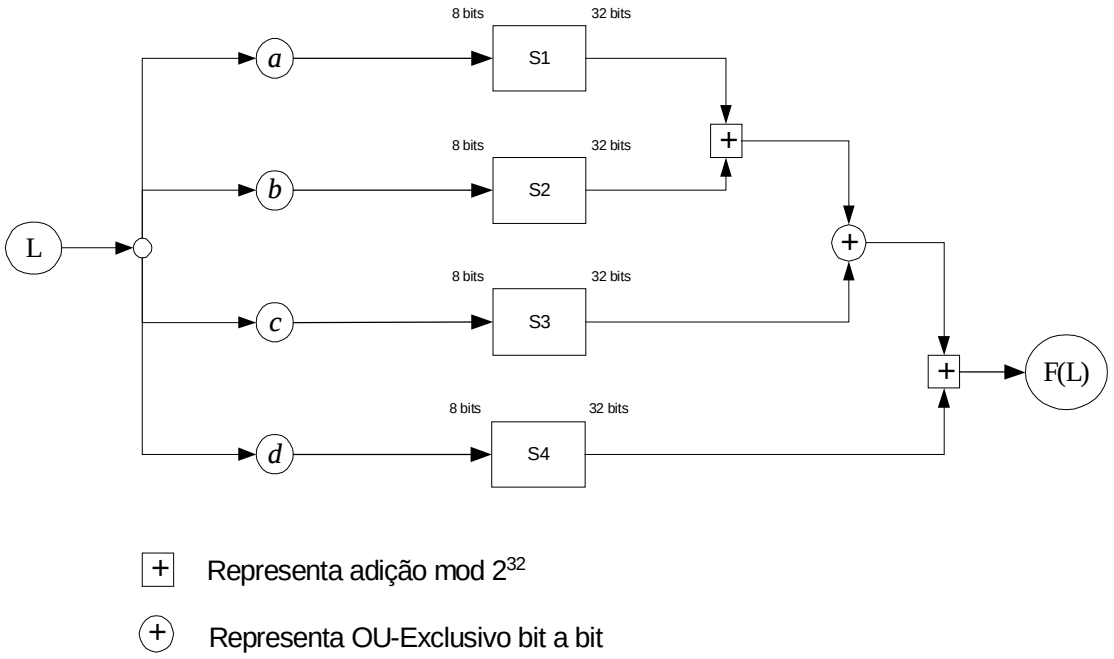

Figura 3.14 - Função *F* do Blowfish

3.4.2 Algoritmo de Inicialização

As subchaves devem ser pré-computadas antes de qualquer cifragem ou decifragem. O algoritmo é conforme a seguir:

1. O vetor P consiste de 18 subchaves de 32 bits: K1, K2, ..., K18.
2. Há 4 S-Boxes de 32 bits com 256 entradas cada:
   S1(0), S1(1), ..., S1(255);
   S2(0), S2(1), ..., S2(255);
   S3(0), S3(1), ..., S3(255);
   S4(0), S4(1), ..., S4(255);

As subchaves são calculadas usando o algoritmo blowfish. O método é o seguinte:

1. Inicialize o vetor K e os 4 S-Boxes com uma *string* fixa. Esta consiste dos dígitos *pi* em hexadecimal (com exceção do valor inicial 3). Por exemplo:
   K1 = 0x 243f6a88

K2 = 0x 85a308d3

K3 = 0x 13198a2e

K4 = 0x 03707344

2. OU-Exclusivo de K1 com os primeiros 32 bits da chave, OU-Exclusivo de K2 com o segundo bloco de 32 bits da chave, e assim por diante para todos os bits da chave (possivelmente até K14). Ciclos são repetidos nos bits da chave até que em todo vetor K seja feito o OU-Exclusivo com os bits da chave. (Para toda chave pequena, há no mínimo uma chave maior equivalente; por exemplo, se A é uma chave de 64 bits, então AA, AAA, etc., são chaves equivalentes).
3. Cifre a *string* toda zero com o algoritmo blowfish, usando as subchaves descritas nos passos (1) e (2).
4. Troque K1 e K2 com a saída do passo (3).
5. Cifre a saída do passo (3) usando o algoritmo blowfish com as subchaves modificadas.
6. Troque K3 e K4 com a saída do passo (5).
7. Continue o processo, trocando todas entradas do vetor K, e então todos os 4 S-Box, com a saída do algoritmo blowfish que é mudado continuamente.

No total são 521 iterações para gerar todas as subchaves.

3.4.3 Criptoanálise do Blowfish

John Kelsey desenvolveu um ataque que quebraria o blowfish de 3 rodadas, mas foi incapaz de estendê-lo. Este ataque explorou a função *F* e o fato que adição mod 2^{32} e XOR não comutam. Viknanjit Singh Chhabra procurou por maneiras de implementar eficientemente uma máquina de procura de chave por força bruta [BS95]. Sege Vaudenay examinou uma versão simplificada do Blowfish, com os S-Boxes conhecidos e sem dependência da chave. Para esta versão, um ataque diferencial recuperou o vetor K com 2r+1 textos claro escolhidos (r é o número de rodadas). Este ataque é impossível para blowfish de 8 rodadas ou superior [BS95].

3.5 O Cripto-Sistema IDEA

A cifra de bloco IDEA (de International Data Encryption Algorithm) foi desenvolvida por Xuejia Lai, James L. Massey e S. Murphy [XL92], como uma versão melhorada do PES (Proposed Encryption Standard). O IDEA cifra blocos de texto claro de 64 bits gerando blocos de texto cifrado também de 64 bits. O tamanho da chave secreta do usuário é de 128 bits. O IDEA foi baseado no conceito de projeto em que se "mistura" operações de diferentes grupos algébricos. A "confusão" requerida foi alcançada pelo uso de três operações de grupos incompatíveis em pares de sub-blocos de 16 bits e a sua estrutura foi escolhida para fornecer a difusão necessária. A cifra tem o propósito de aumentar a segurança contra criptoanálise diferencial.

O cripto-sistema IDEA é uma cifra de bloco iterativa consistindo de 8 rodadas seguidas por uma transformação de saída. A primeira rodada e a transformação de saída são mostradas na figura 3.14.

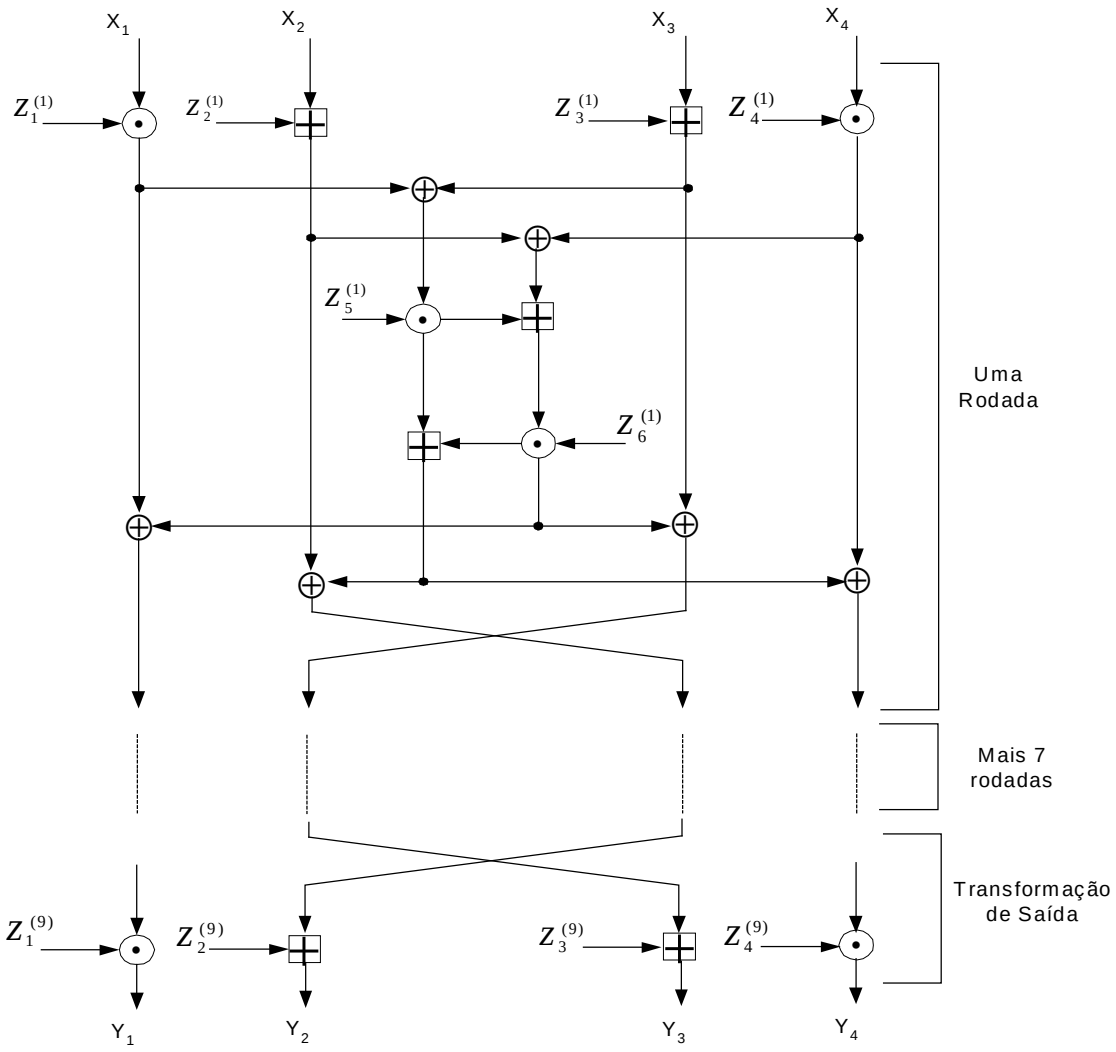

Figura 3.14 – Processo de cifragem no IDEA.

Na figura 3.14 temos a seguinte legenda:

Xi : sub-bloco de texto claro de 16 bits;

Yi : sub-bloco de texto cifrado de 16 bits;

$Z_i^{(r)}$: subchave de 16 bits;

$\oplus$: OU-Exclusivo bit a bit em sub-blocos de 16 bits;

$\boxplus$: adição módulo 2^{16} de inteiros de 16 bits;

$\odot$: multiplicação módulo $2^{16} + 1$ em inteiros de 16 bits com o sub-bloco zero correpondendo a 2^{16}.

3.5.1 Processo de Cifragem

No processo de cifragem mostrado na figura 3.14, três diferentes operações de grupo em pares de sub-blocos de 16 bits são usadas,

- Operação Ou-Exclusivo em dois sub-blocos de 16 bits, denotado por $\oplus$;
- Adição de inteiros módulo 2^{16} onde o sub-bloco de 16 bits é tratado com a representação de inteiro na base 2; a operação resultante é denotada por $\boxplus$;
- Multiplicação de inteiros $2^{16} + 1$ onde o sub-bloco de 16 bits é tratado com a representação de um inteiro na base 2, com exceção para sub-blocos com zero que é tratado como representação 2^{16}; a operação resultante é denotada por $\odot$;

Como um exemplo destas operações de grupo, note que

$$(0,\mathbf{K},0)\odot(1,0,\mathbf{K},0)=(1,0,\mathbf{K},0,1)$$

porque

$$2^{16}2^{15}\bmod(2^{16}+1)=2^{15}+1$$

O bloco de texto claro de 64 bits X é dividido em 4 sub-blocos de 16 bits X_1, X_2, X_3 e X_4, isto é, $X=(X_1,X_2,X_3,X_4)$. Os quatro sub-blocos de texto claro são transformados em 4 sub-blocos de texto cifrado de 16 bits Y_1, Y_2, Y_3 e Y_4 (isto é, $Y=(Y_1,Y_2,Y_3,Y_4)$) sob o controle de 52 subchaves de 16 bits que são geradas a partir da chave secreta do usuário de 128 bits, pelo algoritmo de programação de chave. Para $r=1,2,\mathbf{K},8$, as seis subchaves usadas na r-ésima rodada serão denotadas por $Z_1^{(r)},\mathbf{K},Z_6^{(r)}$. Quatro subchaves de 16 bits são usadas na transformação de saída, e serão denotadas por $Z_1^{(9)},Z_2^{(9)},Z_3^{(9)},Z_4^{(9)}$.

3.5.2 Processo de Decifragem

O processo de decifragem é essencialmente o mesmo que o processo de cifragem, somente uma mudança em relação as subchaves de decifragem $K_i^{(r)}$, pois elas são computadas a partir das subchaves de cifragem $Z_i^{(r)}$ da seguinte maneira:

$$(K_1^{(r)}, K_2^{(r)}, K_3^{(r)}, K_4^{(r)}) = (Z_1^{(10-r)^{-1}}, -Z_3^{(10-r)}, -Z_2^{(10-r)}, Z_4^{(10-r)^{-1}}) \qquad para\ r = 2,3,\mathbf{K},8;$$

$$(K_1^{(r)}, K_2^{(r)}, K_3^{(r)}, K_4^{(r)}) = (Z_1^{(10-r)^{-1}}, -Z_2^{(10-r)}, -Z_3^{(10-r)}, Z_4^{(10-r)^{-1}}) \qquad para\ r = 1\ e\ 9;$$

$$(K_5^{(r)}, K_6^{(r)}) = (Z_5^{(r)}, Z_6^{(r)}) \qquad para\ r = 1,2,3,\mathbf{K},8;$$

onde Z^{-1} denota o inverso multiplicativo (módulo 2^{16} + 1) de Z, isto é, $Z \bullet Z^{-1} = 1$ e –Z denota o inverso aditivo (módulo 2^{16}) de Z, isto é -Z $\boxplus$ Z = 0.

A geração das subchaves de decifragem a partir das subchaves de cifragem é mostrada na tabela 3.21.

Tabela 3.21 – Subchaves de Cifragem e de Decifragem

	Subchaves de Cifragem	**Subchaves de Decifragem**
1º Rodada	$Z_1^{(1)} Z_2^{(1)} Z_3^{(1)} Z_4^{(1)}$ $Z_5^{(1)} Z_6^{(1)}$	$Z_1^{(9)^{-1}} - Z_2^{(9)} - Z_3^{(9)} Z_4^{(9)^{-1}}$ $Z_5^{(8)} Z_6^{(8)}$
2º Rodada	$Z_1^{(2)} Z_2^{(2)} Z_3^{(2)} Z_4^{(2)}$ $Z_5^{(2)} Z_6^{(2)}$	$Z_1^{(8)^{-1}} - Z_2^{(8)} - Z_3^{(8)} Z_4^{(8)^{-1}}$ $Z_5^{(7)} Z_6^{(7)}$
3º Rodada	$Z_1^{(3)} Z_2^{(3)} Z_3^{(3)} Z_4^{(3)}$ $Z_5^{(3)} Z_6^{(3)}$	$Z_1^{(7)^{-1}} - Z_2^{(7)} - Z_3^{(7)} Z_4^{(7)^{-1}}$ $Z_5^{(6)} Z_6^{(6)}$
4º Rodada	$Z_1^{(4)} Z_2^{(4)} Z_3^{(4)} Z_4^{(4)}$ $Z_5^{(4)} Z_6^{(4)}$	$Z_1^{(6)^{-1}} - Z_2^{(6)} - Z_3^{(6)} Z_4^{(6)^{-1}}$ $Z_5^{(5)} Z_6^{(5)}$
5º Rodada	$Z_1^{(5)} Z_2^{(5)} Z_3^{(5)} Z_4^{(5)}$ $Z_5^{(5)} Z_6^{(5)}$	$Z_1^{(5)^{-1}} - Z_2^{(5)} - Z_3^{(5)} Z_4^{(5)^{-1}}$ $Z_5^{(4)} Z_6^{(4)}$
6º Rodada	$Z_1^{(6)} Z_2^{(6)} Z_3^{(6)} Z_4^{(6)}$ $Z_5^{(6)} Z_6^{(6)}$	$Z_1^{(4)^{-1}} - Z_2^{(4)} - Z_3^{(4)} Z_4^{(4)^{-1}}$ $Z_5^{(3)} Z_6^{(3)}$
7º Rodada	$Z_1^{(7)} Z_2^{(7)} Z_3^{(7)} Z_4^{(7)}$ $Z_5^{(7)} Z_6^{(7)}$	$Z_1^{(3)^{-1}} - Z_2^{(3)} - Z_3^{(3)} Z_4^{(3)^{-1}}$ $Z_5^{(2)} Z_6^{(2)}$
8º Rodada	$Z_1^{(8)} Z_2^{(8)} Z_3^{(8)} Z_4^{(8)}$ $Z_5^{(8)} Z_6^{(8)}$	$Z_1^{(2)^{-1}} - Z_2^{(2)} - Z_3^{(2)} Z_4^{(2)^{-1}}$ $Z_5^{(1)} Z_6^{(1)}$
Transformação de Saída	$Z_1^{(9)} Z_2^{(9)} Z_3^{(9)} Z_4^{(9)}$	$Z_1^{(1)^{-1}} - Z_2^{(1)} - Z_3^{(1)} Z_4^{(1)^{-1}}$

3.5.3 Geração das Subchaves

As 52 subchaves de 16 bits usadas no processo de cifragem são geradas da chave secreta do usuário de 128 bits da seguinte maneira: A chave secreta do usuário de 128 bits é dividida

em 8 sub-blocos que são diretamente usados como as primeiras 8 subchaves. A ordenação das 52 subchaves é definida da seguinte maneira:

$$Z_1^{(1)}, Z_2^{(1)}, \mathbf{K}, Z_6^{(1)}, Z_1^{(2)}, \mathbf{K}, Z_6^{(2)}, \mathbf{K}, Z_1^{(8)}, \mathbf{K}, Z_6^{(8)}, Z_1^{(9)}, Z_2^{(9)}, Z_3^{(9)}, Z_4^{(9)}$$

A chave do usuário é rotacionada a esquerda de 25 posições, em seguida o bloco de 128 bits resultante é dividido novamente em 8 sub-blocos que serão as próximas 8 subchaves. O bloco resultante de 128 bits é, novamente, rotacionada a esquerda de 25 posições para produzir as próximas 8 subchaves e este processo é repetido até que se obtenha as 52 subchaves.

3.5.4 Característica de Segurança do IDEA

3.5.4.1 Confusão

A confusão requerida para uma cifra segura é alcançada no IDEA pela "mistura" de 3 operações de grupo incompatíveis. No processo de cifragem do IDEA, mostrada na figura 3.14, as 3 diferentes operações de grupo são arranjadas de tal forma que a saída de uma operação de um tipo nunca é usada como a entrada a uma operação de mesmo tipo.
As 3 operações são incompatíveis nos seguintes sentidos:

1. Nenhum par das 3 operações satisfaz uma lei de distribuitividade. Por exemplo, para as operações $\odot$ e $\boxplus$, existe *a, b* e *c* em F_2^{16}, tal que,

   $$a \boxplus (b \odot c) \neq (a \boxplus b) \odot (a \boxplus c).$$

Por exemplo, quando $a = b = c = 1 = (0,0,\mathbf{K},0,1)$, o lado esquerdo da desigualdade acima é $2 = (0,0,\mathbf{K},0,1,0)$, enquanto que o lado direito é igual a $4 = (0,0,\mathbf{K},0,1,0,0)$.

2. Nenhum par das 3 operações satifaz a lei de associatividade. Por exemplo, para as operações $\boxplus$ e $\oplus$, existe *a, b* e *c* em F_2^{16}, tal que,

$$a \boxplus (b \oplus c) \neq (a \boxplus b) \oplus c.$$

Por exemplo, para $a = b = c = 1 = (0,0,\mathbf{K},0,1)$ em F_2^{16}, o lado esquerdo da inequação acima é $1 = (0,0,\mathbf{K},0,1)$, enquanto o lado direito é igual a $3 = (0,0,\mathbf{K},0,1,1)$. Então, não pode ser mudada arbitrariamente a ordem das operações para simplificar a análise.

3. Sob o mapeamento direto *d* e seu inverso *d-1*, é possível considerar as operações $\odot$ e $\boxplus$ como ações no mesmo conjunto (ou no anel Z_{2^n} ou no corpo $Z_{2^{16}+1}$). Além disso, devemos analisar algumas funções altamente não-lineares no sentido da multiplicação módulo 2^{16} + 1, que é uma função bi-linear sobre $Z_{2^{16}+1}$, que corresponde a uma função não-polinomial sobe $Z_{2^{16}}$. Similarmente, adição módulo 2^{16}, que é uma função afim em cada argumento sobre $Z_{2^{16}}$, corresponde a um polinômio de duas variáveis de grau $2^{16} - 1$ em cada variável sobre $Z_{2^{16}+1}$.

3.5.4.2 Difusão

Uma verificação direta da computação mostra que a função de rodada é "completa", isto é, que cada bit de saída da primeira rodada depende de todo bit do texto claro e de todo bit da subchave usado para cada rodada.

Esta difusão é fornecida na cifra IDEA pela transformação chamada Multiplicação-Adição (MA); a estrutura é mostrada na figura 3.15. A estrutura MA transforma 2 sub-blocos de 16 bits em 2 sub-blocos de 16 bits controlado por duas subchaves de 16 bits. Esta estrutura tem as seguintes propriedades:

- Para escolha das subchaves Z_5 e Z_6, $MA(\cdot,\cdot,Z_5,Z_6)$ é uma transformação inversível; para qualquer escolha de U_1 e U_2, $MA(U_1,U_2,\cdot,\cdot)$ é também uma transformação inversível.
- Esta estrutura tem um efeito de "difusão completa" no sentido que cada sub-bloco de saída depende em todo sub-bloco de entrada, e

- Esta estrutura usa um número mínimo de operações (quatro) requirido para alcançar difusão completa.

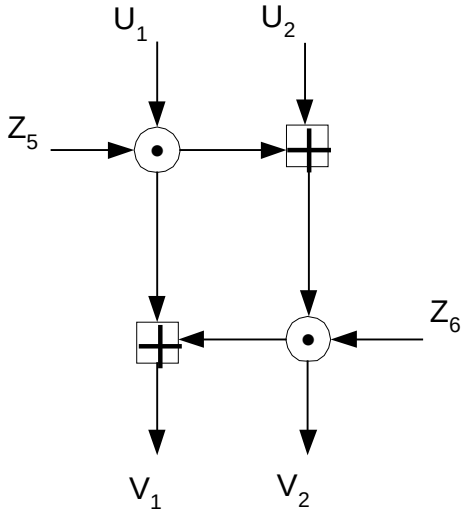

Figura 3.15 – Estrutura MA.

3.5.4.3 Similaridades da cifragem e decifragem

As similaridades da cifragem e decifragem significam que a decifragem é essencialmente o mesmo processo que a cifragem, a diferença está nas subchaves usadas. Então, o mesmo dispositivo pode ser usado tanto para cifragem como para decifragem, sendo feita a pré-computação das subchaves a partir da chave secreta. A função da rodada do IDEA consiste de uma cifra de grupo seguida por uma cifra de involução mais uma permutação involutiva que é um automorfismo do grupo $(F_2^{64}, \otimes)$. Então o IDEA tem similaridades de cifragem e decifragem.

Para o processo de cifragem do IDEA mostrado na figura 3.14, define-se

$$X \otimes Z_A = (X_1 \bullet Z_1, X_2 \boxplus Z_2, X_3 \boxplus Z_3, X_4 \bullet Z_4).$$

Seja $P_I(x)$ a permutação em X que permuta os sub-blocos X_2 e X_3 de $X = (X_1, X_2, X_3, X_4)$ no final de cada rodada. É óbvio que P_I é uma permutação e que $P_I(x \otimes Z_A) = P_I(x) \otimes P_I(Z_A)$, de modo que P_I é um automorfismo do grupo $(F_2^{64}, \otimes)$.

Resta mostrar que a função $In(\cdot, Z_B)$, mostrada na figura 3.16, com a entrada de 64 bits (S_1, S_2, S_3, S_4) e a saída de 64 bits (T_1, T_2, T_3, T_4) controlada pela chave de 32 bits $Z_B = (Z_5, Z_6)$ é uma involução. Isto é, para qualquer Z_B fixo, o inverso da função $In(\cdot, Z_B)$ é ele mesmo. Esta propriedade de auto-inversão é uma conseqüência do fato que o OU-Exclusivo de (S_1, S_2) e (S_3, S_4) é igual ao OU-Exclusivo de (T_1, T_2) e (T_3, T_4); então, a entrada da estrutura MA na figura 3.15 é imutável quando S_1, S_2, S_3 e S_4 são trocados por T_1, T_2, T_3 e T_4.

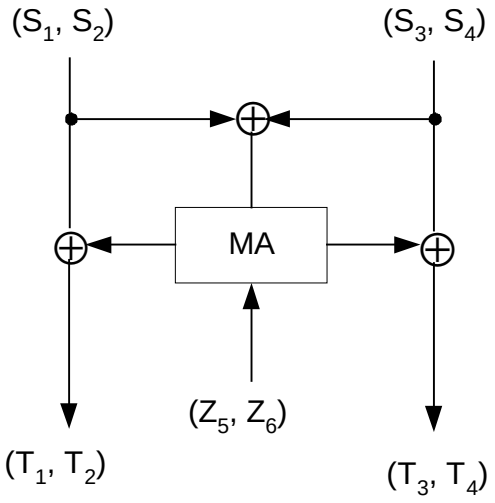

Figura 3.16 – A involução $In(\cdot, Z_B)$

3.6 Criptoanálise

A criptoanálise estuda métodos que possam quebrar a cifra que está sendo atacada. Segundo Denning [DDE82] existem os seguintes tipos de ataques:

- **Ataque por somente texto cifrado**: neste método o criptoanalista deve determinar a chave somente a partir do texto cifrado interceptado, no entanto o método de cifragem, a linguagem do texto claro, o conteúdo do texto cifrado, e certas possíveis palavras podem ser conhecidos. Por exemplo, uma mensagem descrevendo a localização de um tesouro perdido poderia provavelmente conter palavras tais como: Perdido, Tesouro, Norte, Volta, Direita, Kilometros.

- **Ataque por texto claro conhecido**: o criptoanalista conhece alguns pares de textos claro - texto cifrado. Como um exemplo, suponha que uma mensagem cifrada; transmitida de um terminal de usuário, seja interceptada por um cirptoanalista que conhece que a mensagem inicia com um cabeçalho padrão, tal como: “LOGIN”. Em alguns casos, o conhecimento de possíveis palavras é parecido com o ataque por “texto claro conhecido”. Programas cifrados são particularmente vulneráveis por causa do aparecimento regular de palavras chaves – tais como, begin, end, var, procedure, if, then, entre outras.

- **Ataque por texto claro escolhido**: o criptoanalista é hábil para obter o texto cifrado correspondente ao texto claro selecionado. Um sistema de banco de dados pode ser particularmente vulnerável para este tipo de ataque se os usuários podem inserir elementos no banco de dados, e então observar as mudanças no texto cifrado armazenado. Bayer e Metzger [Baye76] chamam este de “problema de registro plantado”.

- **Ataque por texto cifrado escolhido**: apareceu junto com o sistema de chave pública. Embora, o texto claro não seja inteligível, o criptoanalista pode usá-lo para deduzir a chave.

Em 1990, foi proposto por E. Biham e A. Shamir [BS90] um tipo de ataque de texto claro escolhido, onde era atacado o DES, uma cifra de bloco de chave secreta. A descrição deste método foi apresentada no artigo *Cryptoanalysis Diferential of the DES-like cyptosistems* [BS90]. A partir de então a criptoanálise diferencial tornou-se uma base para avaliação da segurança de cifras de chave secreta que eram do tipo Feistel.

4

CRIPTOANÁLISE DIFERENCIAL

4.1 Introdução

O objetivo deste capítulo é fornecer uma visão geral da técnica de criptoanálise diferencial proposta por Biham e Shamir em 1990 [BS90].

Criptoanálise diferencial é um método que analisa os efeitos de diferenças particulares em pares de texto claro, nas diferenças dos pares de texto cifrado resultantes. Estas diferenças são usadas para atribuir probabilidades às possíveis chaves e localizar as chaves mais prováveis.

4.2 Notação

Para $X \in \{0,1\}^n$ os bits de X serão numerados a partir de 0 (zero) da direita para a esquerda, sendo o bit 0 o menos significativo (tal numeração de bits facilita a comparação da parte teórica com a implementação deste ataque e também com algumas das referências bibliográficas pertinentes a este capítulo). A notação usada na criptoanálise diferencial é a seguinte:

XOR : Operação OU-Exclusivo de duas palavras de mesmo tamanho.

n_x : um número hexadecimal é denotado por um x subscrito (ex.: $10_x = 16$);

P: O texto claro é denotada por P. P^* é o outro texto claro no par e $\Delta P = P' \oplus P''$ é a diferença de pares dos textos claro.

P(x): a permutação *P* é denotada por *P(x)*. Note que *P* é uma variável que representa o texto claro.

C: Os textos cifrados dos correspondentes textos claros *P'* e *P''* são denotados por C' e C''. $\Delta C = C' \oplus C''$ é a diferença de pares dos textos cifrado.

S_i: Os S-boxes S_1, S_2, S_3, S_4.

ΔX: Representa o XOR de pares palavras na entrada do S-Box. Representa valores intermediários da cifra.

ΔY: Representa o XOR de pares palavras na saída do S-Box. Representa valores intermediários da cifra.

4.3 Diferença de Pares de Textos Claro

Definição 4.1 - Uma chave independente é uma lista de *n* subchaves que não é necessariamente derivada do algoritmo de programação de chaves.

Definição 4.2 - Uma tabela que mostra a distribuição das entradas e saídas XORs de todos os possíveis pares de entrada e saída de um S-box, é chamada de Tabela de Distribuição dos pares XOR do S-box. Nesta tabela, cada linha corresponde a uma particular entrada XOR, cada coluna corresponde a uma particular saída XOR e os valores da tabela contam o número de pares de saída XOR possíveis com tais entradas.

Definição 4.3 - Seja X um valor de n-bits e Y um valor de m-bits. Digamos que X pode causar Y por um S-box se há um par no qual a entrada XOR do S-box é igual a X e a saída XOR do S-box é igual a Y. Se há tal par, escrevemos $X \rightarrow Y$, e senão dizemos que X não pode causar Y pelo S-box e escrevemos $X \not\rightarrow Y$.

Definição 4.4 - Dizemos que X pode causar Y com probabilidade *p* por um S-box se para uma fração *p* dos pares em que a entrada XOR do S-box é igual a X, a saída XOR é igual a Y.

Em uma cifra idealmente aleatória, a probabilidade de que uma determinada diferença de saída ΔY ocorra dado uma determinada diferença de entrada ΔX, é $1/2^n$, onde *n* é o número de bits de entrada do S-Box [HE00].

Definição 4.5 - Dizemos que X pode causar Y com probabilidade *p* pela função F, se para uma fração *p* de todos os possíveis pares de entrada cifrados por todos os valores possíveis da subchave em que a entrada XOR da função F é igual a X, a saída XOR é igual a Y. Se *p>0*, denotamos esta possibilidade como $X \rightarrow Y$.

Definição 4.6 - Associado a qualquer par de cifragens, estão os valores XOR de seus dois textos claro, o XOR de seus textos cifrados, os XORs das entradas de cada rodada nas duas execuções e os XORs das saídas de cada rodada nas duas execuções. Estes valores XOR formam uma característica de *n* rodadas. Uma característica tem uma probabilidade, que é a probabilidade que um par aleatório com o XOR do texto claro escolhido tem os pares XOR da rodada e do texto cifrado especificado na característica. Denotamos os XORs dos textos claro de uma característica por Ω_P e os XOR de textos cifrados por Ω_C.

Definição 4.7 - Uma característica de *n* rodadas é uma tripla $\Omega = (\Omega_P, \Omega_\Delta, \Omega_C)$, onde Ω_P e Ω_C são palavras de *m* bits e Ω_Δ é uma lista de n elementos $\Omega_\Delta = (\Delta_1, \Delta_2, \mathbf{K}, \Delta_n)$, cada um dos quais é um par da forma $\Delta_i = (\Delta X_i, \Delta Y_i)$, onde ΔX_i e ΔY_i são números de *m* bits, onde *m* é o tamanho do bloco do cripto-sistema.

Definição 4.8 – Um par correto com relação a uma característica de *n* rodadas $\Omega = (\Omega_P, \Omega_\Delta, \Omega_C)$ e uma chave independente K é um par para o qual $\Delta P = \Omega_P$ e para as primeiras *n* rodadas da cifragem do par usando a chave independente K, a entrada XOR da i-ésima rodada é igual a ΔX_i e a saída XOR do S-Box é igual a ΔY_i. Todo par, que não é

correto com relação à característica e à chave independente, é chamado um par errado com relação à característica e à chave independente.

Definição 4.9 - A rodada i de uma característica Ω tem probabilidade p_i^{Ω} se $\Delta X_i \rightarrow \Delta Y_i$ com probabilidade p_i^{Ω} pela função *F*.

Definição 4.10 - Uma característica Ω de *n*-rodadas tem probabilidade p^{Ω}, se p^{Ω} é o produto das probabilidades de suas *n*-rodadas:

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

Definição 4.11 - Uma característica $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_C)$ é chamada uma característica iterativa, se os valores trocados dos semi-blocos de Ω_P são iguais a Ω_C.

Definição 4.12 - A razão entre o número de pares corretos e a contagem média em um esquema de contagem é chamada razão sinal-ruído do esquema de contagem e é denotado por *S/N*.

Se procurarmos por k bits da chave então contamos o número de ocorrências de 2^k possíveis valores da chave em 2^k contadores. Os contadores contém uma contagem média de $2^{\frac{m.a.b}{k}}$, 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 (isto é, contados e descartados). A razão Sinal Ruído de um esquema de contagem é:

$$S/N = \frac{m.p}{m.a.b/2^k} = \frac{2^k.p}{a.b}$$

Um corolário desta fórmula é que a razão sinal ruído de um esquema de contagem é independente da quantidade de pares usado no esquema. Outro corolário é que diferentes

esquemas de contagem baseado na mesma característica mas com um diferente número de bits da subchave tem diferente S/N.

Quando o S/N é alto o bastante, apenas poucas ocorrências de pares corretos são necessários para identificar unicamente o valor correto dos bits da subchave.

4.4 A Aplicação da Criptoanálise diferencial

Para mostrar o funcionamento da criptoanálise diferencial vamos usar uma pequena cifra não tendo fins práticos e sim didáticos. Esta cifra é uma estrutura básica de substituição-permutação, como mostrado na figura 4.1. Ela cifra blocos de texto claro de 16 bits e gera blocos de texto cifrado de 16 bits, isto através da repetição de uma rodada durante 4 vezes. Cada rodada consiste de uma substituição, uma permutação e uma operação de XOR com uma subchave. Essas operações são típicas e podem ser encontradas em cifras como o DES e o Rijndael.

4.4.1 Cifragem

Para o processo de cifragem, a primeira operação efetuada em cada rodada é a de XOR entre o bloco de entrada e a subchave da referida rodada. Logo em seguida dividimos o bloco resultante de 16 bits em 4 blocos de 4 bits cada. Cada bloco forma uma entrada para um S-Box 4x4 (ou seja, uma substituição com 4 bits de entrada e 4 bits de saída). A propriedade fundamental de um S-Box é seu mapeamento não-linear, isto é, os bits de saída não representam uma operação linear dos bits de entrada. Para os 4 S-Boxes usaremos a mesma representação, conforme a tabela 4.1.

Tabela 4.1 – Representação do S-Box

Entrada	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
Saída	E_x	4_x	D_x	1_x	2_x	F_x	B_x	8_x	3_x	A_x	6_x	C_x	5_x	9_x	0_x	7_x

À saída dos S-Boxes é aplicada a permutação indicada na figura 4.1 é mostrada na tabela 4.2. Aqui o bit mais a esquerda representa o bit 1 e o mais à direita o bit 16.

Tabela 4.2 – Permutação

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

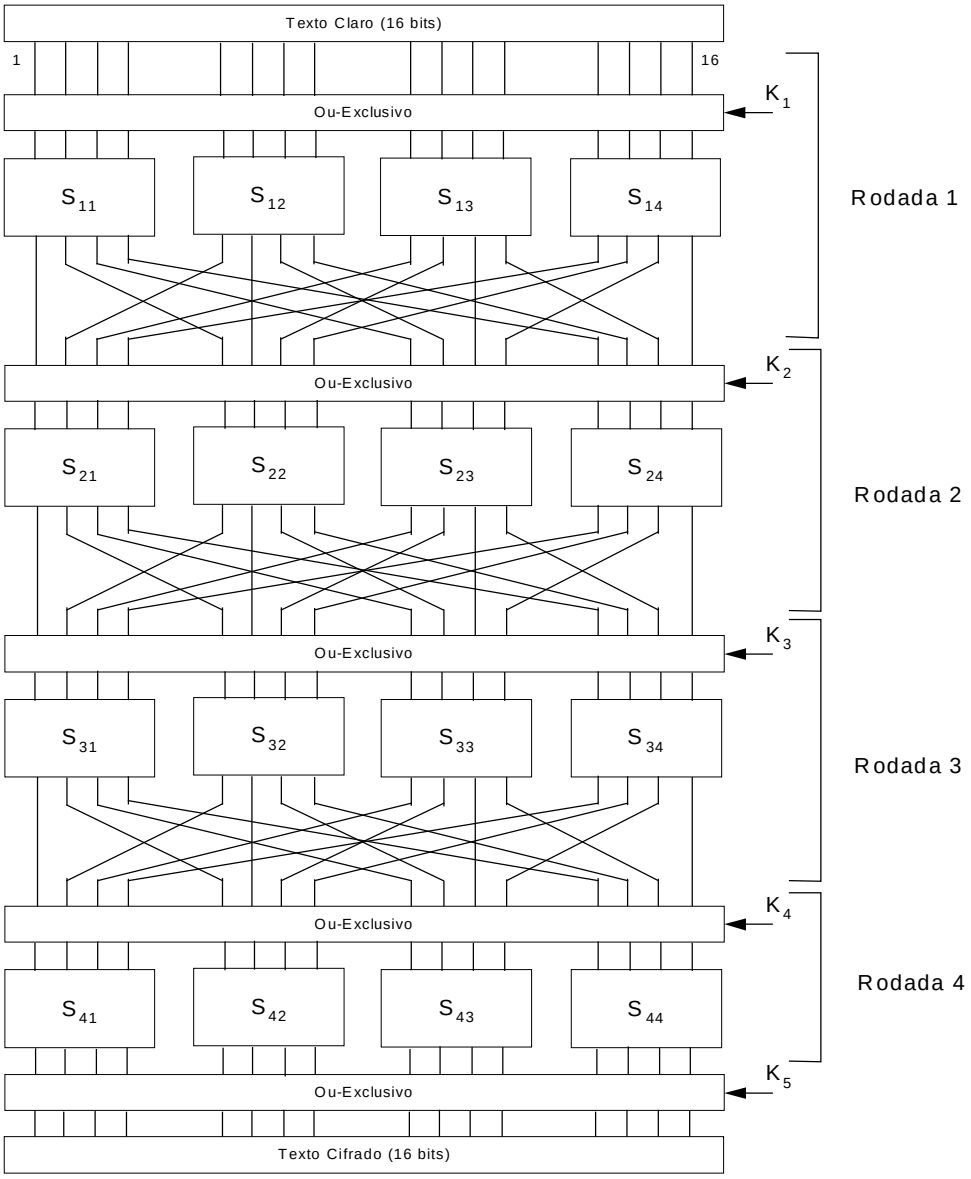

Figura 4.1 – Processo de Cifragem

4.4.2 Decifragem

Para decifrar, os dados são passados de volta através da rede. Com isso, a decifragem é também da forma de uma rede de substituição-permutação como ilustrado na figura 4.2. Observe que as subchaves são aplicadas do modo reverso, ou seja, na primeira rodada aplica-se a K5, na segunda K4, e assim por diante. Devemos salientar que os S-Boxes devem ser bijetivos, isto é, um mapeamento um-para-um com o mesmo número de bits de entrada e saída. Sem essa característica fica difícil obter a decifragem com a mesma estrutura.

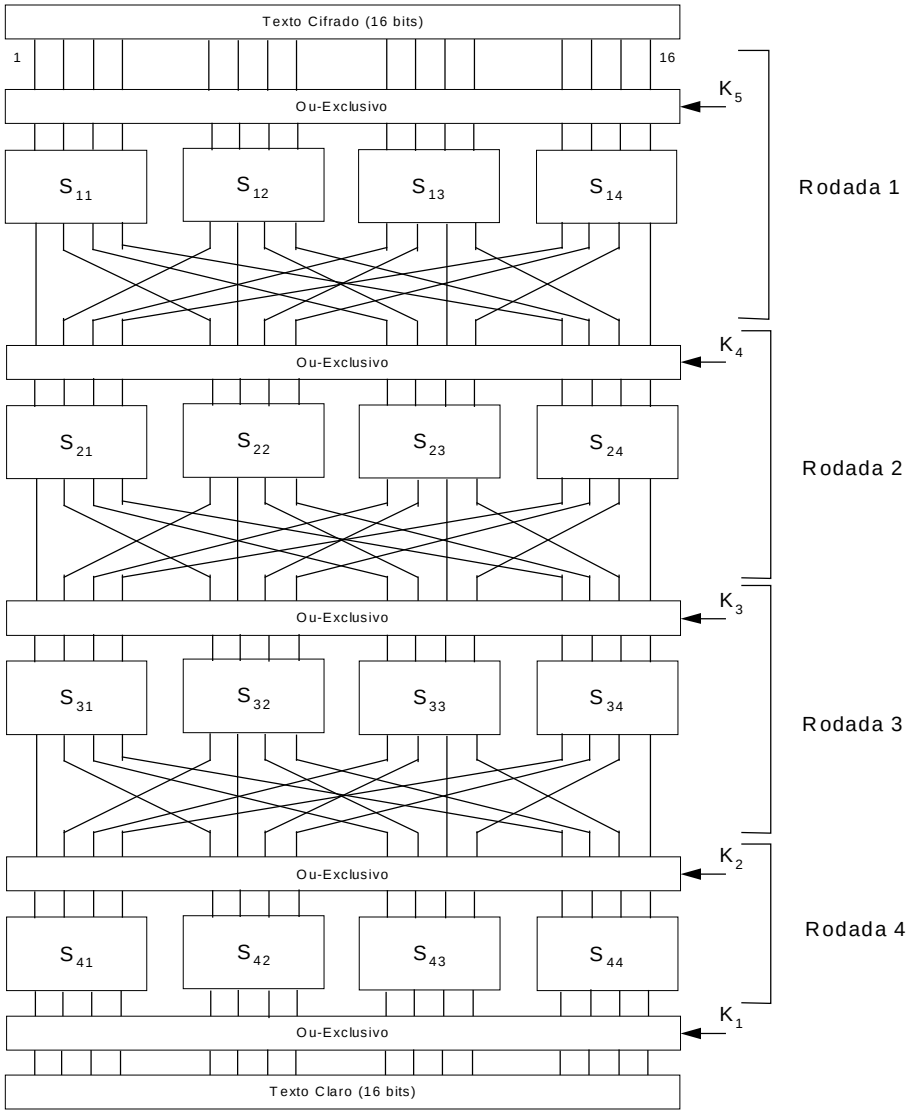

Figura 4.2 – Processo de Decifragem

4.4.3 Tabela de Distribuição de Diferenças

Para a construção da Tabela de Distribuição de Diferenças, vamos examinar os pares de diferenças de um S-Box. Considerando a representação do S-Box 4x4 dada na figura 3 com entrada $X = [X_1 X_2 X_3 X_4]$ e saída $Y = [Y_1 Y_2 Y_3 Y_4]$, todas as diferenças de pares de um S-Box , $(\Delta X, \Delta Y)$, são examinadas e a probabilidade de se ter ΔY dado ΔX é derivada pela consideração dos pares de entrada (X', X''), tal que $X' \oplus X'' = \Delta X$. Para um S-Box 4x4, ΔX terá 16 valores, então para montarmos a tabela vamos considerar os 16 valores para X' e então os valores de ΔX restringe os valores para X'', tal que $X'' = X' \oplus \Delta X$. O algoritmo para a geração da tabela é o seguinte:

Para $\Delta X = 0_x$ até F_x faça

Para $X' = 0_x$ até F_x faça

Para $X'' = 0_x$ até F_x faça

Se $\Delta X = X' \oplus X''$ então

Tabular $\Delta Y = S(X') \oplus S(X'')$

Se o S-Box fosse "ideal", o número de ocorrências de todos os pares de diferença seria 1, para uma probabilidade de 1/16 de um valor ΔY dado ΔX . A tabela 4.3 representa a Tabela de Distribuição de Diferenças para o S-Box da cifra, em que cada linha representa valores de ΔX e as colunas representam os valores de ΔY . Cada elemento da tabela representa o número de ocorrências do correspondente valor de saída ΔY dado a diferença de entrada ΔX . Note que, além do caso especial $(\Delta X = 0_x, \Delta Y = 0_x)$, o maior valor na tabela é 8, correspondendo a $\Delta X = B_x$ e $\Delta Y = 2_x$. Então, a probabilidade que $\Delta Y = 2_x$ dado um arbitrário par de valores de entrada que satisfaça $\Delta X = B_x$ é 8/16. O menor valor na tabela é 0 e ocorre para alguns pares de diferença. Neste caso, a probabilidade de ΔY ocorrer, dado o valor de ΔX , é 0.

Tabela 4.3 – Tabela de Distribuição de Diferenças

		Diferenças de Saída															
Diferença Entrada		**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	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1_x	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
	2_x	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
	3_x	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
	4_x	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
	5_x	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
	6_x	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
	7_x	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
	8_x	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
	9_x	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
	A_x	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
	B_x	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
	C_x	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
	D_x	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
	E_x	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
	F_x	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Analisando a tabela podemos verificar algumas propriedades:

1. A soma de todos os elementos de uma linha é $2^n = 16$, assim como a soma de todos os elementos e uma coluna é $2^n = 16$.
2. Todos os elementos tem valor Par, pois um par de entrada (ou saída) representado como (X', X'') tem o mesmo valor ΔX que o par (X'', X') pois $\Delta X = X' \oplus X'' = X'' \oplus X'$.
3. A diferença de entrada $\Delta X = 0_x$ resulta em uma diferença de saída $\Delta Y = 0_x$.

Antes de derivarmos a característica diferencial para a cifra, vamos ver como o par diferencial retira a influência da chave sobre os dados. Considere a figura 4.3. A entrada ao S-Box é X e a saída é Y. Entretanto, na estrutura da cifra devemos considerar a chave aplicada na entrada de cada S-Box. Neste caso, se a entrada ao S-Box é $W = [W_1 W_2 W_3 W_4]$, considerando as diferenças para esta entrada como

$$\Delta W = [W_1^{'} \oplus W_1^{''} W_2^{'} \oplus W_2^{''} W_3^{'} \oplus W_3^{''} W_4^{'} \oplus W_4^{''}]$$

onde $W' = [W_1^{'} W_2^{'} W_3^{'} W_4^{'}]$ e $W'' = [W_1^{''} W_2^{''} W_3^{''} W_4^{''}]$ representam os dois valores de entrada que formam o par para referida diferença.

Desde que os bits da chave mantenham o mesmo para ambos W' e W'',

$$\Delta W_i = W_i^{'} \oplus W_i^{''} = (X_i^{'} \oplus K_i) \oplus (X_i^{''} \oplus K_i) = X_i^{'} \oplus X_i^{''} = \Delta X_i$$

sendo $K_i \oplus K_i = 0$. Então os bits da chave não têm influência sobre os valores de diferença de entrada e, portanto, são ignorados,

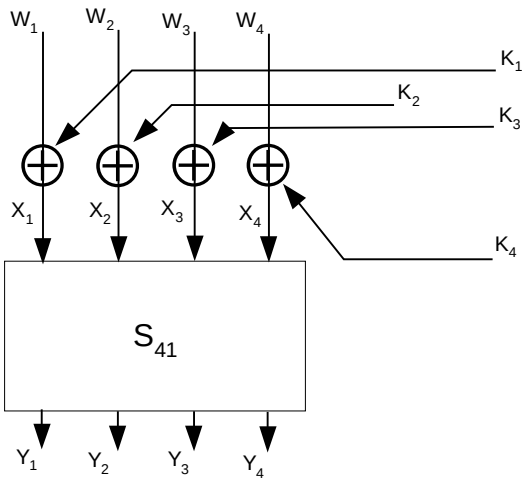

Figura 4.3 – S-Box 4x4

4.4.4 Construindo uma Característica Diferencial

Conforme a definição 4.6, uma característica diferencial envolve textos claro e as entradas dos S-boxes tendo uma probabilidade relevante. A partir do S-Box da última rodada podemos atacar a cifra pela recuperação de um determinado número de bits da subchave da última rodada. A característica diferencial construída é ilustrada na figura 4.4.

Para esta característica envolvemos os S-Boxes S_{12}, S_{23}, S_{32} e S_{33}. O diagrama da figura 4.4 ilustra a influência de diferenças em bits diferentes de 0, indicando como eles percorrem a rede e salientando os S-Boxes que podem ser considerados como ativos (isto é, para qual há diferencial diferente de 0). Note que isto desenvolve uma característica diferencial para as 3 primeiras rodada da cifra e não para as 4 rodadas.

Usamos os seguintes pares diferencial do S-Box:

$$S_{12} : \Delta X = B_x \rightarrow \Delta Y = 2_x \quad - \quad p = 8/16$$
$$S_{23} : \Delta X = 4_x \rightarrow \Delta Y = 6_x \quad - \quad p = 6/16$$
$$S_{32} : \Delta X = 2_x \rightarrow \Delta Y = 5_x \quad - \quad p = 6/16$$
$$S_{33} : \Delta X = 2_x \rightarrow \Delta Y = 5_x \quad - \quad p = 6/16$$

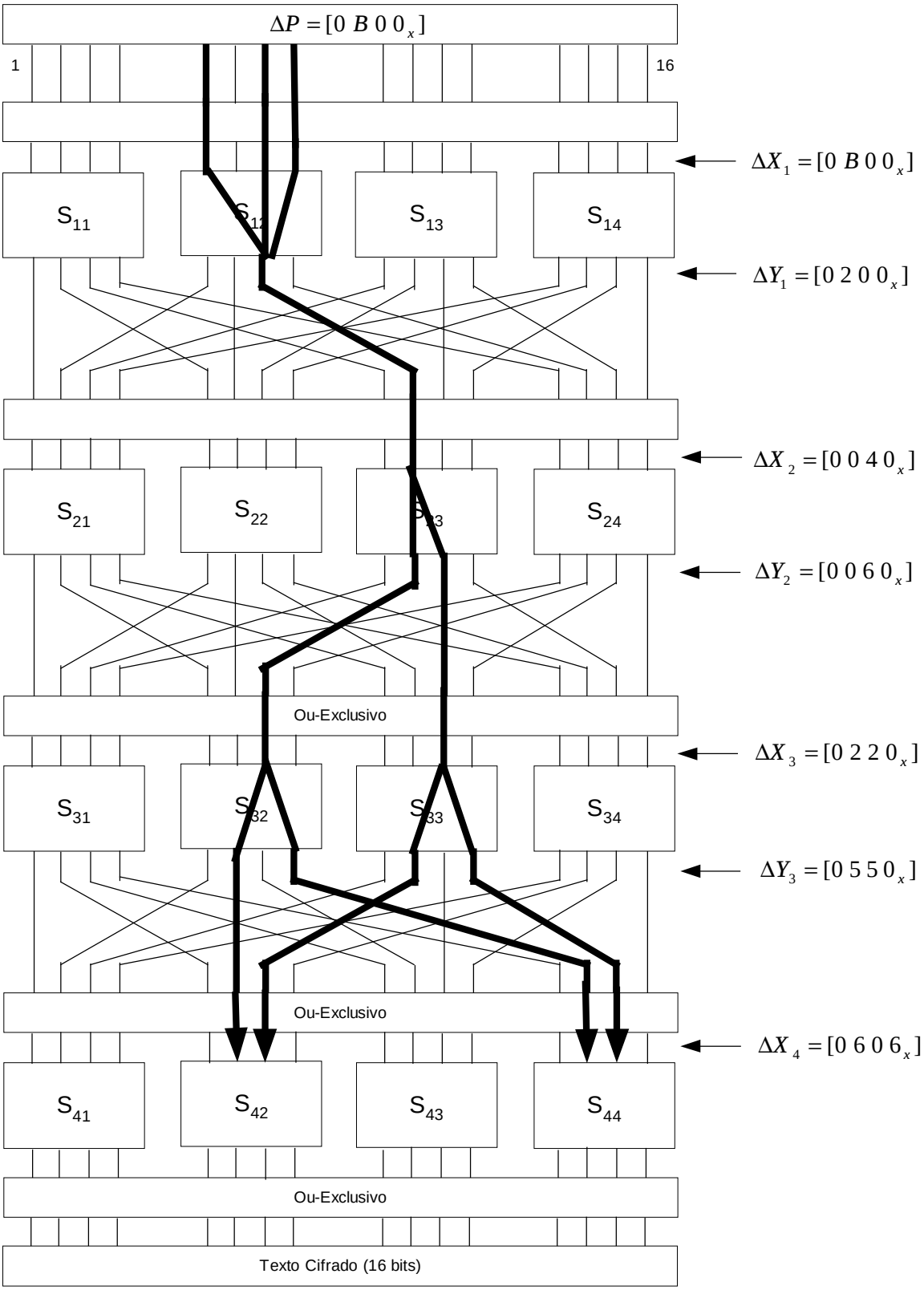

Figura 4.4 – Característica diferencial.

Todos os outros S-Boxes terão 0 na entrada e conseqüentemente 0 na saída.

A diferença de entrada à cifra é a diferença de entrada a primeira rodada, e é dada por

$$\Delta P = \Delta X_1 = [0000\,1011\,0000\,0000]$$

Como resultado

$$\Delta Y_1 = [0000\,0010\,0000\,0000]$$

Considerando o par diferença para S_{12} listado acima e aplicando a permutação da primeira rodada

$$\Delta X_2 = [0000\,0000\,0100\,0000]$$

com probabilidade de 8/16 = 1/2, dada a diferença de texto claro ΔP.

Agora o diferencial na segunda rodada, usando o par diferencial para S_{23}, resulta em

$$\Delta Y_2 = [0000\,0000\,0110\,0000]$$

e permutação na rodada 2 gera

$$\Delta X_3 = [0000\,0010\,0010\,0000]$$

com probabilidade 6/16 dado ΔY_2 e pela definição 4.10 a probabilidade da característica será 8/16 x 6/16 = 3/16 em relação a ΔP.

Em seguida, usamos as diferenças para os S-Boxes da terceira rodada, S_{32} e S_{33}, e a permutação da terceira rodada, gerando

$$\Delta Y_3 = [0000\ 0101\ 0101\ 0000]$$

e

$$\Delta X_4 = [0000\ 0110\ 0000\ 0110] \qquad (4.1)$$

com uma probabilidade de $(6/16)^2$ dado ΔX_3 e, então, tem-se uma probabilidade de 8/16 x 6/16 x $(6/16)^2$ = 27/1024 em relação a diferença ΔP, onde recorremos a definição 4.10.

Durante o processo de criptoanálise, alguns pares de textos claro cujo $\Delta P = [0000\ 1011\ 0000\ 0000]$ serão cifrados. Com alta probabilidade, 27/1024, a característica diferencial ocorrerá. Denominamos tais pares como pares corretos. Pares diferencial de textos claro cuja característica não ocorre são chamados pares errados, conforme definição 4.8.

4.4.5 Extraindo os Bits da subchave

Uma vez que uma característica diferencial de $(R - 1)$ rodadas é descoberta para uma cifra de R rodadas com uma probabilidade bastante aceitável, é concebível atacar a cifra pela recuperação de bits da última subchave. Para nossa cifra, é possível extrair bits da subchave K_5.

Uma decifragem parcial é executada para cada par de textos cifrado correspondendo aos pares de textos claro usados para gerar a diferença de entrada ΔP, para todos os possíveis valores parciais da subchave. Uma contagem é mantida para cada valor parcial da subchave. A contagem é incrementada quando a diferença para a entrada da última rodada corresponde ao valor esperado da característica diferencial. O valor parcial da subchave que tem a maior contagem é assumido indicar os valores corretos dos bits da subchave. Isto funciona porque é assumido que o valor parcial correto da subchave resultará da ocorrência de pares corretos, conforme definição 4.8, desde que a característica tenha uma alta probabilidade de ocorrer. Quando um par errado ocorrer, mesmo com a decifragem parcial com a subchave correta, a contagem para a subchave correta não será incrementada.

Considerando o ataque em nossa cifra, a característica diferencial afeta as entradas dos S-Boxes S_{42} e S_{44} na última rodada. Para cada par de texto cifrado, tentaremos todos os 256 valores para $[K_{5,5}\,\mathbf{K}\,K_{5,8}, K_{5,13}\,\mathbf{K}\,K_{5,16}]$. Para cada valor parcial da subchave, incrementaremos a contagem quando a diferença da entrada na rodada final determinada pela decifragem parcial for a mesma de (4.1), onde determinamos os valores de $[\Delta U_{4,5}\,\mathbf{K}\,\Delta U_{4,8}, \Delta U_{4,13}\,\mathbf{K}\,\Delta U_{4,16}]$ pela execução "de volta" dos dados através da subchave parcial e através dos S-Boxes S_{42} e S_{44}. Para cada valor parcial da subchave, a contagem representa o número de ocorrências de diferenças que são consistentes com os pares corretos (assumindo que a subchave parcial é o valor correto). A maior contagem é assumida como o valor correto, desde que assumimos que estamos observando a ocorrência com alta probabilidade do par correto.

Note que não é necessário executar a decifragem parcial para todo par de texto cifrado. Desde que a diferença de entrada na última rodada influência apenas 2 S-Boxes quando a característica ocorre (isto é, para pares corretos), as diferenças de bits do texto cifrado correspondendo aos S-Boxes S_{41} e S_{43} devem ser 0. Então filtramos alguns pares errados ao descartar pares de textos cifrado para os quais não aparecem zeros nos sub-blocos apropriado da diferença de texto cifrado. Nestes casos, desde que pares de textos cifrado não correspondam a um par correto, não é necessário examinar $[\Delta U_{4,5}\,\mathbf{K}\,\Delta U_{4,8}, \Delta U_{4,13}\,\mathbf{K}\,\Delta U_{4,16}]$.

Simulamos o ataque a nossa cifra usando subchaves geradas aleatoriamente ao gerar 5000 pares escolhidos de texto claro / texto cifrado (isto é, 10000 cifragens com pares de texto claro satisfazendo $\Delta P = [0000\,1011\,0000\,0000]$) e seguindo o processo descrito acima. O valor parcial da subchave foi $[K_{5,5}\,\mathbf{K}\,K_{5,8}, K_{5,13}\,\mathbf{K}\,K_{5,16}] = [0010,0100] = [2,4]_x$. Como esperado, a maior contagem foi observada para o valor parcial da subchave $[2,4]_x$, confirmando que o ataque bem sucedido derivou os bits da subchave. A tabela 4.4 mostra um resumo parcial dos dados derivados das subchaves contadas (Os dados completos envolvem 256 dados de entrada, uma para cada valor parcial da suchave). Os valores da tabela indicam que a probabilidade estimada da ocorrência de pares corretos para a subchave candidata derivou de

$$prob = cont / 5000$$

onde cont é a contagem de um valor de subchave.

Como pode ser visto na tabela, a maior probabilidade ocorre para o valor parcial da subchave $[K_{5,5}\,\mathbf{K}\,K_{5,8}, K_{5,13}\,\mathbf{K}\,K_{5,16}] = [2,4]_x$ e a observação foi, na realidade, verdadeira para o conjunto completo de valores da subchave parcial.

Em nosso exemplo, esperaríamos a probabilidade da ocorrência dos pares corretos ser $p_D = 27/1024 = 0{,}0264$ e encontramos experimentalmente a probabilidade para o valor correto da subchave $[2,4]_x$ de $p_D = 0{,}0244$. Note que as vezes os valores incorretos da subchave tem grande contagem. Isto indica que o exame de subchaves incorretos não é precisamente equivalente à comparação de diferenças aleatórias ao valor diferencial esperado. Há diversos fatores que fazem com que a contagem seja diferente da nossa expectativa teórica, isto inclui as propriedades dos S-Boxes influenciando a decifragem parcial para diferentes subchaves, a imprecisão da suposição de independência requerida para determinação da probabilidade da característica e o fato de que diferenciais podem ser compostas de múltiplas características diferenciais [XLS91].

Tabela 4.4 – Resultados para o Ataque Diferencial

Subchave Parcial $[K_{5,5}\,\mathbf{K}\,K_{5,8}, K_{5,13}\,\mathbf{K}\,K_{1,16}]$	**Prob**	**Subchave Parcial** $[K_{5,5}\,\mathbf{K}\,K_{5,8}, K_{5,13}\,\mathbf{K}\,K_{1,16}]$	**Prob**
1C	0,0000	2A	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

5

CRIPTOANÁLISE DIFERENCIAL DO CRIPTO-SISTEMA BLOWFISH

O objetivo deste capítulo é apresentar uma aplicação da criptoanálise diferencial ao cripto-sistema de chave secreta Blowfish.

5.1 Blowfish Reduzido

Antes de procedermos à aplicação da técnica de criptoanálise diferencial, vamos analisar as características do Blowfish. Para isso, devido às dificuldades de tratar diretamente com a cifra originalmente descrita no capítulo 3, faremos uma versão reduzida da mesma, que chamaremos Blowfish Reduzido.

O Blowfish reduzido cifra blocos de texto claro de 16 bits, gerando blocos de texto cifrado de 16 bits. As subchaves são de 8 bits cada, denotadas por K1, K2, ..., K17, K18. Os S-Boxes são de 2x8, ou seja, tem como entrada palavras de 2 bits e como saída palavras de 8 bits. O número de rodadas é o mesmo da cifra original, ou seja, 16 rodadas. O algoritmo de cifragem é conforme apresentado na figura 5.1. Os S-Boxes são dependentes da chave e servem para compor a função F, conforme a figura 5.2.

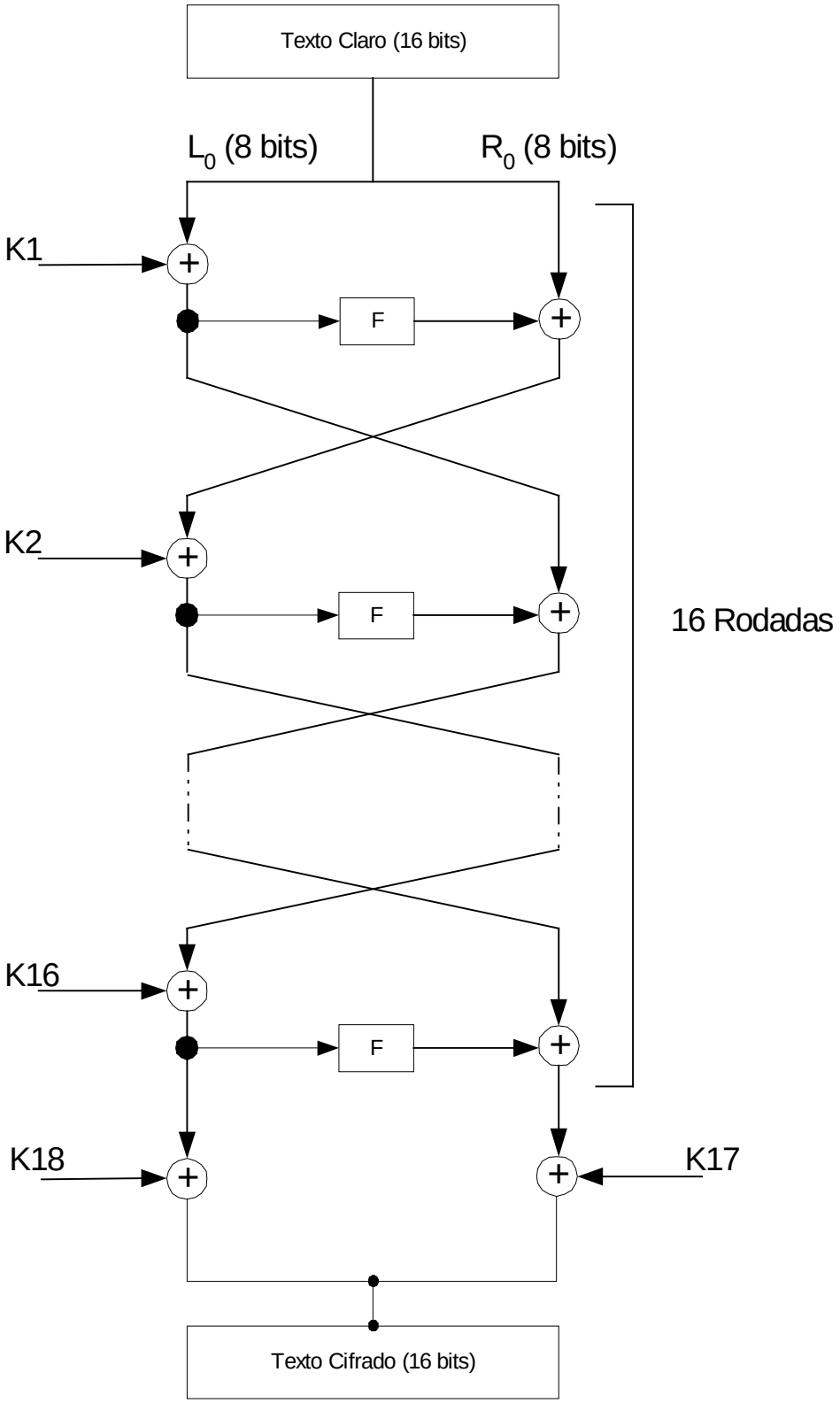

Figura 5.1 – Algoritmo de Cifragem do Blowfish reduzido.

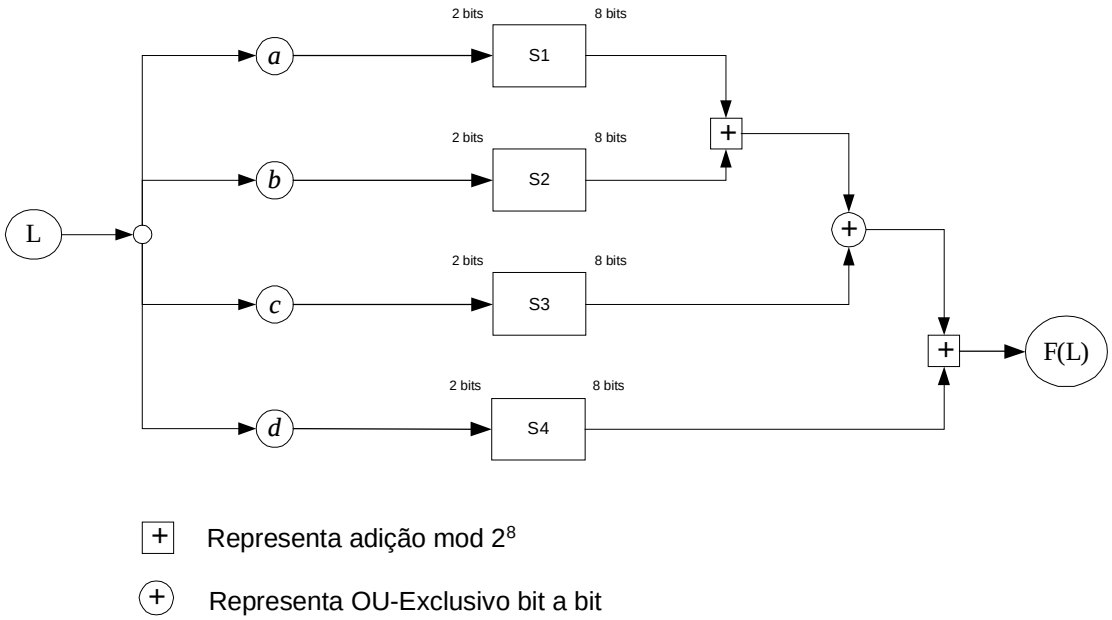

Figura 5.2 – Função **F** do Blowfish reduzido.

É importante mencionar que essa redução não tira as características da cifra e nem muda, essencialmente, o seu funcionamento. Ela serve para nós estudarmos o comportamento dos S-Boxes dependentes da chave e podermos avaliar se a cifra gera S-Boxes que oferecem pouca resistência à criptoanálise diferencial.

5.1.1 Análise dos S-Boxes

Com o Blowfish reduzido podemos fazer a avaliação dos valores que são gerados nos S-Boxes, pois estes são valores gerados através do próprio algoritmo de cifragem, não indicando que temos alguma informação sobre a chave. Outra vantagem do Blowfish reduzido é que podemos verificar a tabela de distribuição para todos os valores de chave possível e com isso podemos observar esses valores e verificar uma possível fraqueza em relação a criptoanálise diferencial.

Um fato que pudemos verificar ao analisar todos os valores possíveis dos S-Boxes e que comprova, para o modelo reduzido, a suposição feita por Vaudenay [SV95], é que existem chaves que geram S-Box, tais que suas saídas são iguais, isto é, $S(a) = S(a')$, com $a \neq a'$.

Outro fato importante observado foi a repetição dos valores das diferenças de saída dos S-Boxes. A tabela 5.1 mostra a quantidade dessas repetições para o S-Box 1. A tabela completa para todos os S-Boxes é mostrada no apêndice C.

Tabela 5.1 – Repetições de Valores de Saída dos S-Box 1

Valores de Saída para o S-Box 1	Qde de Repetições
1_x	12
10_x	16
11_x	16
12_x	32
13_x	4
14_x	12
15_x	6
16_x	12
17_x	12
18_x	10
19_x	6
$1a_x$	16
$1b_x$	8
$1c_x$	12
$1d_x$	12
$1e_x$	16
$1f_x$	6
2_x	10
20_x	12
21_x	6
22_x	16
23_x	12
24_x	16
25_x	16
26_x	10
27_x	10
28_x	14
29_x	14
$2a_x$	14
$2b_x$	18
$2c_x$	6
$2d_x$	14
$2e_x$	12
$2f_x$	12
3_x	16
30_x	16

Valores de Saída para o S-Box 1	Qde de Repetições
31_x	28
32_x	12

Com isso podemos conjecturar que o fenômeno irá ocorrer na cifra Blowfish com uma proporção semelhante ao da cifra reduzida.

5.1.2 Característica Diferencial para o Blowfish Reduzido

Para montar o ataque de criptoanálise diferencial na cifra Blowfish reduzida, primeiramente iremos obter todos os valores dos S-Boxes, pois, como a chave será constante, os valores dos S-Boxes não mudarão até que se mude a chave. Salientamos que, com a obtenção desses valores, não teremos informações da chave secreta, pois os valores são gerados através do próprio algoritmo de cifragem. Analisamos cada S-Box variando de 0_x a 3_x o valor de entrada e observando os valores de saída. Estas informações estão nas tabelas 5.2 a 5.5.

Tabela 5.2 – Valores para o S-Box 1

Valores de Entrada	**Valores de Saída**
0_x	23_x
1_x	9_x
2_x	$2d_x$
3_x	$4f_x$

Tabela 5.3 – Valores para o S-Box 2

Valores de Entrada	**Valores de Saída**
0_x	36_x
1_x	80_x
2_x	75_x
3_x	$8b_x$

Tabela 5.4 – Valores para o S-Box 3

Valores de Entrada	**Valores de Saída**
0_x	99_x
1_x	44_x
2_x	$1f_x$
3_x	55_x

Tabela 5.5 – Valores para o S-Box 4

Valores de Entrada	**Valores de Saída**
0_x	$6e_x$
1_x	92_x
2_x	$f8_x$
3_x	df_x

Agora, com os valores dos S-Boxes já obtidos, poderemos obter a tabela de distribuição de diferenças para cada S-Box. Esses valores são tabulados nas tabelas 5.6 a 5.9.

Tabela 5.6 – Tabela de distribuição de diferenças para o S-Box 1

	Diferenças de Saída						
		0_x	**14_x**	**87_x**	**95_x**	**6_x**	**12_x**	**81_x**
Diferenças de Entrada	**0_x**	4	0	0	0	0	0	0
	1_x	0	2	2	0	0	0	0
	2_x	0	0	0	2	2	0	0
	3_x	0	0	0	0	0	2	2

Tabela 5.7 – Tabela de distribuição de diferenças para o S-Box 2.

	Diferenças de Saída						
		0_x	**87_x**	**$1d_x$**	**e_x**	**94_x**	**13_x**	**89_x**
Diferenças de Entrada	**0_x**	4	0	0	0	0	0	0
	1_x	0	2	2	0	0	0	0
	2_x	0	0	0	2	2	0	0
	3_x	0	0	0	0	0	2	2

Tabela 5.8 – Tabela de distribuição de diferenças para o S-Box 3

		Diferenças de Saída						
		0_x	**$5e_x$**	**$f2_x$**	**75_x**	**$d9_x$**	**87_x**	**$2b_x$**
Diferenças de Entrada	**0_x**	4	0	0	0	0	0	0
	1_x	0	2	2	0	0	0	0
	2_x	0	0	0	2	2	0	0
	3_x	0	0	0	0	0	2	2

Tabela 5.9 – Tabela de distribuição de diferenças para o S-Box 4

		Diferenças de Saída						
		0_x	**ca_x**	**$7a_x$**	**$a0_x$**	**10_x**	**da_x**	**$6a_x$**
Diferenças de Entrada	**0_x**	4	0	0	0	0	0	0
	1_x	0	2	2	0	0	0	0
	2_x	0	0	0	2	2	0	0
	3_x	0	0	0	0	0	2	2

Depois de ter levantado os valores dos S-Boxes e a tabela de distribuição de diferenças, podemos montar uma característica diferencial para o blowfish reduzido com 4 rodadas. Com a diferença de texto claro $\Delta P = 8006_x$, montamos a seguinte característica:

$$\Omega = (\Omega_P, \Delta X_1, \Delta Y_1, \Delta X_2, \Delta Y_2, \Delta X_3, \Delta Y_3, \Delta X_4, \Delta Y_4, \Omega_C)$$

$$\Omega = (80\ 06_x, 80_x, 06_x, 00_x, 00_x, 80_x, 06_x, 06_x, 5e_x, 06\ 12_x)$$

Com isso poderemos trabalhar para encontrar os bits das subchaves K6 e K5. Podemos visualizar um esquema para esta característica na figura 5.3.

A probabilidade total da característica é de 2^{-4}. Precisaremos de 2^4 pares de textos claro cuja diferença seja (80 06_x) para obter os valores de K5 e K6. Então examinamos as 2^8 combinações para essas subchaves e veremos quais as subchaves com maior contagem.

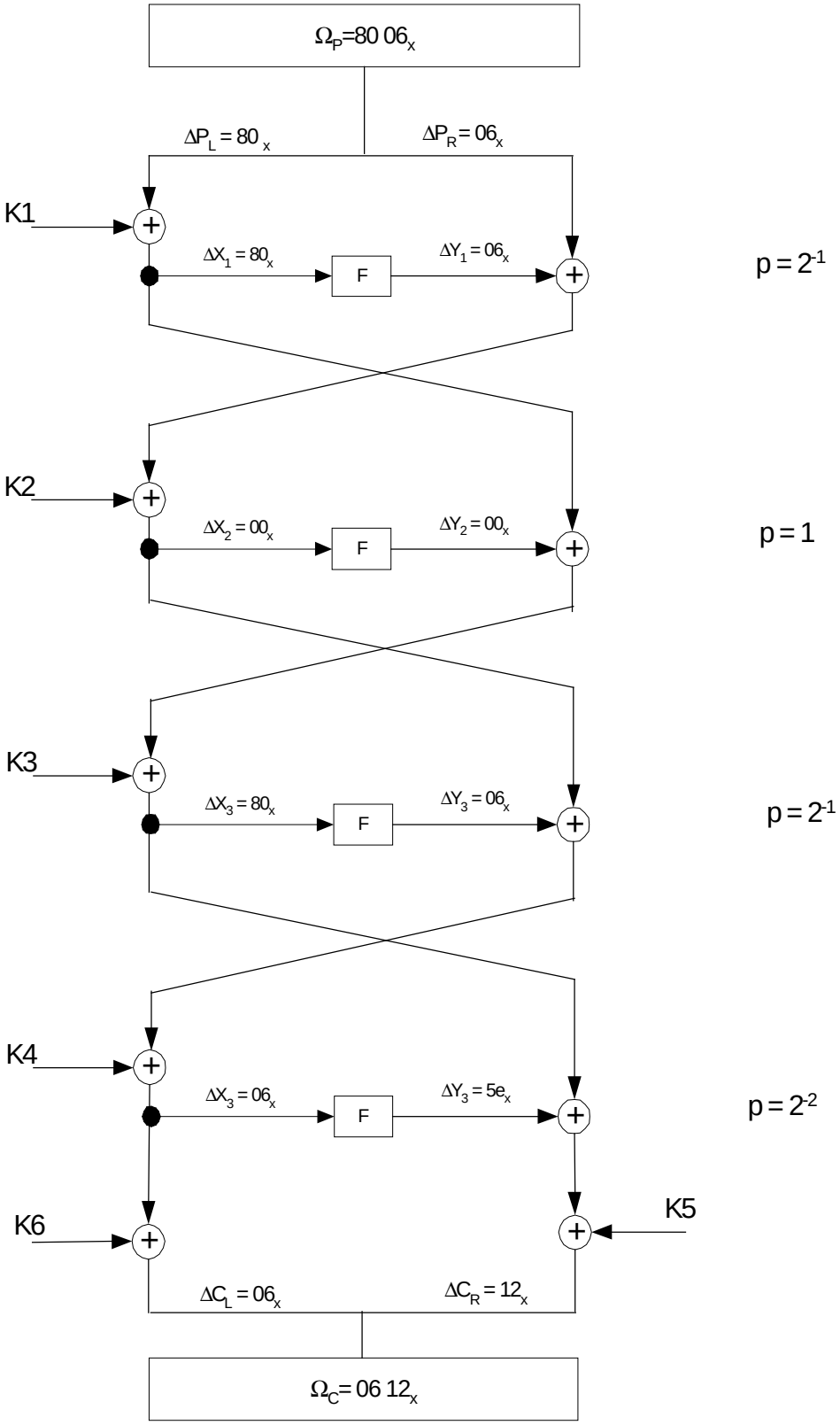

Figura 5.3 – Característica de 4 Rodadas para o Blowfish Reduzido.

5.2 Característica Diferencial para o Blowfish

Para construirmos a característica diferencial para o blowfish, vamos proceder da mesma maneira como fizemos com a versão reduzida, ou seja, iremos obter os valores para cada um dos S-Boxes. Estes valores estão mostrados no apêndice D.

As Tabelas de distribuição de diferenças seguem o mesmo padrão do Blowfish reduzido, sendo que a diferença (0_x) tem ocorrência de 256 e cada valor de entrada tem 128 valores diferentes com distribuição de 2 cada.

5.2.1 Blowfish reduzido a 4 rodadas

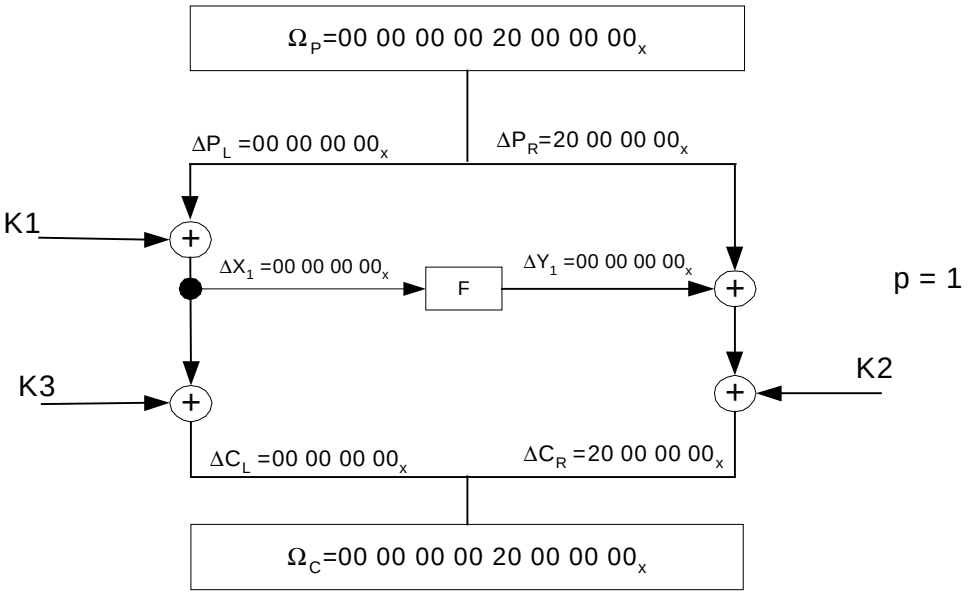

Figura 5.4 – Característica de uma rodada do BlowFish.

Uma característica trivial de uma rodada do Blowfish é mostrada na Figura 5.4. O texto claro é dividido em dois semi-blocos e quando a função F recebe o texto 00 00 00 00_x, gera uma palavra 00 00 00 00_x com probabilidade 1. Com a experiência feita no Blowfish vimos que esta característica ocorrerá.

A característica do Blowfish ficará da seguinte forma:

$$\Omega^1 = (\Omega_P, \Delta X_1, \Delta Y_1, \Omega_C)$$

$$\Omega^1 = (00\ 00\ 00\ 00\ 20\ 00\ 00\ 00_x, 00\ 00\ 00\ 00_x, 00\ 00\ 00\ 00_x, 00\ 00\ 00\ 00\ 20\ 00\ 00\ 00_x)$$

Podemos ilustrar como a diferença de texto claro pode tirar a influência das subchaves:

$$\Delta P = P^{'} \oplus P^{''} \qquad \Delta C = C^{'} \oplus C^{''}$$

$$\Delta P_L = \Delta X_1 = P_L^{'} \oplus P_L^{''} \qquad \Delta X_1 = X_1^{'} \oplus X_1^{''}$$

$$C_L^{'} = X_1^{'} \oplus K_3 \qquad C_L^{''} = X_1^{''} \oplus K_3$$

$$\Delta C_L = C_L^{'} \oplus C_L^{''} = X_1^{'} \oplus K_3 \oplus X_1^{''} \oplus K_3 = X_1^{'} \oplus X_1^{''} = \Delta X_1$$

$$C_R^{'} = P_R^{'} \oplus Y_1^{'} \oplus K_2 \qquad C_R^{''} = P_R^{''} \oplus Y_1^{''} \oplus K_2$$

$$\Delta C_R = C_R^{'} \oplus C_R^{''} = P_R^{'} \oplus Y_1^{'} \oplus K_2 \oplus P_R^{''} \oplus Y_1^{''} \oplus K_2 = P_R^{'} \oplus P_R^{''} \oplus Y_1^{'} \oplus Y_1^{''} = \Delta P_R \oplus \Delta Y_1$$

Como podemos verificar, a influência que temos na diferença do texto cifrado é somente da função F, ou seja, dos S-Boxes, que são dependentes da chave.

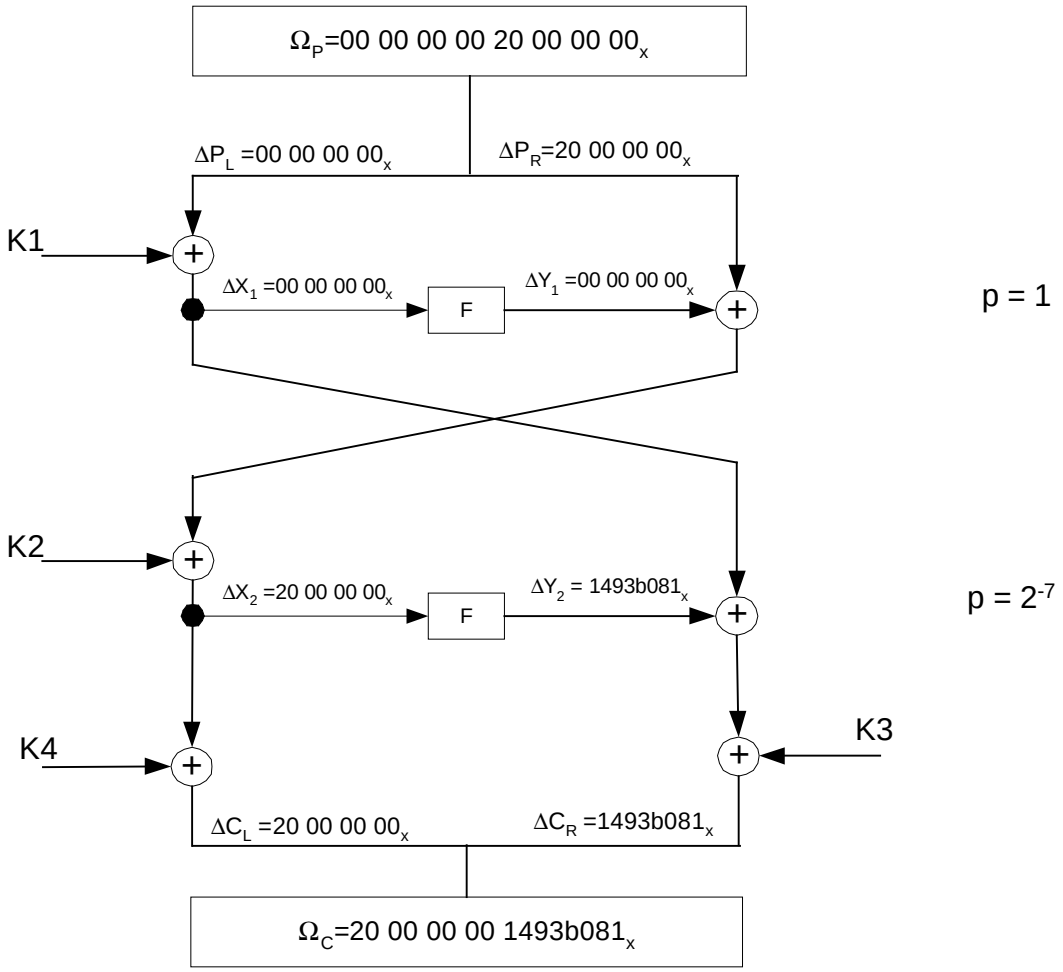

Figura 5.5 – Característica de 2 rodadas do Blowfish.

A característica de 2 rodadas do Blowfish é mostrada na figura 5.5. Observamos que uma simples mudança na palavra que entra na função F (20 00 00 00x) gera uma palavra de saída que influenciou todos os bits (14 93 b0 81x), com probabilidade 2^{-7}. Uma característica boa é que para os possíveis valores de saída da função F há uma distribuição uniforme, pois as probabilidades são iguais, não tendo um valor com probabilidade que se sobressaia.

A probabilidade da característica de 2 rodadas será 2^{-7}, pois não temos valores de alta probabilidade. Portanto, devemos tentar as 128 combinações para se chegar ao valor preciso da chave.

A característica do Blowfish para 2 rodadas fica da seguinte forma:

$$\Omega^2 = (\Omega_P, \Delta X_1, \Delta Y_1, \Delta X_2, \Delta Y_2, \Omega_C)$$

$$\Omega^2 = (00\ 00\ 00\ 00\ 20\ 00\ 00\ 00_x, 00\ 00\ 00\ 00_x, 00\ 00\ 00\ 00_x,$$
$$20\ 00\ 00\ 00_x, 14\ 93\ b0\ 81_x, 20\ 00\ 00\ 00\ 14\ 93\ b0\ 81_x)$$

com probabilidade da característica $P_2^{\Omega} = 2^{-7}$.

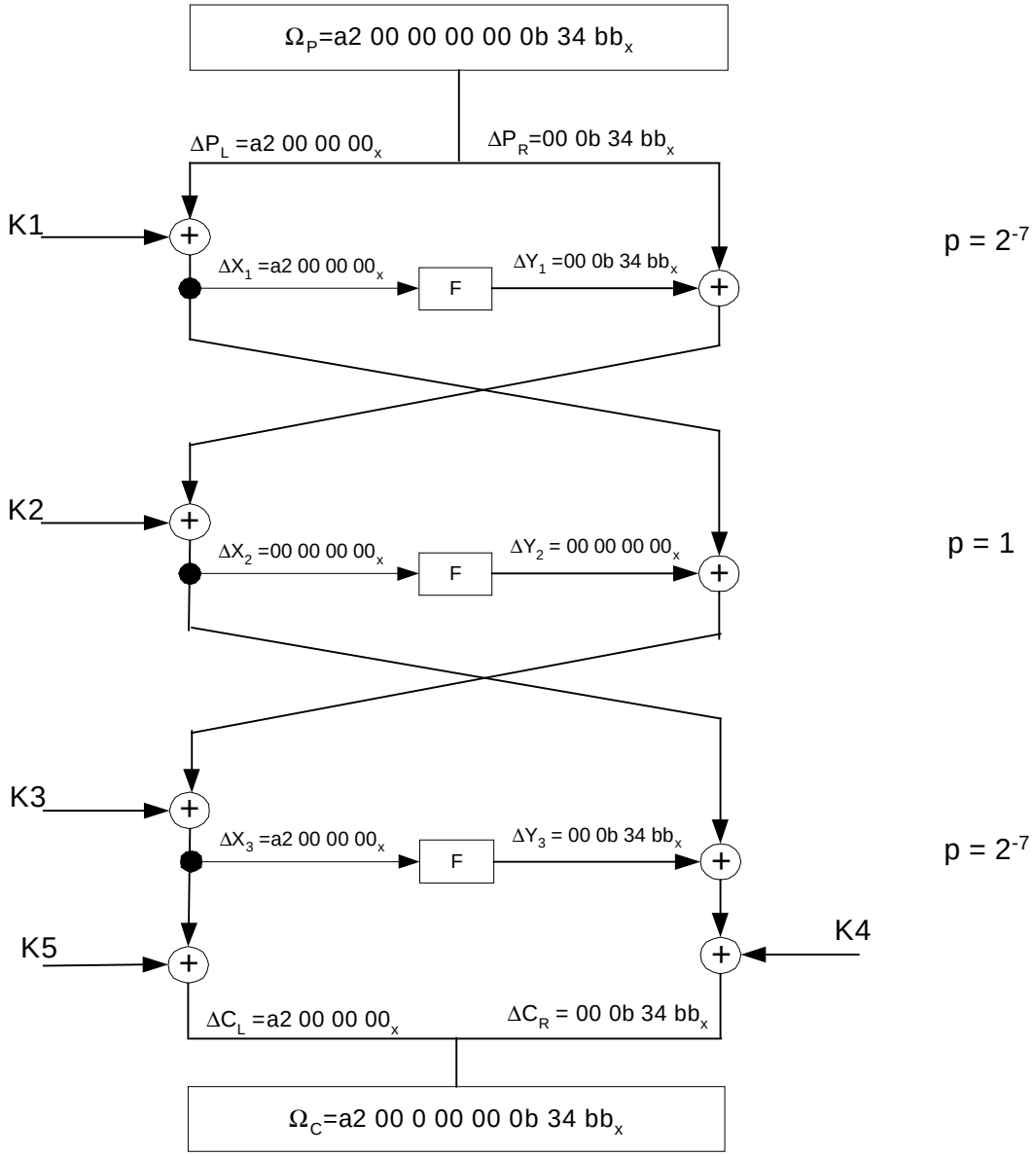

Figura 5.6 – Característica de 3 rodadas do Blowfish.

Para a característica de 3 rodadas mostrada na figura 5.6, temos uma probabilidade $P_{\Omega}^{3} = 2^{-14}$, a qual fica da seguinte forma:

$$\Omega^3 = (\Omega_P^3, \Omega_\Delta^3, \Omega_C^3) = (\Omega_P^3, \Delta X_1^3, \Delta Y_1^3, \Delta X_2^3, \Delta Y_2^3, \Delta X_3^3, \Delta Y_3^3, \Omega_C^3)$$
$$= (a2\ 00\ 00\ 00\ 00\ 0b\ 34\ bb_x, a2\ 00\ 00\ 00_x, 00\ 0b\ 34\ bb_x, 00\ 00\ 00\ 00_x,$$
$$00\ 00\ 00\ 00_x, a2\ 00\ 00\ 00_x, 00\ 0b\ 34\ bb_x, a2\ 00\ 00\ 00\ 00\ 0b\ 34\ bb_x)$$

Quatro contadores em paralelo de 8 bits é uma boa opção para descobrir os valores da chave quando todos os S-Boxes são influenciados. Observamos que com este tipo de esquema de contagem, podemos descobrir o valor exato da chave numa proporção muito menor, ou seja, por uma razão de 4.

Para a característica de 4 rodadas do Blowfish temos uma probabilidade $P_{\Omega}^{4} = 2^{-35}$ conforme Figura 5.7. E a característica fica da seguinte forma:

$$\Omega^4 = (\Omega_P^4, \Omega_\Delta^4, \Omega_C^4) = (\Omega_P^4, \Delta X_1^4, \Delta Y_1^4, \Delta X_2^4, \Delta Y_2^4, \Delta X_3^4, \Delta Y_3^4, \Delta X_4^4, \Delta Y_4^4, \Omega_C^4)$$
$$= (a2\ 00\ 00\ 00\ 00\ 0b\ 34\ bb_x,\ a2\ 00\ 00\ 00_x,\ 00\ 0b\ 34\ bb_x,\ 00\ 00\ 00\ 00_x,$$
$$00\ 00\ 00\ 00_x,\ a2\ 00\ 00\ 00_x,\ 00\ 0b\ 34\ bb_x,\ 00\ 0b\ 34\ bb_x,\ 7a\ d0\ 5f\ 0e_x,$$
$$00\ 0b\ 34\ bb\ d8\ d0\ 5f\ 0e_x)$$

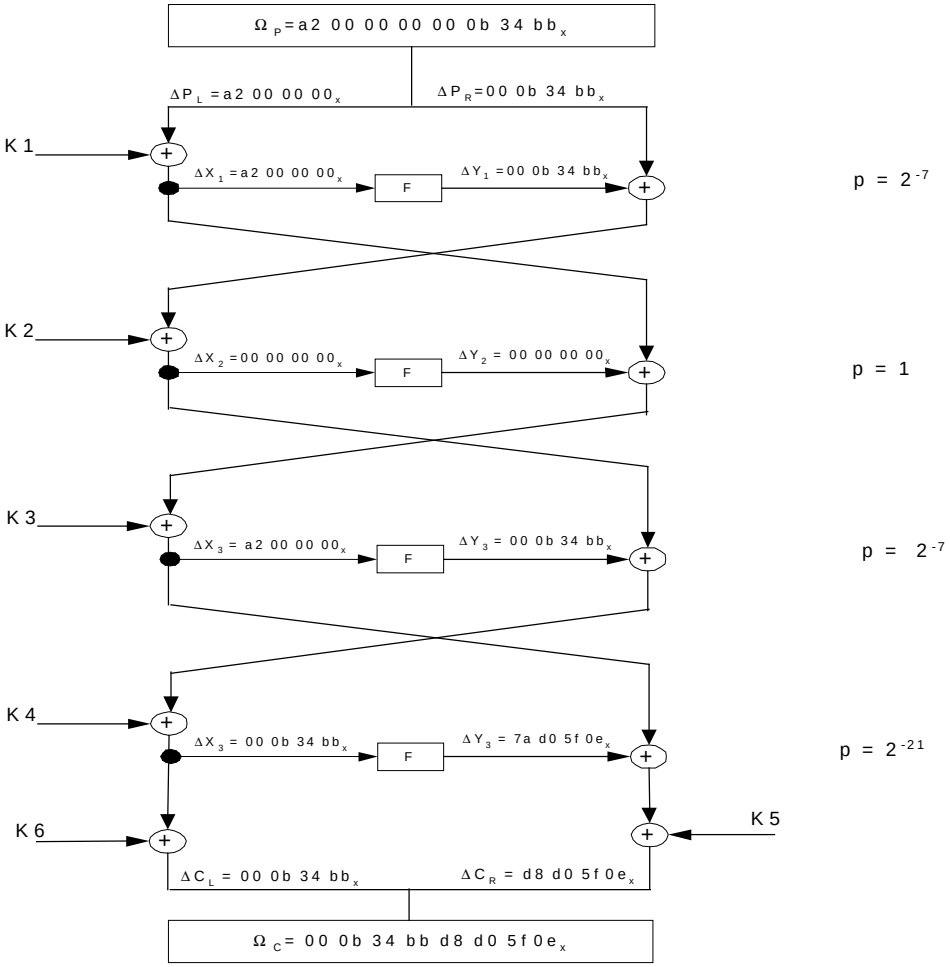

Figura 5.7 – Característica de 4 rodadas do Blowfish.

5.3 Chaves Fracas do Blowfish

5.3.1 Ataque de chave fraca com conhecimento de F

Definição 5.1 - Chave fraca significa que existe um S-Box, *S1*, que tem uma colisão. Isto é, existem dois bytes diferentes *a* e *a'* tais que *S1(a) = S1(a')*.

Seja δ a diferença ou-exclusivo das entradas *a* e *a'* de S1 (isto é $d = a \oplus a'$). Inicialmente considerando a característica iterativa mostrada na Figura 5.8. Assumindo que há apenas uma colisão para S1 com a diferença δ, a probabilidade desta característica é 2^{-7}.

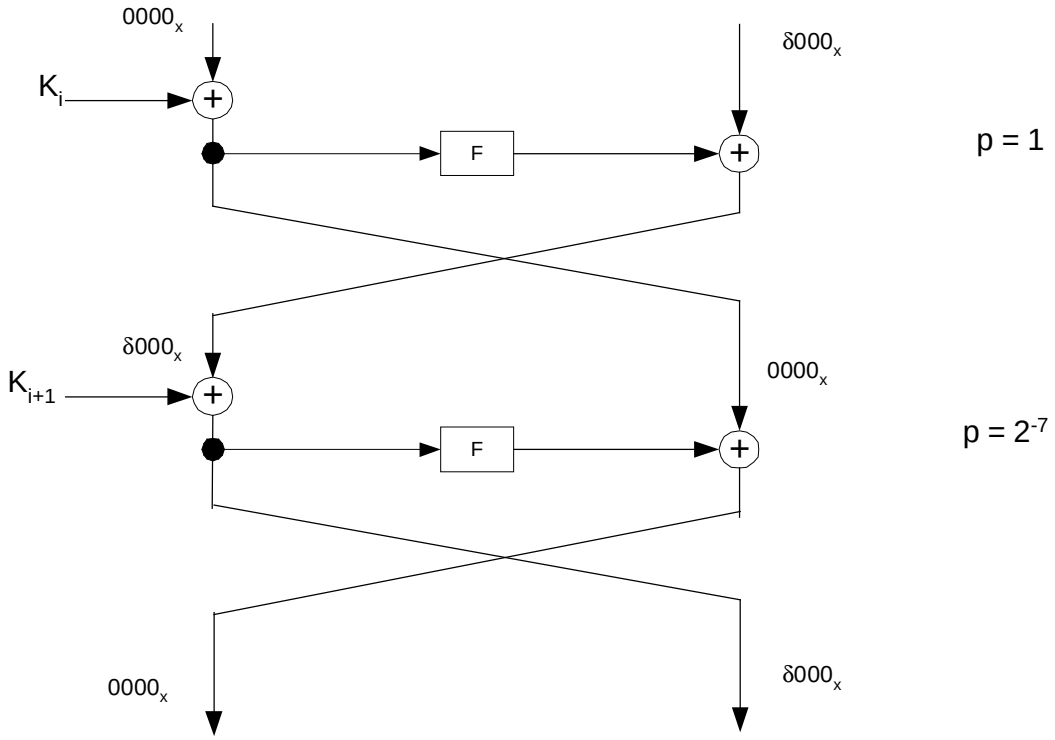

Figura 5.8 - Característica Iterativa [SV95].

Para o Blowfish reduzido a *t = 8* rodadas, iteramos esta característica 3 vezes como mostrado na Figura 5.9 (*xyzt* representa um valor indeterminado). A característica resultante tem probabilidade 2^{-21}. Então, tentando 2^{21} pares de texto claro escolhido com XOR [0000δ000], fica fácil detectar um par de texto cifrado (*C, C'*) com XOR [δ000*xyzt*].

Para um par aleatório, a probabilidade de obter tal par é 2^{-32}. Então, um par detectado com um bom XOR é certamente um par correto. Com tal par, e denotando *C = (L,R)*, tem-se:

$$F(L \oplus K_{10}) \oplus F(L \oplus K_{10} \oplus [d\,000]) = [xyzt]$$

foram tentadas exaustivamente todas as 2^{32} possibilidades para K_{10}, até que essa equação fosse satisfeita. É fácil verificar que Blowfish com *t* rodadas e um K_{t+2} conhecido é equivalente ao Blowfish com *t-1* rodadas. Então este ataque permite reduzir a cifra para *t = 7* rodadas.

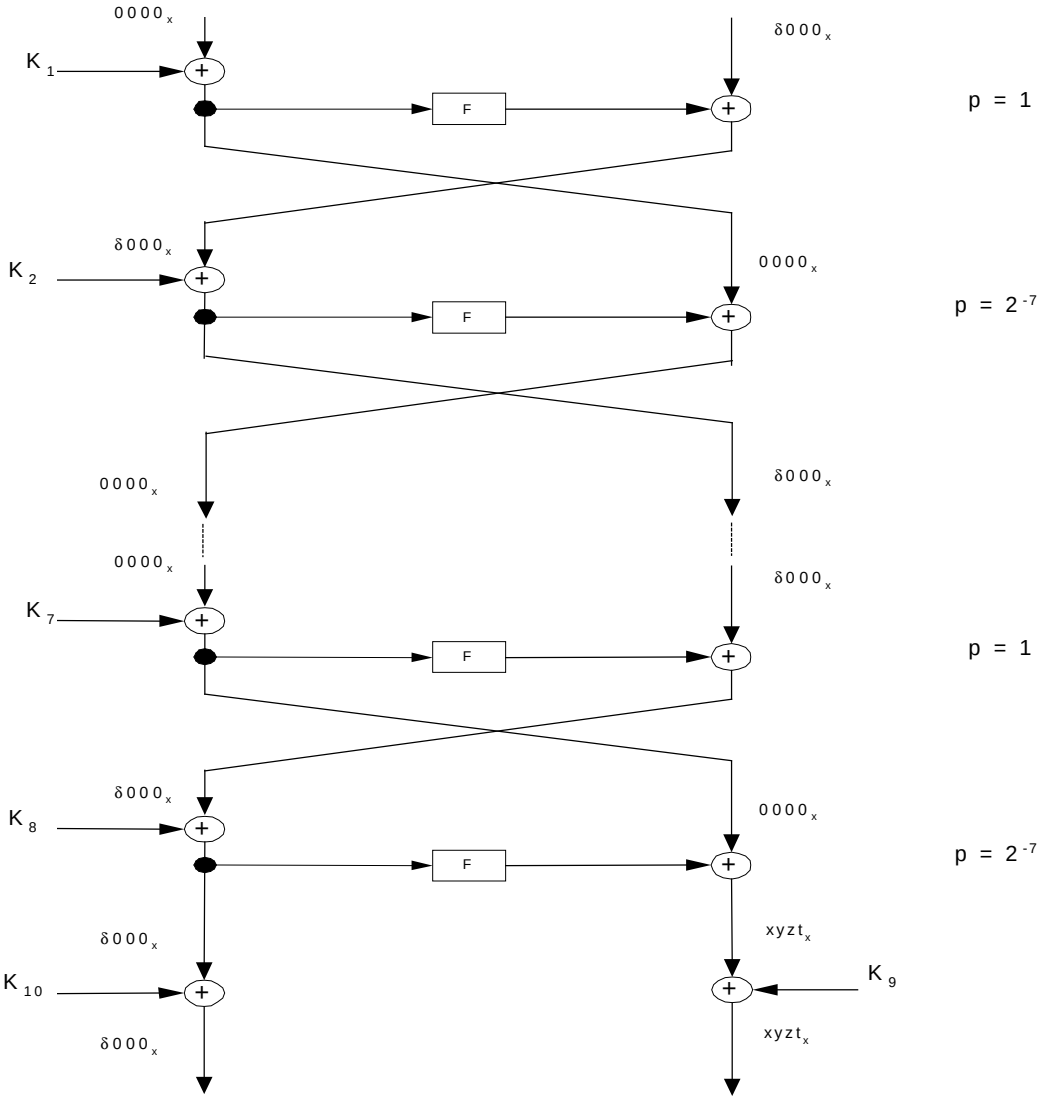

Figura 5.9 – Característica para 8 rodadas [SV95].

Porém, geralmente, para Blowfish com *t* rodadas, a mesma característica iterativa tem probabilidade $2^{-7\left\lceil \frac{t-2}{2} \right\rceil}$. Então, tenta-se $2^{7\left\lceil \frac{t-2}{2} \right\rceil}$ pares de texto claro escolhido e, para cada par de texto cifrado [δ000*xyzt*], lista-se os possíveis valores de K_{t+2}. Um par aleatório tem este XOR com probabilidade 2^{-32}, e cada par sugere em média um valor de K_{10}.

Para *t £ 10*, todos os pares que tem este XOR são corretos, mas para *t ³ 11*, foram obtidos $2^{7\times\left\lceil \frac{t-2}{2} \right\rceil-32}$ pares errados. Cada par errado sugere em média um valor aleatório para K_{t+2}. Então, tentando $3\times 2^{7\left\lceil \frac{t-2}{2} \right\rceil}$ pares de texto claro escolhido, foram obtidos três pares corretos que sugerem o mesmo valor com alta probabilidade, e nenhum outro valor é sugerido mais que duas vezes para *t £ 16*.

Como a complexidade do ataque em *t-1* rodadas é o mesmo que para *t* rodadas, se *t* é par, o número de texto claro escolhido requerido é $3\times 2^{2+7\times\left\lceil \frac{t-2}{2} \right\rceil}$. Para *t = 16*, isto é 3×2^{51}. Para *t = 8*, desde que não haja problemas com pares errados, o número de textos claro requerido é 2^{23}, podemos visualizar as quantidades de textos claro pela tabela 5.10.

Tabela 5.10 – Quantidade de Texto Claro Requerido

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}

5.3.2 Ataque de chave aleatória com conhecimento de F

O mapeamento $(a,b) \rightarrow S1(a) + S2(b)$ é uma função de 16 para 32 bits, então podemos ter uma colisão $S1(a) + S2(b) = S1(a') + S2(b')$ com alta probabilidade.

Seja $d = a \oplus a'$ e $m = b \oplus b'$, considerando a característica iterativa mostrada na figura 5.10. Supondo que há somente uma colisão para S1 + S2 com diferença $\delta\mu$, a probabilidade desta característica é 2^{-15}. Então, para o Blowfish com $t = 8$ rodadas, esta característica iterada como na figura 2.9 tem probabilidade 2^{-45}, e 2^{46} pares de texto claro escolhidos incluiem dois pares corretos e 2^{14} pares errados. Então, o valor correto para K_{10} pode ser o

único valor sugerido 2 vezes. O ataque com *t = 7* rodadas tem a mesma complexidade, então o número de texto claro requerido é 2^{48}.

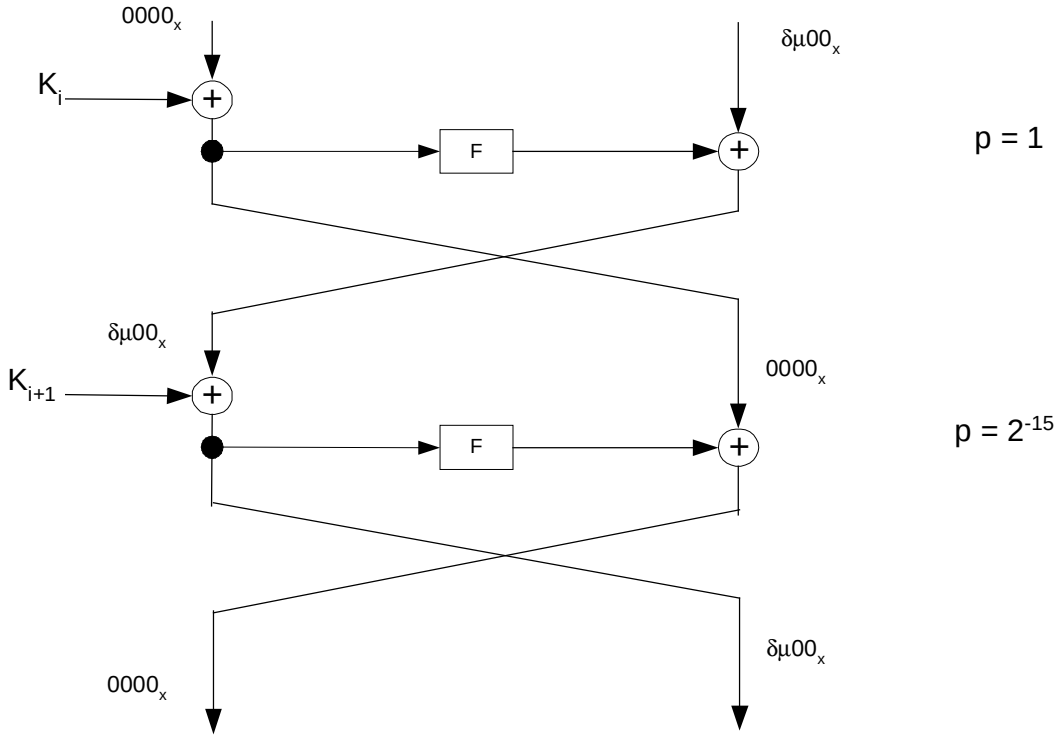

Figura 5.10 – Característica Iterativa com variação de 2 bytes (δμ) [SV95].

5.3.3 Detecção de Chaves Fracas

Para um S-Box S1 aleatório, a probabilidade que não haja colisão é [SV95]:

$$\prod_{i=0}^{2^8-1}(1-\frac{i}{2^{32}}) = \frac{2^{32}!}{2^{32\times 2^8}(2^{32}-2^8)!} \approx 1-2^{-17.0}$$

Então, para uma função F feita com S-Boxes aleatório, a probabilidade que exista uma colisão para um S-Box é $2^{-15.0}$. Então, uma chave das 2^{15} pode ser fraca.

Para distinguir que uma chave é fraca por um ataque de texto claro escolhido, primeiro consideramos S1 e usamos a característica da figura 5.9. Se os bytes B_1, B_2, B_3, B_4, B_6, B_7 e B_8 são aleatórios (sem o B_5), na estrutura de todos os 2^8 textos claro

$$P = [B_1B_2B_3B_4B_5B_6B_7B_8]$$

há 2^7 pares com o XOR correto. Seja

$$C = [C_1C_2C_3C_4C_5C_6C_7C_8]$$

a representação dos correspondentes textos cifrado. Em um par correto, observamos que

$$X = [(B_5 \oplus C_5)C_6C_7C_8]$$

deve ser o mesmo para ambas mensagens do par. Então, procurou-se por pares na estrutura que fizesse X colidir. Se não há pares corretos, isto ocorre com probabilidade aproximadamente 2^{-17}. Tentando 2^{14} estruturas, obtém-se um par correto com alta probabilidade, e nenhum par errado com probabilidade aproximadamente 2^{-3}. Então, com 2^{22} textos claro escolhidos, podemos detectar uma colisão em S1 e obter o XOR δ da colisão. O mesmo ataque vale para S2, S3 e S4.

6

CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

6.1 Introdução

Devido ao crescente aumento no uso de computadores conectados através de redes de comunicação, a segurança no tráfego de dados se tornou um problema real. Além da transmissão de informações por canais inseguros, empresas e pessoas armazenam informações de grande valor para as mesmas, que podem vir a ser acessadas por uma terceira parte não autorizada. Nesse cenário, sistemas proporcionando sigilo, autenticidade e integridade, podem ser projetados com o auxílio da criptografia, pois esta é uma ferramenta fundamental para a construção de sistemas seguros.

Essa dissertação apresentou um estudo sobre uma técnica de criptoanálise, denominada criptoanálise diferencial (CD), que é adequada para atacar cripto-sistemas de chave secreta baseados em cifras iterativas de bloco. Após uma breve revisão de alguns preliminares matemáticos, foram descritos os conceitos básicos da ciência da Criptologia, o que incluiu alguns dos principais cripto-sistemas de chave secreta em utilização comercial hoje em dia, tais como os sistemas DES, RC5, Blowfish e IDEA. A criptoanálise diferencial, em sua versão original proposta por Biham e Shamir em 1990, foi apresentada no capítulo 4, tendo seu funcionamento sido ilustrado através de uma cifra de chave secreta artificial, construída unicamente com o objetivo didático de melhor esclarecer alguns aspectos dessa técnica de criptoanálise. Essa cifra, que consiste de uma rede de substituição e permutação de 4 rodadas, contendo S-boxes do tipo 4x4, apesar de ter suas características simplificadas, contém os elementos essenciais para ilustrar adequadamente os conceitos mais relevantes da CD, constituindo-se portanto numa ferramenta útil para a disseminação da mesma.

No capitulo 5, a CD foi aplicada à cifra Blowfish. Inicialmente foi construída uma versão reduzida da cifra (Blowfish reduzido), cujos parâmetros mantiveram as mesmas proporções que aqueles do Blowfish. Os valores menores dos parâmetros do Blowfish reduzido permitiram uma exploração detalhada do funcionamento da cifra, principalmente no que diz respeito ao comportamento dos S-boxes. Assim, foi possível comprovar a existência de chaves fracas, que resultavam na construção de S-boxes com distribuição desbalanceada de diferenças, portanto tornando a cifra um alvo atrativo para a CD.

Ainda nesse capítulo exploramos a criptoanálise do Blowfish, através da construção de características diferenciais envolvendo até 4 rodadas.

Os programas fonte desenvolvidos nesse trabalho estão apresentados nos Apêndices ao final da dissertação.

6.2 Conclusões

6.2.1 Criptoanálise Diferencial

Em nosso estudo observamos que a criptoanálise diferencial de uma cifra de R rodadas emprega uma característica diferencial para determinar, por um processo de busca não exaustivo, os bits da subchave da rodada R-1. Uma vez executada essa etapa inicial, prosseguimos para determinar os bits da subchave da rodada anterior e assim por diante, até que todas as subchaves do cripto-sistema estejam determinadas, o que leva à quebra do mesmo. O aspecto essencial desse processo é o número de textos claro escolhidos que precisam ser determinados para levantar uma característica diferencial com alta probabilidade de ocorrência. Um indicador desses parâmetros é a chamada relação sinal ruído da característica, que indica a quantidade de pares diferenciais necessários para a criptoanálise bem sucedida da cifra. Fica evidente então que as chances de sucesso do ataque dependem fortemente dos recursos computacionais à disposição do criptoanalista, como era de se esperar.

Nesse contexto, observamos também que é possível empregar estratégias de projeto de uma cifra, que a tornem imune a CD. Dentre essas técnicas podemos destacar:

i - A construção de S-boxes com característica diferencial balanceada.

ii - A construção de S-boxes dependentes da chave.

iii- A utilização, na função F do algoritmo de cifragem, de rotações dependentes dos dados.

O estudo dos cripto-sistemas apresentados no capítulo 3 demonstrou que técnicas desse tipo tem norteado o projeto não apenas desses sistemas, mas também de padrões de cifragem de dados mais recentes, como o AES/RIJNDAEL. Esse fato é, de per sí, um indicativo claro da importancia da CD na área de segurança de dados.

Os resultados dos experimentos conduzidos ao longo da realização desse trabalho, questionam a eficácia da técnica (ii) mencionada acima, que foi utilizada no cripto-sistema Blowfish.

6.2.2 O Cripto-Sistema Blowfish

Na investigação levada a cabo com o Blowfish reduzido, mostramos que o uso de S-boxes dependentes da chave não leva necessariamente à construção de bons S-boxes, isto é, não garante nem a inexistência de chaves fracas, nem a imunidade à CD. De fato, esse é um resultado de certa forma esperado, pois a construção dos S-boxes depende inteiramente do processo de inicialização e do próprio algoritmo de cifragem, não implicando necessariamente que a parte não linear da cifra atenda a algum critério específico de projeto.

Em relação ao Blowfish, conjecturamos que resultados semelhantes irão ocorrer e que os S-boxes 8x32 desse sistema compartilham deficiências similares àquelas observadas na versão reduzida. Até o presente momento, a limitação de recursos computacionais ainda tem sido um obstáculo para que os resultados de uma análise exaustiva da distribuição das diferenciais de saída dos S-boxes do Blowfish possam ser apresentados. Diante desse cenário, concluímos que S-boxes fixos bem projetados, tais como os usados no DES e no RIJNDAEL, podem vir a ter um melhor desempenho, em relação àqueles que mudam com a chave secreta do sistema.

Os resultados que obtivemos com a CD do Blowfish, no capítulo 5, mostram que uma versão reduzida com 4 rodadas do mesmo, pode ser atacada com sucesso. Para a chave escolhida, a partir da quinta rodada, a probabilidade da característica diminui o suficiente para que o número de textos claro requeridos para o ataque torne-o inviável.

6.3 Sugestões para Trabalhos Futuros

Esta dissertação serve como ponto de partida do conhecimento sobre criptoanálise diferencial e o seu emprego em outros cripto-sistemas servirá para o aprofundamento dos estudos dessa importante técnica de criptoanálise. Além dessa extensão natural desse estudo, propomos a investigação das seguintes questões relativas à CD:

1) Construir características diferenciais baseadas nas chamadas diferenças impossíveis, isto é, padrões de diferença na saída dos S-boxes que nunca ocorrem. Comparar essa possibilidade com o método usual estudado nesse trabalho e estabelecer critérios que indiquem uma maior viabilidade de aplicação de uma das alternativas.

2) Considerar a CD de cripto-sistemas em que o conceito de diferença empregado não seja o de ou-exclusivo das entradas de um S-box.

3) Analisar as características de imunidade a CD de uma nova versão reduzida do Blowfish, com comprimento de chave 32 e S-boxes do tipo 4x16.

4) Construir uma tabela de freqüência relativa das diferenças de saída dos S-boxes do Blowfish, considerando todo o espaço de chaves.

5) Por fim, sugerimos a análise e construção de cripto-sistemas que sejam imunes à criptoanálise diferencial.

[MR00] S. MURPHY e M.J.B. ROBSHAW, Key-dependent S-Boxes, Differential Cryptanalysis, and Twofish. www.isg.rhul.ac.uk/~sean/tf_dckdsb.ps

[BS90] E. BIHAM e A. SHAMIR, Differential cryptanalysis of DES-like cryptosystems. The Weizmann Institute of sciense Department of apllied mathematics, Dissertação de mestrado, 1990.

[BS94] B. SCHNEIER, Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993), Springer-Verlag, pp. 1981-204, 1994.

[BS95] B. SCHNEIER, The Blowfish Encryption Algorithm - One Year Later. Dr. Dobb's Journal, September 1995.

[HE00] H. M. HEYS. A Tutorial on Linear and Differential Cryptanalysis. Eletrical ana computer engineering, Faculty of engineering and apllied science, 2000.

[SV95] S. VAUDENAY. On the Weak Keys of Blowfish. Ecole Normale Superieure, Département de Mathématiques et Informatique, Liens - 95-27. Novembro de 1995.

[XLS91] LAI, X., MASSEY, J.L. e MURPHY, S. Markov Ciphers and Differential Cryptanalysis, Advances in Cryptology – EUROCRYPT'91, Springer-Verlag, 1991.

[DWH76] DIFFIE, Whitfield e HELLMAN, Martin E. New Directions in Cryptography. IEEE Trans. Inform. Theory. IT-22, 6 (Nov. 1976), 644-654.

[DDW89] DAVIES, Donald Watts e PRICE, W. L. Security for Computers Networks: an introduction to data security in teleprocessing and eletronic funds transfer. 2ª Edição.

[NKY00] TORII, Naoya e YOKOYAMA, Kazuhiro. Elliptic Curve Cryptosystem. FUJITSU Sci. Tech. J., 36, 2, pp. 140-146 (December 2000).

[MOV96] MENEZES, A., OORSCHOT, P. Van e VANSTONE, S. Applied

Cryptography, by CRC Press, 1996.

[SGJ92] SIMMONS, Gustavus J. Comtemporary cryptology: the science of information integrity. IEEE Press. 1992.

[DDE82] DENNING, Dorothy E., Cryptography and data security. Addison-Wesley Publishing Company, Inc. 1982.

[RSA78] RIVEST, R.L., SHAMIR, A. e ADLEMAN, L.M. "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, volume 21, pages 120-126, 1978.

[FIP81] National Bureau of Standards. DES modes of operation. Federal Information Processing Standard (FIPS), publication 81, National Bureau of Standards, U.S. Departament of Commerce, Washington D.C., December 1980.

[MH79] M Hellman "DES will be totally insecure within ten years". IEEE Spectrum 16(7), Jul 1979, pp 31-41

[FIP46] National Bureau of Standards. Data Encryption Standard (DES). Federal Information Processing Standard (FIPS), publication 46-3, National Bureau of Standards, U.S. Departament of Commerce, Washington D.C., October 1999.

[MO96] MENEZES, A. J. e OORSCHOT, P. C. Van. Handbook of Applied Cryptography, CRC Press, 1996.

[RR] RIVEST, Ronald L. The RC5 Encryption Algorithm. In the *Proceedings of the Second Intenational Workshop on Fast Software Encryption (FSE) 1994*, p86–96.

[LK97] KNUDSEN, L. R.. Block Ciphers – a survey. To appear in the proceedings of the international Course on the State of the Art and Evolution on Computer Security and Industrial Cryptography 1997, to be published in the LNCS Series from Springer Verlag.

[XL92] Lai, Xuejia. On the design na security of block ciphers. Zürich, Techn. Hochsch., Diss., 1992. Vol 1, ETH Series in Information Processing.

[JD92] Durbin, J. R.. Modern Algebra – An Introduction, 3ª Edição, John Wisley & Sons, 1992.

[RM87] McEliece, R. J.. Finite Fields for Computer Scientists and Engineers,

Kluwer Academic Publishers, 1987.

[MO96] Menezes, A. J. e Oorschot, P. C. Van. Handbook of Applied Cryptography, CRC Press, 1996.

[LP98] Lidl, Rudolf e Pilz, Günher. Applied Abstract Álgebra. 2ª Edição. Springer-Verlag, New York, 1998.

[WCK95] Wicker, Stephen B.. Error Control Systems for Digital Communication and Storage. Englewood Cliffs: Prentice Hall, 1995.

[NKY00] TORII, Naoya e YOKOYAMA, Kazuhiro. Elliptic Curve Cryptosystem. FUJITSU Sci. Tech. J., 36, 2, pp. 140-146 (December 2000).

Apêndice A: Algoritmo do Blowfish Reduzido em Linguagem C

A.1 Listagem do programa Blowfish.cpp

```
/**********************blowfish.c**********************/
/* Blowfish Reduzido: Cifra blocos de textos claro de 16 bits */
/* Tamanho da chave: 16 bits. */

#include <iostream>

using std::cout;
using std::cin;
using std::ios;
using std::cerr;
using std::endl;


#include <fstream>

using std::ofstream;

#include <iomanip>

using std::ios;
using std::setiosflags;
using std::hex;
using std::dec;
using std::setw;

#include <cstdlib>
#include <assert.h>


#include "blowfish.h"
#include "bf_tab.h"              /* P-box P-array, S-box  */

#define S(x,i) (bf_S[i][x.w.byte##i])
#define bf_F(x) (((S(x,0) + S(x,1)) ^ S(x,2)) + S(x,3))
#define ROUND(a,b,n) (a.word ^= bf_F(b) ^ bf_P[n])

inline
```

```
void Blowfish_encipher(UBYTE_08bits *xl, UBYTE_08bits *xr)
{
  union aword  Xl;
  union aword  Xr;

  Xl.word = *xl;
  Xr.word = *xr;

  Xl.word ^= bf_P[0];
  ROUND (Xr, Xl, 1);  ROUND (Xl, Xr, 2);
  ROUND (Xr, Xl, 3);  ROUND (Xl, Xr, 4);
  ROUND (Xr, Xl, 5);  ROUND (Xl, Xr, 6);
  ROUND (Xr, Xl, 7);  ROUND (Xl, Xr, 8);
  ROUND (Xr, Xl, 9);  ROUND (Xl, Xr, 10);
  ROUND (Xr, Xl, 11); ROUND (Xl, Xr, 12);
  ROUND (Xr, Xl, 13); ROUND (Xl, Xr, 14);
  ROUND (Xr, Xl, 15); ROUND (Xl, Xr, 16);
  Xr.word ^= bf_P[5];
  Xl.word ^= bf_P[6];

  *xr = Xl.word;
  *xl = Xr.word;
}

void Blowfish_decipher(UBYTE_08bits *xl, UBYTE_08bits *xr)
{
   union aword  Xl;
   union aword  Xr;

   Xl.word = *xl;
   Xr.word = *xr;

   Xl.word ^= bf_P[17];
   ROUND (Xr, Xl, 16);  ROUND (Xl, Xr, 15);
   ROUND (Xr, Xl, 14);  ROUND (Xl, Xr, 13);
   ROUND (Xr, Xl, 12);  ROUND (Xl, Xr, 11);
   ROUND (Xr, Xl, 10);  ROUND (Xl, Xr, 9);
   ROUND (Xr, Xl, 8);   ROUND (Xl, Xr, 7);
   ROUND (Xr, Xl, 6);   ROUND (Xl, Xr, 5);
   ROUND (Xr, Xl, 4);   ROUND (Xl, Xr, 3);
   ROUND (Xr, Xl, 2);   ROUND (Xl, Xr, 1);
   Xr.word ^= bf_P[0];

   *xl = Xr.word;
   *xr = Xl.word;
}
```

```
/* FIXME: Blowfish_Initialize() ??? */
short InitializeBlowfish(UBYTE_08bits key[], short keybytes)
{
  short         i;          /* FIXME: unsigned int, char? */
  short         j;          /* FIXME: unsigned int, char? */
  UBYTE_08bits  data;
  UBYTE_08bits  datal;
  UBYTE_08bits  datar;
  union aword temp;


  j = 0;
  for (i = 0; i < bf_N + 2; ++i) {
    temp.word = 0;
    temp.w.byte0 = key[j];
    temp.w.byte1 = key[(j+1)%keybytes];
    temp.w.byte2 = key[(j+2)%keybytes];
    temp.w.byte3 = key[(j+3)%keybytes];
    data = temp.word;
    bf_P[i] = bf_P[i] ^ data;
    j = (j + 4) % keybytes;
  }

  datal = 0x00;
  datar = 0x00;

  for (i = 0; i < bf_N + 2; i += 2) {
    Blowfish_encipher(&datal, &datar);

    bf_P[i] = datal;
    bf_P[i + 1] = datar;
  }

  for (i = 0; i < 4; ++i) {

    for (j = 0; j < 4; j += 2) {

      Blowfish_encipher(&datal, &datar);


      bf_S[i][j] = datal;

      bf_S[i][j + 1] = datar;

    }
  }
```

```
  return 0;
}
```

A.2 Listagem do programa Blowfish.h

```
#define subkeyfilename   "Blowfish.dat"

#define UWORD_32bits  unsigned short
#define UWORD_16bits  unsigned short
#define UBYTE_08bits  unsigned char

#define ORDER_DCBA true
/* choose a byte order for your hardware */

/* DCBA - little endian - intel */
/* DCBA - little endian - intel */

#ifdef ORDER_DCBA
union aword {
  UBYTE_08bits word;
  UBYTE_08bits byte [4];
  struct {
    unsigned int byte3:2;
    unsigned int byte2:2;
    unsigned int byte1:2;
    unsigned int byte0:2;
  } w;
};
#endif  /* ORDER_DCBA */


short opensubkeyfile(void);
unsigned long F(unsigned long x);
void Blowfish_encipher(UBYTE_08bits *xl, UBYTE_08bits *xr);
void Blowfish_decipher(UBYTE_08bits *xl, UBYTE_08bits *xr);
short InitializeBlowfish(unsigned char key[], short keybytes);
```

A.3 Listagem do programa bf_tab.h

```
/* bf_tab.h: Blowfish P-box and S-box tables */
/* reduzida para entrada de 2 e saida para 8 bits. */

static UWORD_32bits bf_P[bf_N + 2] = {
  0x24, 0x3f, 0x6a, 0x88,
```

```
  0x85, 0xa3, 0x08, 0xd3,
  0x13, 0x19, 0x8a, 0x2e,
  0x03, 0x70, 0x73, 0x44,
  0xa4, 0x09,
};
static UWORD_32bits bf_S[4][4] = {
  0xd1, 0x31, 0x0b, 0xa6,
  0x98, 0xdf, 0xb5, 0xac,
  0x2f, 0xfd, 0x72, 0xdb,
  0xd0, 0x1a, 0xdf, 0xb7,
};
```

Apêndice B: Algoritmo do Blowfish em Linguagem C

B.1 Listagem do programa Blowfish.cpp

```
/**********************blowfish.c**********************/

/* TODO: test with zero length key */
/* TODO: test with a through z as key and plain text */
/* TODO: make this byte order independent */

#include <iostream>

using std::cout;
using std::cin;
using std::ios;
using std::cerr;
using std::endl;


#include <fstream>

using std::ofstream;

#include <iomanip>

using std::ios;
using std::setiosflags;
using std::hex;
using std::dec;
using std::setw;

#include <cstdlib>
#include <assert.h>


#include "blowfish.h"
#include "bf_tab.h"              /* P-box P-array, S-box  */

#define S(x,i) (bf_S[i][x.w.byte##i])
#define bf_F(x) (((S(x,0) + S(x,1)) ^ S(x,2)) + S(x,3))
#define ROUND(a,b,n) (a.word ^= bf_F(b) ^ bf_P[n])

inline
```

```
void Blowfish_encipher(UWORD_32bits *xl, UWORD_32bits *xr)
{
  union aword  Xl;
  union aword  Xr;

  Xl.word = *xl;
  Xr.word = *xr;

  Xl.word ^= bf_P[0];
  ROUND (Xr, Xl, 1);  ROUND (Xl, Xr, 2);
  ROUND (Xr, Xl, 3);  ROUND (Xl, Xr, 4);
  ROUND (Xr, Xl, 5);  ROUND (Xl, Xr, 6);
  ROUND (Xr, Xl, 7);  ROUND (Xl, Xr, 8);
  ROUND (Xr, Xl, 9);  ROUND (Xl, Xr, 10);
  ROUND (Xr, Xl, 11); ROUND (Xl, Xr, 12);
  ROUND (Xr, Xl, 13); ROUND (Xl, Xr, 14);
  ROUND (Xr, Xl, 15); ROUND (Xl, Xr, 16);
  Xr.word ^= bf_P[17];
  Xl.word ^= bf_P[18];

  *xr = Xl.word;
  *xl = Xr.word;
}

void Blowfish_decipher(UWORD_32bits *xl, UWORD_32bits *xr)
{
   union aword  Xl;
   union aword  Xr;

   Xl.word = *xl;
   Xr.word = *xr;

   Xl.word ^= bf_P[17];
   ROUND (Xr, Xl, 16);  ROUND (Xl, Xr, 15);
   ROUND (Xr, Xl, 14);  ROUND (Xl, Xr, 13);
   ROUND (Xr, Xl, 12);  ROUND (Xl, Xr, 11);
   ROUND (Xr, Xl, 10);  ROUND (Xl, Xr, 9);
   ROUND (Xr, Xl, 8);   ROUND (Xl, Xr, 7);
   ROUND (Xr, Xl, 6);   ROUND (Xl, Xr, 5);
   ROUND (Xr, Xl, 4);   ROUND (Xl, Xr, 3);
   ROUND (Xr, Xl, 2);   ROUND (Xl, Xr, 1);
   Xr.word ^= bf_P[0];

   *xl = Xr.word;
   *xr = Xl.word;
}
```

```
/* FIXME: Blowfish_Initialize() ??? */
short InitializeBlowfish(UBYTE_08bits key[], short keybytes)
{
  short         i;          /* FIXME: unsigned int, char? */
  short         j;          /* FIXME: unsigned int, char? */
  UWORD_32bits  data;
  UWORD_32bits  datal;
  UWORD_32bits  datar;
  union aword temp;

  j = 0;
  for (i = 0; i < bf_N + 2; ++i) {
    temp.word = 0;
    temp.w.byte0 = key[j];
    temp.w.byte1 = key[(j+1)%keybytes];
    temp.w.byte2 = key[(j+2)%keybytes];
    temp.w.byte3 = key[(j+3)%keybytes];
    data = temp.word;
    bf_P[i] = bf_P[i] ^ data;
    j = (j + 4) % keybytes;
  }

  datal = 0x00000000;
  datar = 0x00000000;

  for (i = 0; i < bf_N + 2; i += 2) {
    Blowfish_encipher(&datal, &datar);

    bf_P[i] = datal;
    bf_P[i + 1] = datar;
  }

  for (i = 0; i < 4; ++i) {

    for (j = 0; j < 256; j += 2) {

      Blowfish_encipher(&datal, &datar);


      bf_S[i][j] = datal;

      bf_S[i][j + 1] = datar;

    }
  }

  return 0;
```

```
}
```

B.2 Listagem do programa Blowfish.h

```
/**********************blowfish.h*********************/

/* $Id: blowfish.h,v 1.3 1995/01/23 12:38:02 pr Exp pr $*/

#define MAXKEYBYTES 56                  /* 448 bits */
#define bf_N            16
#define noErr           0
#define DATAERROR         -1
#define KEYBYTES        8
#define subkeyfilename   "Blowfish.dat"

#define UWORD_32bits  unsigned long
#define UWORD_16bits  unsigned short
#define UBYTE_08bits  unsigned char

#define ORDER_DCBA true
/* choose a byte order for your hardware */

/* DCBA - little endian - intel */
/* DCBA - little endian - intel */

#ifdef ORDER_DCBA
union aword {
  UWORD_32bits word;
  UBYTE_08bits byte [4];
  struct {
    unsigned int byte3:8;
    unsigned int byte2:8;
    unsigned int byte1:8;
    unsigned int byte0:8;
  } w;
};
#endif  /* ORDER_DCBA */


short opensubkeyfile(void);
unsigned long F(unsigned long x);
void Blowfish_encipher(unsigned long *xl, unsigned long *xr);
void Blowfish_decipher(unsigned long *xl, unsigned long *xr);
short InitializeBlowfish(unsigned char key[], short keybytes);
```

B.2 Listagem do programa bf_tab.h

```
/* bf_tab.h: Blowfish P-box and S-box tables */

static UWORD_32bits bf_P[bf_N + 2] = {
  0x243f6a88, 0x85a308d3, 0x13198a2e, 0x03707344,
  0xa4093822, 0x299f31d0, 0x082efa98, 0xec4e6c89,
  0x452821e6, 0x38d01377, 0xbe5466cf, 0x34e90c6c,
  0xc0ac29b7, 0xc97c50dd, 0x3f84d5b5, 0xb5470917,
  0x9216d5d9, 0x8979fb1b,
};
static UWORD_32bits bf_S[4][256] = {
  0xd1310ba6, 0x98dfb5ac, 0x2ffd72db, 0xd01adfb7,
  0xb8e1afed, 0x6a267e96, 0xba7c9045, 0xf12c7f99,
  0x24a19947, 0xb3916cf7, 0x0801f2e2, 0x858efc16,
  0x636920d8, 0x71574e69, 0xa458fea3, 0xf4933d7e,
  0x0d95748f, 0x728eb658, 0x718bcd58, 0x82154aee,
  0x7b54a41d, 0xc25a59b5, 0x9c30d539, 0x2af26013,
  0xc5d1b023, 0x286085f0, 0xca417918, 0xb8db38ef,
  0x8e79dcb0, 0x603a180e, 0x6c9e0e8b, 0xb01e8a3e,
  0xd71577c1, 0xbd314b27, 0x78af2fda, 0x55605c60,
  0xe65525f3, 0xaa55ab94, 0x57489862, 0x63e81440,
  0x55ca396a, 0x2aab10b6, 0xb4cc5c34, 0x1141e8ce,
  0xa15486af, 0x7c72e993, 0xb3ee1411, 0x636fbc2a,
  0x2ba9c55d, 0x741831f6, 0xce5c3e16, 0x9b87931e,
  0xafd6ba33, 0x6c24cf5c, 0x7a325381, 0x28958677,
  0x3b8f4898, 0x6b4bb9af, 0xc4bfe81b, 0x66282193,
  0x61d809cc, 0xfb21a991, 0x487cac60, 0x5dec8032,
  0xef845d5d, 0xe98575b1, 0xdc262302, 0xeb651b88,
  0x23893e81, 0xd396acc5, 0x0f6d6ff3, 0x83f44239,
  0x2e0b4482, 0xa4842004, 0x69c8f04a, 0x9e1f9b5e,
  0x21c66842, 0xf6e96c9a, 0x670c9c61, 0xabd388f0,
  0x6a51a0d2, 0xd8542f68, 0x960fa728, 0xab5133a3,
  0x6eef0b6c, 0x137a3be4, 0xba3bf050, 0x7efb2a98,
  0xa1f1651d, 0x39af0176, 0x66ca593e, 0x82430e88,
  0x8cee8619, 0x456f9fb4, 0x7d84a5c3, 0x3b8b5ebe,
  0xe06f75d8, 0x85c12073, 0x401a449f, 0x56c16aa6,
  0x4ed3aa62, 0x363f7706, 0x1bfedf72, 0x429b023d,
  0x37d0d724, 0xd00a1248, 0xdb0fead3, 0x49f1c09b,
  0x075372c9, 0x80991b7b, 0x25d479d8, 0xf6e8def7,
  0xe3fe501a, 0xb6794c3b, 0x976ce0bd, 0x04c006ba,
  0xc1a94fb6, 0x409f60c4, 0x5e5c9ec2, 0x196a2463,
  0x68fb6faf, 0x3e6c53b5, 0x1339b2eb, 0x3b52ec6f,
  0x6dfc511f, 0x9b30952c, 0xcc814544, 0xaf5ebd09,
  0xbee3d004, 0xde334afd, 0x660f2807, 0x192e4bb3,
```

0xc0cba857, 0x45c8740f, 0xd20b5f39, 0xb9d3fbdb,
0x5579c0bd, 0x1a60320a, 0xd6a100c6, 0x402c7279,
0x679f25fe, 0xfb1fa3cc, 0x8ea5e9f8, 0xdb3222f8,
0x3c7516df, 0xfd616b15, 0x2f501ec8, 0xad0552ab,
0x323db5fa, 0xfd238760, 0x53317b48, 0x3e00df82,
0x9e5c57bb, 0xca6f8ca0, 0x1a87562e, 0xdf1769db,
0xd542a8f6, 0x287effc3, 0xac6732c6, 0x8c4f5573,
0x695b27b0, 0xbbca58c8, 0xe1ffa35d, 0xb8f011a0,
0x10fa3d98, 0xfd2183b8, 0x4afcb56c, 0x2dd1d35b,
0x9a53e479, 0xb6f84565, 0xd28e49bc, 0x4bfb9790,
0xe1ddf2da, 0xa4cb7e33, 0x62fb1341, 0xcee4c6e8,
0xef20cada, 0x36774c01, 0xd07e9efe, 0x2bf11fb4,
0x95dbda4d, 0xae909198, 0xeaad8e71, 0x6b93d5a0,
0xd08ed1d0, 0xafc725e0, 0x8e3c5b2f, 0x8e7594b7,
0x8ff6e2fb, 0xf2122b64, 0x8888b812, 0x900df01c,
0x4fad5ea0, 0x688fc31c, 0xd1cff191, 0xb3a8c1ad,
0x2f2f2218, 0xbe0e1777, 0xea752dfe, 0x8b021fa1,
0xe5a0cc0f, 0xb56f74e8, 0x18acf3d6, 0xce89e299,
0xb4a84fe0, 0xfd13e0b7, 0x7cc43b81, 0xd2ada8d9,
0x165fa266, 0x80957705, 0x93cc7314, 0x211a1477,
0xe6ad2065, 0x77b5fa86, 0xc75442f5, 0xfb9d35cf,
0xebcdaf0c, 0x7b3e89a0, 0xd6411bd3, 0xae1e7e49,
0x00250e2d, 0x2071b35e, 0x226800bb, 0x57b8e0af,
0x2464369b, 0xf009b91e, 0x5563911d, 0x59dfa6aa,
0x78c14389, 0xd95a537f, 0x207d5ba2, 0x02e5b9c5,
0x83260376, 0x6295cfa9, 0x11c81968, 0x4e734a41,
0xb3472dca, 0x7b14a94a, 0x1b510052, 0x9a532915,
0xd60f573f, 0xbc9bc6e4, 0x2b60a476, 0x81e67400,
0x08ba6fb5, 0x571be91f, 0xf296ec6b, 0x2a0dd915,
0xb6636521, 0xe7b9f9b6, 0xff34052e, 0xc5855664,
0x53b02d5d, 0xa99f8fa1, 0x08ba4799, 0x6e85076a,
0x4b7a70e9, 0xb5b32944, 0xdb75092e, 0xc4192623,
0xad6ea6b0, 0x49a7df7d, 0x9cee60b8, 0x8fedb266,
0xecaa8c71, 0x699a17ff, 0x5664526c, 0xc2b19ee1,
0x193602a5, 0x75094c29, 0xa0591340, 0xe4183a3e,
0x3f54989a, 0x5b429d65, 0x6b8fe4d6, 0x99f73fd6,
0xa1d29c07, 0xefe830f5, 0x4d2d38e6, 0xf0255dc1,
0x4cdd2086, 0x8470eb26, 0x6382e9c6, 0x021ecc5e,
0x09686b3f, 0x3ebaefc9, 0x3c971814, 0x6b6a70a1,
0x687f3584, 0x52a0e286, 0xb79c5305, 0xaa500737,
0x3e07841c, 0x7fdeae5c, 0x8e7d44ec, 0x5716f2b8,
0xb03ada37, 0xf0500c0d, 0xf01c1f04, 0x0200b3ff,
0xae0cf51a, 0x3cb574b2, 0x25837a58, 0xdc0921bd,
0xd19113f9, 0x7ca92ff6, 0x94324773, 0x22f54701,
0x3ae5e581, 0x37c2dadc, 0xc8b57634, 0x9af3dda7,
0xa9446146, 0x0fd0030e, 0xecc8c73e, 0xa4751e41,
0xe238cd99, 0x3bea0e2f, 0x3280bba1, 0x183eb331,

0x4e548b38, 0x4f6db908, 0x6f420d03, 0xf60a04bf,
0x2cb81290, 0x24977c79, 0x5679b072, 0xbcaf89af,
0xde9a771f, 0xd9930810, 0xb38bae12, 0xdccf3f2e,
0x5512721f, 0x2e6b7124, 0x501adde6, 0x9f84cd87,
0x7a584718, 0x7408da17, 0xbc9f9abc, 0xe94b7d8c,
0xec7aec3a, 0xdb851dfa, 0x63094366, 0xc464c3d2,
0xef1c1847, 0x3215d908, 0xdd433b37, 0x24c2ba16,
0x12a14d43, 0x2a65c451, 0x50940002, 0x133ae4dd,
0x71dff89e, 0x10314e55, 0x81ac77d6, 0x5f11199b,
0x043556f1, 0xd7a3c76b, 0x3c11183b, 0x5924a509,
0xf28fe6ed, 0x97f1fbfa, 0x9ebabf2c, 0x1e153c6e,
0x86e34570, 0xeae96fb1, 0x860e5e0a, 0x5a3e2ab3,
0x771fe71c, 0x4e3d06fa, 0x2965dcb9, 0x99e71d0f,
0x803e89d6, 0x5266c825, 0x2e4cc, 0x9c10b36a,
0xc6150eba, 0x94e2ea78, 0xa5fc3c53, 0x1e0a2df4,
0xf2f74ea7, 0x361d2b3d, 0x1939260f, 0x19c27960,
0x5223a708, 0xf71312b6, 0xebadfe6e, 0xeac31f66,
0xe3bc4595, 0xa67bc883, 0xb17f37d1, 0x018cff28,
0xc332ddef, 0xbe6c5aa5, 0x65582185, 0x68ab9802,
0xeecea50f, 0xdb2f953b, 0x2aef7dad, 0x5b6e2f84,
0x1521b628, 0x29076170, 0xecdd4775, 0x619f1510,
0x13cca830, 0xeb61bd96, 0x0334fe1e, 0xaa0363cf,
0xb5735c90, 0x4c70a239, 0xd59e9e0b, 0xcbaade14,
0xeecc86bc, 0x60622ca7, 0x9cab5cab, 0xb2f3846e,
0x648b1eaf, 0x19bdf0ca, 0xa02369b9, 0x655abb50,
0x40685a32, 0x3c2ab4b3, 0x319ee9d5, 0xc021b8f7,
0x9b540b19, 0x875fa099, 0x95f7997e, 0x623d7da8,
0xf837889a, 0x97e32d77, 0x11ed935f, 0x16681281,
0x0e358829, 0xc7e61fd6, 0x96dedfa1, 0x7858ba99,
0x57f584a5, 0x1b227263, 0x9b83c3ff, 0x1ac24696,
0xcdb30aeb, 0x532e3054, 0x8fd948e4, 0x6dbc3128,
0x58ebf2ef, 0x34c6ffea, 0xfe28ed61, 0xee7c3c73,
0x5d4a14d9, 0xe864b7e3, 0x42105d14, 0x203e13e0,
0x45eee2b6, 0xa3aaabea, 0xdb6c4f15, 0xfacb4fd0,
0xc742f442, 0xef6abbb5, 0x654f3b1d, 0x41cd2105,
0xd81e799e, 0x86854dc7, 0xe44b476a, 0x3d816250,
0xcf62a1f2, 0x5b8d2646, 0xfc8883a0, 0xc1c7b6a3,
0x7f1524c3, 0x69cb7492, 0x47848a0b, 0x5692b285,
0x095bbf00, 0xad19489d, 0x1462b174, 0x23820e00,
0x58428d2a, 0x0c55f5ea, 0x1dadf43e, 0x233f7061,
0x3372f092, 0x8d937e41, 0xd65fecf1, 0x6c223bdb,
0x7cde3759, 0xcbee7460, 0x4085f2a7, 0xce77326e,
0xa6078084, 0x19f8509e, 0xe8efd855, 0x61d99735,
0xa969a7aa, 0xc50c06c2, 0x5a04abfc, 0x800bcadc,
0x9e447a2e, 0xc3453484, 0xfdd56705, 0x0e1e9ec9,
0xdb73dbd3, 0x105588cd, 0x675fda79, 0xe3674340,
0xc5c43465, 0x713e38d8, 0x3d28f89e, 0xf16dff20,

0x153e21e7, 0x8fb03d4a, 0xe6e39f2b, 0xdb83adf7,
0xe93d5a68, 0x948140f7, 0xf64c261c, 0x94692934,
0x411520f7, 0x7602d4f7, 0xbcf46b2e, 0xd4a20068,
0xd4082471, 0x3320f46a, 0x43b7d4b7, 0x500061af,
0x1e39f62e, 0x97244546, 0x14214f74, 0xbf8b8840,
0x4d95fc1d, 0x96b591af, 0x70f4ddd3, 0x66a02f45,
0xbfbc09ec, 0x03bd9785, 0x7fac6dd0, 0x31cb8504,
0x96eb27b3, 0x55fd3941, 0xda2547e6, 0xabca0a9a,
0x28507825, 0x530429f4, 0x0a2c86da, 0xe9b66dfb,
0x68dc1462, 0xd7486900, 0x680ec0a4, 0x27a18dee,
0x4f3ffea2, 0xe887ad8c, 0xb58ce006, 0x7af4d6b6,
0xaace1e7c, 0xd3375fec, 0xce78a399, 0x406b2a42,
0x20fe9e35, 0xd9f385b9, 0xee39d7ab, 0x3b124e8b,
0x1dc9faf7, 0x4b6d1856, 0x26a36631, 0xeae397b2,
0x3a6efa74, 0xdd5b4332, 0x6841e7f7, 0xca7820fb,
0xfb0af54e, 0xd8feb397, 0x454056ac, 0xba489527,
0x55533a3a, 0x20838d87, 0xfe6ba9b7, 0xd096954b,
0x55a867bc, 0xa1159a58, 0xcca92963, 0x99e1db33,
0xa62a4a56, 0x3f3125f9, 0x5ef47e1c, 0x9029317c,
0xfdf8e802, 0x04272f70, 0x80bb155c, 0x05282ce3,
0x95c11548, 0xe4c66d22, 0x48c1133f, 0xc70f86dc,
0x07f9c9ee, 0x41041f0f, 0x404779a4, 0x5d886e17,
0x325f51eb, 0xd59bc0d1, 0xf2bcc18f, 0x41113564,
0x257b7834, 0x602a9c60, 0xdff8e8a3, 0x1f636c1b,
0x0e12b4c2, 0x02e1329e, 0xaf664fd1, 0xcad18115,
0x6b2395e0, 0x333e92e1, 0x3b240b62, 0xeebeb922,
0x85b2a20e, 0xe6ba0d99, 0xde720c8c, 0x2da2f728,
0xd0127845, 0x95b794fd, 0x647d0862, 0xe7ccf5f0,
0x5449a36f, 0x877d48fa, 0xc39dfd27, 0xf33e8d1e,
0x0a476341, 0x992eff74, 0x3a6f6eab, 0xf4f8fd37,
0xa812dc60, 0xa1ebddf8, 0x991be14c, 0xdb6e6b0d,
0xc67b5510, 0x6d672c37, 0x2765d43b, 0xdcd0e804,
0xf1290dc7, 0xcc00ffa3, 0xb5390f92, 0x690fed0b,
0x667b9ffb, 0xcedb7d9c, 0xa091cf0b, 0xd9155ea3,
0xbb132f88, 0x515bad24, 0x7b9479bf, 0x763bd6eb,
0x37392eb3, 0xcc115979, 0x8026e297, 0xf42e312d,
0x6842ada7, 0xc66a2b3b, 0x12754ccc, 0x782ef11c,
0x6a124237, 0xb79251e7, 0x06a1bbe6, 0x4bfb6350,
0x1a6b1018, 0x11caedfa, 0x3d25bdd8, 0xe2e1c3c9,
0x44421659, 0x0a121386, 0xd90cec6e, 0xd5abea2a,
0x64af674e, 0xda86a85f, 0xbebfe988, 0x64e4c3fe,
0x9dbc8057, 0xf0f7c086, 0x60787bf8, 0x6003604d,
0xd1fd8346, 0xf6381fb0, 0x7745ae04, 0xd736fccc,
0x83426b33, 0xf01eab71, 0xb0804187, 0x3c005e5f,
0x77a057be, 0xbde8ae24, 0x55464299, 0xbf582e61,
0x4e58f48f, 0xf2ddfda2, 0xf474ef38, 0x8789bdc2,
0x5366f9c3, 0xc8b38e74, 0xb475f255, 0x46fcd9b9,

0x7aeb2661, 0x8b1ddf84, 0x846a0e79, 0x915f95e2,
0x466e598e, 0x20b45770, 0x8cd55591, 0xc902de4c,
0xb90bace1, 0xbb8205d0, 0x11a86248, 0x7574a99e,
0xb77f19b6, 0xe0a9dc09, 0x662d09a1, 0xc4324633,
0xe85a1f02, 0x09f0be8c, 0x4a99a025, 0x1d6efe10,
0x1ab93d1d, 0x0ba5a4df, 0xa186f20f, 0x2868f169,
0xdcb7da83, 0x573906fe, 0xa1e2ce9b, 0x4fcd7f52,
0x50115e01, 0xa70683fa, 0xa002b5c4, 0x0de6d027,
0x9af88c27, 0x773f8641, 0xc3604c06, 0x61a806b5,
0xf0177a28, 0xc0f586e0, 0x006058aa, 0x30dc7d62,
0x11e69ed7, 0x2338ea63, 0x53c2dd94, 0xc2c21634,
0xbbcbee56, 0x90bcb6de, 0xebfc7da1, 0xce591d76,
0x6f05e409, 0x4b7c0188, 0x39720a3d, 0x7c927c24,
0x86e3725f, 0x724d9db9, 0x1ac15bb4, 0xd39eb8fc,
0xed545578, 0x08fca5b5, 0xd83d7cd3, 0x4dad0fc4,
0x1e50ef5e, 0xb161e6f8, 0xa28514d9, 0x6c51133c,
0x6fd5c7e7, 0x56e14ec4, 0x362abfce, 0xddc6c837,
0xd79a3234, 0x92638212, 0x670efa8e, 0x406000e0,
0x3a39ce37, 0xd3faf5cf, 0xabc27737, 0x5ac52d1b,
0x5cb0679e, 0x4fa33742, 0xd3822740, 0x99bc9bbe,
0xd5118e9d, 0xbf0f7315, 0xd62d1c7e, 0xc700c47b,
0xb78c1b6b, 0x21a19045, 0xb26eb1be, 0x6a366eb4,
0x5748ab2f, 0xbc946e79, 0xc6a376d2, 0x6549c2c8,
0x530ff8ee, 0x468dde7d, 0xd5730a1d, 0x4cd04dc6,
0x2939bbdb, 0xa9ba4650, 0xac9526e8, 0xbe5ee304,
0xa1fad5f0, 0x6a2d519a, 0x63ef8ce2, 0x9a86ee22,
0xc089c2b8, 0x43242ef6, 0xa51e03aa, 0x9cf2d0a4,
0x83c061ba, 0x9be96a4d, 0x8fe51550, 0xba645bd6,
0x2826a2f9, 0xa73a3ae1, 0x4ba99586, 0xef5562e9,
0xc72fefd3, 0xf752f7da, 0x3f046f69, 0x77fa0a59,
0x80e4a915, 0x87b08601, 0x9b09e6ad, 0x3b3ee593,
0xe990fd5a, 0x9e34d797, 0x2cf0b7d9, 0x022b8b51,
0x96d5ac3a, 0x017da67d, 0xd1cf3ed6, 0x7c7d2d28,
0x1f9f25cf, 0xadf2b89b, 0x5ad6b472, 0x5a88f54c,
0xe029ac71, 0xe019a5e6, 0x47b0acfd, 0xed93fa9b,
0xe8d3c48d, 0x283b57cc, 0xf8d56629, 0x79132e28,
0x785f0191, 0xed756055, 0xf7960e44, 0xe3d35e8c,
0x15056dd4, 0x88f46dba, 0x03a16125, 0x0564f0bd,
0xc3eb9e15, 0x3c9057a2, 0x97271aec, 0xa93a072a,
0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb, 0x26dcf319,
0x7533d928, 0xb155fdf5, 0x03563482, 0x8aba3cbb,
0x28517711, 0xc20ad9f8, 0xabcc5167, 0xccad925f,
0x4de81751, 0x3830dc8e, 0x379d5862, 0x9320f991,
0xea7a90c2, 0xfb3e7bce, 0x5121ce64, 0x774fbe32,
0xa8b6e37e, 0xc3293d46, 0x48de5369, 0x6413e680,
0xa2ae0810, 0xdd6db224, 0x69852dfd, 0x09072166,
0xb39a460a, 0x6445c0dd, 0x586cdecf, 0x1c20c8ae,

```
  0x5bbef7dd, 0x1b588d40, 0xccd2017f, 0x6bb4e3bb,
  0xdda26a7e, 0x3a59ff45, 0x3e350a44, 0xbcb4cdd5,
  0x72eacea8, 0xfa6484bb, 0x8d6612ae, 0xbf3c6f47,
  0xd29be463, 0x542f5d9e, 0xaec2771b, 0xf64e6370,
  0x740e0d8d, 0xe75b1357, 0xf8721671, 0xaf537d5d,
  0x4040cb08, 0x4eb4e2cc, 0x34d2466a, 0x0115af84,
  0xe1b00428, 0x95983a1d, 0x06b89fb4, 0xce6ea048,
  0x6f3f3b82, 0x3520ab82, 0x011a1d4b, 0x277227f8,
  0x611560b1, 0xe7933fdc, 0xbb3a792b, 0x344525bd,
  0xa08839e1, 0x51ce794b, 0x2f32c9b7, 0xa01fbac9,
  0xe01cc87e, 0xbcc7d1f6, 0xcf0111c3, 0xa1e8aac7,
  0x1a908749, 0xd44fbd9a, 0xd0dadecb, 0xd50ada38,
  0x0339c32a, 0xc6913667, 0x8df9317c, 0xe0b12b4f,
  0xf79e59b7, 0x43f5bb3a, 0xf2d519ff, 0x27d9459c,
  0xbf97222c, 0x15e6fc2a, 0x0f91fc71, 0x9b941525,
  0xfae59361, 0xceb69ceb, 0xc2a86459, 0x12baa8d1,
  0xb6c1075e, 0xe3056a0c, 0x10d25065, 0xcb03a442,
  0xe0ec6e0e, 0x1698db3b, 0x4c98a0be, 0x3278e964,
  0x9f1f9532, 0xe0d392df, 0xd3a0342b, 0x8971f21e,
  0x1b0a7441, 0x4ba3348c, 0xc5be7120, 0xc37632d8,
  0xdf359f8d, 0x9b992f2e, 0xe60b6f47, 0x0fe3f11d,
  0xe54cda54, 0x1edad891, 0xce6279cf, 0xcd3e7e6f,
  0x1618b166, 0xfd2c1d05, 0x848fd2c5, 0xf6fb2299,
  0xf523f357, 0xa6327623, 0x93a83531, 0x56cccd02,
  0xacf08162, 0x5a75ebb5, 0x6e163697, 0x88d273cc,
  0xde966292, 0x81b949d0, 0x4c50901b, 0x71c65614,
  0xe6c6c7bd, 0x327a140a, 0x45e1d006, 0xc3f27b9a,
  0xc9aa53fd, 0x62a80f00, 0xbb25bfe2, 0x35bdd2f6,
  0x71126905, 0xb2040222, 0xb6cbcf7c, 0xcd769c2b,
  0x53113ec0, 0x1640e3d3, 0x38abbd60, 0x2547adf0,
  0xba38209c, 0xf746ce76, 0x77afa1c5, 0x20756060,
  0x85cbfe4e, 0x8ae88dd8, 0x7aaaf9b0, 0x4cf9aa7e,
  0x1948c25c, 0x02fb8a8c, 0x01c36ae4, 0xd6ebe1f9,
  0x90d4f869, 0xa65cdea0, 0x3f09252d, 0xc208e69f,
  0xb74e6132, 0xce77e25b, 0x578fdfe3, 0x3ac372e6,
};
```

Apêndice C: Tabela de Freqüência para as Diferenças de Saída dos S-Boxes

As tabelas apresentadas são para o Blowfish Reduzido.

Tabela C.1 – Tabela de Freqüência para o S-Box 1.

Saída (S-Box - 1)	Freqüência Ocorrência		Saída (S-Box - 1)	Freqüência Ocorrência
$7f_x$	3298		fc_x	2998
83_x	3266		ee_x	2996
66_x	3264		ca_x	2994
$2a_x$	3262		$e9_x$	2992
7_x	3256		$8a_x$	2992
$4c_x$	3242		$9c_x$	2990
23_x	3242		$d4_x$	2988
80_x	3238		$b8_x$	2988
47_x	3234		ff_x	2984
$b7_x$	3210		$8b_x$	2984
cf_x	3206		45_x	2982
$b9_x$	3206		67_x	2980
$9d_x$	3204		$a4_x$	2978
28_x	3204		$b0_x$	2978
82_x	3202		38_x	2978
$3e_x$	3202		fd_x	2978
46_x	3202		40_x	2976
84_x	3200		$e1_x$	2970
$6b_x$	3196		$c1_x$	2962
$5b_x$	3194		ba_x	2958
ae_x	3190		17_x	2956
c_x	3188		54_x	2954
64_x	3188		$d3_x$	2950
cd_x	3188		$b6_x$	2950
26_x	3182		$6a_x$	2948
$7e_x$	3180		$8f_x$	2946
79_x	3180		$f0_x$	2946
$e4_x$	3180		$4b_x$	2946
$f5_x$	3176		74_x	2946
53_x	3176		$1b_x$	2944
$c4_x$	3172		$2b_x$	2940
fa_x	3170		$a1_x$	2932
87_x	3168		73_x	2930
$d2_x$	3164		$e6_x$	2928
ef_x	3162		$a8_x$	2926
$d8_x$	3156		32_x	2922
$f2_x$	3154		9_x	2918
cc_x	3152		15_x	2912
$3a_x$	3152		$8c_x$	2896

Saída (S-Box - 1)	Freqüência Ocorrência		Saída (S-Box - 1)	Freqüência Ocorrência
aa_x	3152		$2e_x$	2888
$d6_x$	3148		16_x	2850
$2c_x$	3148		$a7_x$	3016
$a3_x$	3146		$e2_x$	3016
27_x	3146		$1d_x$	3014
94_x	3144		$f6_x$	3014
19_x	3144		$6f_x$	3012
52_x	3144		ac_x	3012
dc_x	3144		d_x	3010
$a5_x$	3140		$4e_x$	3010
$c6_x$	3134		ce_x	3008
2_x	3132		34_x	3008
$e3_x$	3130		$2f_x$	3006
a_x	3130		93_x	3006
$6e_x$	3130		63_x	3006
$1c_x$	3128		95_x	3006
$b5_x$	3126		dd_x	3004
22_x	3124		11_x	3004
$c7_x$	3124		49_x	3000
ec_x	3122		30_x	3000
$f7_x$	3122		$b3_x$	2998
61_x	3122		$1a_x$	3030
98_x	3120		59_x	3030
$d0_x$	3118		bb_x	3028
ad_x	3118		85_x	3028
$a9_x$	3116		48_x	3028
$4a_x$	3114		18_x	3028
12_x	3114		4_x	3026
$5f_x$	3114		76_x	3026
$3d_x$	3112		$9a_x$	3026
68_x	3110		db_x	3024
77_x	3110		1_x	3024
62_x	3110		$c3_x$	3020
$d5_x$	3110		3_x	3020
$f4_x$	3110		5_x	3020
$3c_x$	3110		$4f_x$	3020
fe_x	3108		$8e_x$	3020
$4d_x$	3108		$e5_x$	3020
bf_x	3108		35_x	3042
$7b_x$	3106		$e0_x$	3042
89_x	3106		96_x	3042
$6d_x$	3106		$a0_x$	3040
$5a_x$	3104		$e7_x$	3038
f_x	3104		91_x	3038
$c2_x$	3104		97_x	3036
0x21	3100		0x36	3036
0xd7	3100		0x39	3036
0x6c	3098		0xea	3036
0x3b	3098		0x69	3036
0x56	3098		0xc8	3036
0xed	3096		0xd1	3036

Saída (S-Box - 1)	Freqüência Ocorrência		Saída (S-Box - 1)	Freqüência Ocorrência
0x20	3096		0xc0	3034
0xdf	3096		0xeb	3034
0x43	3092		0x71	3034
0x10	3092		0xe	3032
0x1f	3092		0x42	3032
0x6	3092		0x9f	3032
0x44	3092		0x72	3062
0x13	3092		0xc5	3062
0x31	3092		0x3f	3060
0xf1	3090		0x9b	3060
0x24	3090		0x86	3058
0xaf	3090		0x57	3056
0xf8	3090		0xde	3054
0xab	3090		0x8	3054
0x55	3090		0x7d	3054
0x41	3088		0x2d	3050
0x33	3086		0x7a	3050
0x5c	3086		0x5e	3050
0xfb	3084		0x88	3046
0x99	3084		0x51	3046
0x29	3084		0x7c	3044
0x81	3082		0x8d	3044
0x65	3082		0xb2	3044
0x5d	3082		0xbe	3042
0xbc	3082		0xd9	3070
0x58	3080		0x75	3070
0xb4	3080		0xf9	3068
0x50	3078		0xc9	3068
0x90	3076		0xb1	3066
0x60	3076		0x1e	3066
0xbd	3076		0x9e	3066
0xcb	3076		0x78	3066
0x92	3074		0xda	3064
0xa2	3072		0xb	3064
0x37	3072		0x14	3064
0xf3	3072		0x70	3064
0xe8	3070		0xa6	3064
			0x25	3062

Tabela C.2 – Tabela de Freqüência para o S-Box 2.

Saída (S-Box – 2)	Freqüência Ocorrência		Saída (S-Box – 2)	Freqüência Ocorrência
27_x	3264		$3f_x$	2974
$F8_x$	3234		$a1_x$	2972
$C2_x$	3230		30_x	2972
$6a_x$	3224		96_x	2970
$1a_x$	3222		$d3_x$	2970
$D6_x$	3210		$4d_x$	2968
dc_x	3206		$2d_x$	2968
77_x	3206		$a2_x$	2966

Saída (S-Box – 2)	Freqüência Ocorrência		Saída (S-Box – 2)	Freqüência Ocorrência
$c7_x$	3204		$e1_x$	2966
5_x	3202		e_x	2964
$2f_x$	3200		$4e_x$	2964
fc_x	3194		cf_x	2962
58_x	3194		$b1_x$	2954
42_x	3192		$f3_x$	2950
af_x	3190		$b2_x$	2950
57_x	3188		$e4_x$	2948
92_x	3186		$f4_x$	2946
$2e_x$	3186		93_x	2944
bf_x	3186		$a4_x$	2942
f_x	3182		$1c_x$	2942
14_x	3182		$a6_x$	2940
fa_x	3178		$c4_x$	2938
33_x	3178		59_x	2928
$e9_x$	3178		$a0_x$	2916
$8b_x$	3176		69_x	2906
$f1_x$	3172		73_x	2880
$b9_x$	3172		$e7_x$	2850
de_x	3168		c_x	3016
66_x	3166		35_x	3014
ff_x	3166		95_x	3014
$1b_x$	3164		$b7_x$	3014
21_x	3164		$6b_x$	3012
$d9_x$	3162		$3b_x$	3010
52_x	3162		$2c_x$	3010
a_x	3162		16_x	3010
$8e_x$	3160		32_x	3008
$a3_x$	3156		$3e_x$	3006
55_x	3154		ef_x	3006
26_x	3154		ac_x	3006
79_x	3152		45_x	3004
85_x	3152		$4b_x$	3002
$5e_x$	3152		$9f_x$	3000
ca_x	3150		81_x	3000
68_x	3150		$c5_x$	3000
64_x	3150		$f0_x$	3000
$8c_x$	3148		61_x	3000
24_x	3146		$a7_x$	3000
38_x	3144		36_x	2998
$5c_x$	3144		$8a_x$	2998
41_x	3142		$e6_x$	2996
39_x	3142		da_x	2996
37_x	3140		67_x	2994
$3a_x$	3138		$6d_x$	2994
$8d_x$	3136		$6e_x$	2992
$c0_x$	3136		$2a_x$	2990
$6c_x$	3134		62_x	2988
$5a_x$	3132		$b6_x$	2984
$9d_x$	3132		db_x	2982
7_x	3132		$e3_x$	2980

Saída (S-Box – 2)	Freqüência Ocorrência		Saída (S-Box – 2)	Freqüência Ocorrência
6_x	3130		$f9_x$	2980
$3d_x$	3130		$e0_x$	2980
4_x	3128		20_x	2978
$6f_x$	3126		56_x	2976
bd_x	3126		74_x	3062
78_x	3126		$a9_x$	3062
51_x	3124		ee_x	3062
$e5_x$	3124		97_x	3060
75_x	3124		$7d_x$	3058
$1d_x$	3124		$f6_x$	3056
25_x	3124		ba_x	3052
15_x	3124		99_x	3052
9_x	3122		$9b_x$	3050
$c9_x$	3122		50_x	3050
$b8_x$	3116		$9c_x$	3046
12_x	3116		$b3_x$	3042
$9e_x$	3114		22_x	3042
49_x	3114		cd_x	3040
$9a_x$	3112		71_x	3040
$e2_x$	3112		ea_x	3038
bb_x	3110		fb_x	3038
63_x	3110		cb_x	3038
ae_x	3108		$c6_x$	3038
90_x	3108		$a8_x$	3038
53_x	3108		ce_x	3038
d_x	3108		91_x	3036
$e8_x$	3108		$f5_x$	3036
aa_x	3106		$d8_x$	3036
b_x	3104		$1f_x$	3034
$4f_x$	3104		82_x	3034
46_x	3100		98_x	3034
17_x	3100		ad_x	3034
$f2_x$	3100		88_x	3034
$7a_x$	3100		$5f_x$	3032
54_x	3100		$a5_x$	3032
$4c_x$	3098		$d4_x$	3028
$f7_x$	3098		94_x	3028
fd_x	3098		43_x	3026
$5d_x$	3098		$8f_x$	3024
$1e_x$	3094		eb_x	3024
$7c_x$	3094		3_x	3024
80_x	3092		40_x	3024
df_x	3090		84_x	3024
65_x	3090		$c1_x$	3022
$d7_x$	3088		34_x	3020
47_x	3088		$2b_x$	3020
89_x	3088		fe_x	3018
$b4_x$	3088		$c3_x$	3018
86_x	3088		87_x	3018
2_x	3088		70_x	3016
11_x	3088		18_x	3016

Saída (S-Box – 2)	Freqüência Ocorrência		Saída (S-Box – 2)	Freqüência Ocorrência
13_x	3086		ec_x	3066
$d0_x$	3084		ed_x	3066
$b0_x$	3084		$c8_x$	3066
31_x	3082		83_x	3066
dd_x	3080		60_x	3064
$d5_x$	3080		$7f_x$	3064
$4a_x$	3080		ab_x	3062
cc_x	3078		1_x	3062
$7e_x$	3078		19_x	3062
76_x	3078		bc_x	3070
8_x	3078		29_x	3070
$7b_x$	3076		$d1_x$	3070
72_x	3076		$5b_x$	3070
48_x	3074		be_x	3070
23_x	3074		$d2_x$	3068
28_x	3072		10_x	3068
$b5_x$	3072		$3c_x$	3068
44_x	3070			

Tabela C.3 – Tabela de Freqüência para o S-Box 3.

Saída (S-Box – 3)	Freqüência Ocorrência		Saída (S-Box – 3)	Freqüência Ocorrência
0x57	3286		0xc7	3054
0x8b	3270		0x9a	3054
0xe	3258		0x78	3052
0x68	3246		0x9	3052
0xd	3244		0xc6	3052
0xa8	3238		0x11	3050
0xc0	3230		0x8	3050
0x22	3228		0x39	3050
0xe0	3220		0xe1	3050
0x96	3218		0xdd	3048
0xe5	3216		0x17	3046
0x54	3216		0x55	3044
0x20	3212		0x1b	3044
0x34	3208		0xa0	3042
0xbb	3202		0xaa	3040
0x76	3200		0xbf	3040
0x65	3196		0xd7	3040
0x3e	3192		0x2	3038
0x92	3190		0x85	3036
0xa4	3182		0xc4	3036
0x48	3182		0xbe	3034
0xd1	3178		0xc	3034
0x93	3178		0x70	3034
0xf4	3178		0xe3	3032
0x7	3174		0xa7	3032
0x6e	3174		0x49	3032
0xb7	3172		0xc8	3032

Saída (S-Box – 3)	Freqüência Ocorrência		Saída (S-Box – 3)	Freqüência Ocorrência
0xfc	3172		0x66	3030
0x5b	3172		0x58	3030
0xda	3170		0x40	3028
0x4c	3168		0xf	3026
0xb6	3168		0x6d	3026
0x63	3168		0xf8	3026
0xca	3168		0xf7	3024
0x60	3166		0xd3	3024
0xcb	3164		0x23	3024
0xb5	3164		0x2d	3024
0xe4	3164		0xb1	3024
0x29	3158		0xa2	3022
0xeb	3158		0x6f	3020
0x59	3156		0x4a	3020
0x56	3156		0xac	3018
0x73	3154		0xf1	3018
0xf3	3150		0x24	3018
0xfb	3148		0x41	3018
0x1a	3148		0xaf	3016
0xd4	3146		0x2f	3016
0xf5	3146		0x8a	3016
0x5a	3144		0x6c	3014
0x81	3144		0x94	3014
0x33	3142		0xef	3012
0x8c	3142		0x7c	3012
0xfa	3142		0x90	3012
0xee	3140		0x21	3010
0xce	3140		0x1f	3010
0xa5	3140		0xe2	3008
0x3d	3140		0xc9	3008
0xdb	3140		0x30	3006
0xd9	3140		0xa6	3006
0x89	3134		0x3b	3006
0x12	3134		0x9e	3004
0x77	3128		0x1e	3004
0x8f	3126		0x4f	3004
0xb	3122		0x3	3002
0xa	3122		0x1	3002
0xba	3122		0x5f	3002
0x18	3122		0xc5	3002
0xfd	3122		0x19	3000
0x69	3122		0x67	3000
0x1d	3122		0x7d	2998
0xc1	3122		0x31	2994
0xdf	3122		0xe8	2994
0xae	3120		0xd5	2994
0x10	3120		0xb9	2994
0xad	3120		0x7f	2992
0x6b	3118		0xde	2990
0x4b	3116		0x4e	2990
0xdc	3116		0xe6	2988

Saída (S-Box – 3)	Freqüência Ocorrência		Saída (S-Box – 3)	Freqüência Ocorrência
0x16	3114		0x27	2984
0x42	3114		0x50	2982
0x5e	3112		0x43	2980
0x4	3112		0xa1	2980
0xff	3112		0x91	2978
0x72	3110		0x2b	2978
0xf2	3104		0x32	2976
0x13	3102		0x5c	2974
0xe7	3100		0x7e	2970
0x44	3100		0x3c	2962
0x8e	3100		0xd8	2956
0x5	3100		0xa3	2956
0x15	3100		0x62	2952
0xa9	3100		0x2c	2950
0xed	3098		0xb0	2948
0x7a	3094		0x14	2948
0x38	3090		0x37	2944
0x87	3088		0xf6	2944
0xf0	3088		0xab	2942
0xc2	3088		0x71	2940
0x79	3088		0x5d	2940
0xc3	3086		0x51	2938
0x7b	3086		0x8d	2932
0xea	3086		0x1c	2928
0xb3	3086		0x47	2926
0x26	3084		0x3f	2918
0x86	3084		0xe9	2910
0x35	3084		0xb4	2908
0xbc	3082		0x9b	2902
0x6	3082		0xcf	2894
0x45	3080		0xfe	2870
0x9c	3080			
0x95	3080			
0x2a	3080			
0xcd	3080			
0x28	3078			
0x9d	3078			
0x3a	3078			
0x46	3076			
0xd6	3076			
0x9f	3076			
0x36	3074			
0xd2	3074			
0xd0	3074			
0x99	3074			
0x2e	3070			
0xb8	3070			
0xbd	3070			
0x98	3068			
0x53	3068			
0x88	3068			

Saída (S-Box – 3)	Freqüência Ocorrência		Saída (S-Box – 3)	Freqüência Ocorrência
0x61	3068			
0x6a	3068			
0xcc	3066			
0x25	3066			
0x83	3066			
0x84	3066			
0x4d	3064			
0xb2	3064			
0xec	3064			
0x64	3062			
0x97	3062			
0x74	3060			
0x80	3058			
0x52	3056			
0x75	3056			
0xf9	3056			
0x82	3054			

Tabela C.4 – Tabela de Freqüência para o S-Box 4.

Saída (S-Box – 4)	Freqüência Ocorrência		Saída (S-Box – 4)	Freqüência Ocorrência
0xae	3328		0xe6	3074
0x9d	3262		0x2e	3074
0x8c	3246		0xf3	3074
0xf2	3242		0xd4	3074
0x62	3238		0xa4	3072
0x94	3232		0x47	3072
0xb6	3232		0x97	3072
0xe4	3228		0x14	3070
0x8	3218		0xbd	3070
0x41	3218		0x86	3070
0x3b	3216		0xc1	3068
0xd5	3212		0x69	3068
0x9e	3208		0x54	3068
0x35	3208		0x5d	3068
0x75	3202		0x4c	3066
0xcd	3200		0x60	3060
0x74	3196		0xcf	3060
0x5f	3196		0xa6	3058
0x96	3194		0x81	3056
0xca	3188		0xce	3056
0x5b	3182		0xcb	3056
0x4e	3182		0xa2	3056
0x3d	3178		0x13	3054
0x11	3178		0xd6	3050
0x9f	3176		0x85	3050
0xb3	3176		0x6d	3048
0x15	3174		0xe5	3048
0xc	3170		0xed	3048

Saída (S-Box – 4)	Freqüência Ocorrência		Saída (S-Box – 4)	Freqüência Ocorrência
0xbb	3170		0x3c	3046
0x3a	3170		0xb4	3046
0x3f	3166		0xd0	3046
0x6c	3164		0x23	3044
0xd3	3164		0x79	3042
0x30	3162		0x7a	3042
0x1	3162		0x37	3042
0x40	3160		0x20	3042
0xb2	3160		0xda	3040
0x64	3158		0x2c	3040
0xde	3158		0x9a	3040
0x7c	3156		0x1b	3038
0x6	3154		0x39	3036
0xe7	3154		0x1c	3036
0x57	3152		0x4	3036
0x9	3150		0x49	3036
0x5c	3150		0xb7	3036
0x67	3144		0x34	3034
0x51	3142		0xd	3034
0x21	3142		0xdc	3032
0x33	3142		0xb1	3032
0xc3	3140		0x6a	3032
0x93	3138		0x7d	3030
0x91	3138		0xc4	3030
0x4d	3138		0xe	3028
0x73	3136		0x70	3028
0xcc	3136		0x1e	3028
0x2d	3136		0xff	3026
0xf6	3134		0xdf	3026
0xf0	3134		0x7e	3026
0x16	3132		0xe8	3026
0xf9	3132		0xc9	3026
0xa3	3132		0x2b	3024
0xab	3130		0xa5	3022
0x8d	3130		0x66	3020
0xba	3128		0x58	3020
0xb5	3126		0x38	3020
0x8b	3126		0xa1	3020
0xf1	3126		0x68	3018
0x1d	3126		0x24	3018
0x8e	3126		0xe3	3018
0x99	3126		0x55	3016
0x71	3126		0xa	3016
0xbc	3124		0xf4	3014
0xc7	3124		0xd1	3012
0x90	3124		0x59	3012
0x95	3122		0x4b	3010
0x78	3120		0x52	3010
0x50	3120		0xad	3008
0x82	3120		0x22	3008
0x10	3120		0x5	3008

Saída (S-Box – 4)	Freqüência Ocorrência		Saída (S-Box – 4)	Freqüência Ocorrência
0x6e	3118		0xb8	3006
0x89	3116		0xeb	3006
0xf	3116		0x56	3004
0x72	3114		0xdd	3004
0x98	3114		0xfa	3000
0x46	3114		0xec	2998
0x9c	3112		0x1f	2996
0x76	3110		0x31	2996
0x6f	3110		0xe9	2996
0x87	3110		0xd7	2994
0x43	3110		0x7b	2994
0x83	3108		0x65	2990
0x1a	3108		0xc5	2988
0xc2	3106		0xc6	2988
0xa8	3106		0x48	2988
0xf5	3106		0x32	2988
0x77	3104		0x18	2986
0x5a	3102		0x28	2986
0x3e	3102		0x8f	2982
0x8a	3102		0x36	2976
0xc0	3102		0xfb	2976
0xfd	3100		0xe2	2976
0xf7	3096		0x2a	2974
0xef	3092		0xfc	2972
0x53	3092		0xb	2966
0x2f	3092		0xa7	2964
0x88	3090		0x84	2964
0x9b	3090		0x2	2964
0xf8	3090		0xe0	2962
0xa0	3088		0x27	2960
0xb9	3088		0x17	2956
0x44	3086		0x6b	2954
0x5e	3086		0x4f	2952
0xb0	3086		0xc8	2946
0xa9	3086		0x19	2940
0x92	3086		0xac	2936
0xfe	3084		0xd8	2936
0x3	3082		0xaa	2932
0x7	3082		0x26	2930
0xea	3080		0x29	2930
0xbf	3080		0x25	2928
0x80	3078		0xaf	2924
0x4a	3078		0xe1	2924
0x63	3076		0x61	2918
0xbe	3076		0xee	2916
			0x7f	2916
			0x45	2916
			0xd2	2908
			0x12	2908
			0x42	2886
			0xd9	2816

Saída (S-Box – 4)	Freqüência Ocorrência		Saída (S-Box – 4)	Freqüência Ocorrência
			0xdb	2802

Apêndice D: Tabela de Valores dos S-Boxes do Blowfish

As tabelas apresentadas mostram os valores dos S-Boxes para o Blowfish.

Tabela D.1 – Valores para o S-Box 1.

Entrada	Saída	Entrada	Saída
0x0	0xdc36a449	0x80	0x9fde597
0x1	0x7dc33a4a	0x81	0x57ff8f99
0x2	0x136a2219	0x82	0xb978287b
0x3	0x22522132	0x83	0x19937c80
0x4	0xf178fe32	0x84	0x657d2858
0x5	0x4f865b2c	0x85	0x5ccb3c4d
0x6	0xac122f10	0x86	0x1ebe58a4
0x7	0x3ea1294f	0x87	0xbceb97c1
0x8	0x81246c1	0x88	0x2595a04b
0x9	0xfe679e2a	0x89	0xbaf3ba4
0xa	0x8a0ed4e	0x8a	0x51a3d290
0xb	0x8e1d4bec	0x8b	0x58055d14
0xc	0xc6de4ae3	0x8c	0xef91542f
0xd	0xc634698e	0x8d	0xc88d8ac4
0xe	0x9d30eb15	0x8e	0x5052b91e
0xf	0x12a8a5bc	0x8f	0x85518585
0x10	0x8eb8657	0x90	0x756295c5
0x11	0x816fc923	0x91	0x335ff7b2
0x12	0xc077ffe7	0x92	0xaa4cd972
0x13	0x2e71961c	0x93	0xf8b039d9
0x14	0x1bc7a050	0x94	0xe133047
0x15	0xe0eeb64b	0x95	0xbac97d0e
0x16	0x909ff7f7	0x96	0x3b56e4dc
0x17	0x2f01ecc7	0x97	0x503d5e04
0x18	0xbcfef2bb	0x98	0x261de224
0x19	0x212b111	0x99	0x7586288d
0x1a	0xcff04706	0x9a	0xf2e0aff1
0x1b	0x2d24bb19	0x9b	0xc9d0cb5f
0x1c	0xc5c4c84	0x9c	0x96584bdb
0x1d	0xbe468b45	0x9d	0x7d7252da
0x1e	0x17b73f35	0x9e	0xf821f17a
0x1f	0x9ced6004	0x9f	0xc16de254
0x20	0x8f1adde8	0xa0	0xe4350c46
0x21	0xdcca1e1b	0xa1	0x81e11dcb
0x22	0x299b5004	0xa2	0xfff6fea
0x23	0x7a9a8334	0xa3	0x44d06b8b
0x24	0x3399fd8a	0xa4	0xd4e938e6
0x25	0x1b3ab3ce	0xa5	0xfe4b4dba

0x26	0xe84e39d6	0xa6	0xa989434b
0x27	0xcc52de4c	0xa7	0xe35063dc
0x28	0x7a33a1b4	0xa8	0xec78435d
0x29	0x4fd2dedd	0xa9	0x275926b0
0x2a	0xa5fbc8b4	0xaa	0x860dfadd
0x2b	0x72bfe76b	0xab	0x4cc41ab9
0x2c	0xa978b4ea	0xac	0x452d29d9
0x2d	0x2ec576c1	0xad	0xc368cc8e
0x2e	0x60929d58	0xae	0x23d31193
0x2f	0xd95a4961	0xaf	0x7ff6eac8
0x30	0xdd9dbcd5	0xb0	0x2bc5ddfb
0x31	0x59f42b9b	0xb1	0x8e16b870
0x32	0xecbc3372	0xb2	0xc56fa74f
0x33	0x2da3692b	0xb3	0xe4ae10a5
0x34	0x58b4a025	0xb4	0x60f5239d
0x35	0x453b7974	0xb5	0x3edecb0d
0x36	0x8483f929	0xb6	0xe63c9ac4
0x37	0xa05668b7	0xb7	0xcba27a6b
0x38	0x42eccfc1	0xb8	0x508aaf0a
0x39	0x9f6cb58d	0xb9	0x139d3df2
0x3a	0xb8bba15	0xba	0x2a67d39b
0x3b	0xb60f6ae7	0xbb	0xc55f949f
0x3c	0xeec409c2	0xbc	0x163ed892
0x3d	0x66f1ecc6	0xbd	0x83e179be
0x3e	0x6bf52cff	0xbe	0x5b651bf
0x3f	0x24f06dc6	0xbf	0xbf5b91c2
0x40	0xfb463a9	0xc0	0x4240a9cb
0x41	0xe6156f80	0xc1	0x870e9767
0x42	0x79bb211f	0xc2	0x85974253
0x43	0x835ca228	0xc3	0x47ec0e0e
0x44	0x96711fe5	0xc4	0xb5eac0ef
0x45	0x48f30f8e	0xc5	0x819a9759
0x46	0x1b182508	0xc6	0x96131fe1
0x47	0x833078a8	0xc7	0x98b6abad
0x48	0x63206c0e	0xc8	0xbe088154
0x49	0x6acdcbaa	0xc9	0xb7dff7a0
0x4a	0xb943d4e5	0xca	0x39f4c96b
0x4b	0xad40aa8d	0xcb	0x7fc34345
0x4c	0xaddbdfd8	0xcc	0xf24779ab
0x4d	0x1203b12	0xcd	0x2da40f7
0x4e	0xd839c915	0xce	0x21168199
0x4f	0xed2f68d3	0xcf	0x5785a5d3
0x50	0xec3b6a9d	0xd0	0x282a00c1
0x51	0xd5d1e901	0xd1	0x4d497e37
0x52	0x23e24e70	0xd2	0x885bd142
0x53	0xb0e5305d	0xd3	0xfe83a784
0x54	0xc94bf9a8	0xd4	0x1ba828d9
0x55	0xc8f339c2	0xd5	0xe9c61784

0x56	0x2a8a1843	0xd6	0x1e4a2219
0x57	0x359bba32	0xd7	0xc0388bea
0x58	0xbeaef55d	0xd8	0x9976920a
0x59	0x1e20048b	0xd9	0x53680ea8
0x5a	0x2681e2c7	0xda	0xce741f96
0x5b	0x1cb7a8e9	0xdb	0xa4833440
0x5c	0x846dbd28	0xdc	0xbf4eb442
0x5d	0xaa2da203	0xdd	0x7c19f9e
0x5e	0x1717d2ca	0xde	0x616a9dfa
0x5f	0x6f39b2c	0xdf	0x1d656def
0x60	0x3d61a66c	0xe0	0xc9d1ac16
0x61	0xaa7d4ca7	0xe1	0xf964d0d2
0x62	0xcd17316c	0xe2	0x68654f98
0x63	0x822e059b	0xe3	0xe5baca32
0x64	0xf5961e5d	0xe4	0x280d1792
0x65	0x2a23708	0xe5	0x8ade7317
0x66	0x81898a46	0xe6	0xefca77e8
0x67	0xdd9aaf10	0xe7	0xf74a8473
0x68	0x80bc97c8	0xe8	0x2267a8dd
0x69	0xe3f44569	0xe9	0x91591855
0x6a	0xeebf77f8	0xea	0x55ebefb
0x6b	0xbd6b406d	0xeb	0x1e4b179a
0x6c	0x13fafc4b	0xec	0xeaac6969
0x6d	0xf194f64e	0xed	0x9221ea60
0x6e	0xdf2af150	0xee	0x4f159a17
0x6f	0xaa7f5cf2	0xef	0x15671c07
0x70	0xe1a43bc1	0xf0	0x5e7c3b3a
0x71	0xb2ce64	0xf1	0x65facfa6
0x72	0x50521ea4	0xf2	0xd12a4747
0x73	0xcaeb2941	0xf3	0x68d50115
0x74	0x1ac99814	0xf4	0x42b1bb85
0x75	0xcfa3fec8	0xf5	0x2c8dfbcd
0x76	0x5ef20043	0xf6	0x44de835
0x77	0x1b428b95	0xf7	0xd79b23c8
0x78	0x2ba7e075	0xf8	0xe99e1d27
0x79	0x52546210	0xf9	0x24bb1a25
0x7a	0xdfc10170	0xfa	0xf554b595
0x7b	0x83c82d24	0xfb	0x1e2e4521
0x7c	0xf8e03b04	0xfc	0x43aff7bd
0x7d	0xec032725	0xfd	0xa9473290
0x7e	0x1f5d3163	0xfe	0x656ee2ab
0x7f	0x80550b56	0xff	0xcfada464

Tabela D.2 – Valores para o S-Box 2.

Entrada	Saída	Entrada	Saída
0x0	0x7db6fcf	0x80	0x68bc4b8b
0x1	0x68336ffd	0x81	0xccec1140

0x2	0x1a1f4425	0x82	0x40b6d769
0x3	0x17b3d6e6	0x83	0x994e8163
0x4	0x8ef10de5	0x84	0x9ee992f8
0x5	0xd1e8e81c	0x85	0x939b7cb1
0x6	0x768c35fa	0x86	0xa1b25dcd
0x7	0xbefce1ed	0x87	0x556c3c63
0x8	0xf802aee2	0x88	0x289b3ce1
0x9	0x1d041230	0x89	0x2913afe
0xa	0xd9d1c8aa	0x8a	0x9340cd16
0xb	0x79144b96	0x8b	0xcd0ddd8
0xc	0x48f0c005	0x8c	0x38103e90
0xd	0x7720ef26	0x8d	0xd19e0779
0xe	0x3483e373	0x8e	0x8afa40a0
0xf	0xc44d1100	0x8f	0x87a5800b
0x10	0xfef3e59a	0x90	0x9b40a829
0x11	0x84867d30	0x91	0x6df4bed7
0x12	0xcafa7b12	0x92	0x22b9eba1
0x13	0xbea165b4	0x93	0xd6aadbc0
0x14	0xa0fe0240	0x94	0x7b94f1aa
0x15	0xf1c0d27b	0x95	0xe444b4ca
0x16	0x95c08d4b	0x96	0xa7d0b17a
0x17	0x9baef0c1	0x97	0xbdcd015b
0x18	0xe51eec87	0x98	0x30370496
0x19	0xac727c1a	0x99	0xb524ea68
0x1a	0x1b2cd37f	0x9a	0xf2535894
0x1b	0x580b7411	0x9b	0xb5e5f540
0x1c	0x848b218f	0x9c	0x99c8036e
0x1d	0x5f400aa9	0x9d	0x5a002a6e
0x1e	0x494ea3b	0x9e	0xf26a1828
0x1f	0xeb517d1	0x9f	0x552d0157
0x20	0x364c5c42	0xa0	0xc4daf1f3
0x21	0x98b4ee35	0xa1	0xcd9f45b1
0x22	0xc9a8d04	0xa2	0x859f156f
0x23	0x5f4343a6	0xa3	0x55eb1c73
0x24	0x795e8d6a	0xa4	0xfc9cc59e
0x25	0x599d601	0xa5	0xaefcb7de
0x26	0xd5994d1a	0xa6	0x577c3e08
0x27	0x1228f319	0xa7	0xf3deef2
0x28	0xf2d4448	0xa8	0x83a0daad
0x29	0x7eeeb1f4	0xa9	0xf9628215
0x2a	0x2e1b105c	0xaa	0x69d08fd1
0x2b	0x4200b406	0xab	0x9c176417
0x2c	0xe5c74fdd	0xac	0x2f3b0617
0x2d	0xa550b83b	0xad	0x36bb624e
0x2e	0xef85fd0b	0xae	0x7623c039
0x2f	0xb9ab431a	0xaf	0x330931b4
0x30	0xfa12b184	0xb0	0x3ec6b6a8
0x31	0x80cc18a6	0xb1	0xb7570683

0x32	0x3dbc9c3f	0xb2	0x348102a5
0x33	0x2352eb1a	0xb3	0x9ad24af
0x34	0x9fc6985c	0xb4	0x2ac11e77
0x35	0x4bca06e6	0xb5	0x2fc05d50
0x36	0xe324428c	0xb6	0xc6846944
0x37	0x727d79bf	0xb7	0xbaf09134
0x38	0x1c3672f9	0xb8	0x804d2c1d
0x39	0xaebc1d3a	0xb9	0x8fd17b1
0x3a	0x7bb30dde	0xba	0x787b7f96
0x3b	0xdf044159	0xbb	0x32facae0
0x3c	0x5c0847d3	0xbc	0xfbf4fe26
0x3d	0xc00d065d	0xbd	0x3b49bc54
0x3e	0x5e37d71d	0xbe	0x85c9f0cd
0x3f	0x3b7439fd	0xbf	0x33eeb267
0x40	0xc5117f08	0xc0	0x5e4c71e
0x41	0x841b0dab	0xc1	0x3307560e
0x42	0x3dac914a	0xc2	0xc7547372
0x43	0x838eb33b	0xc3	0x5e632867
0x44	0x3bb99758	0xc4	0x7fd8a14f
0x45	0xf3ebbd95	0xc5	0xe6c716e7
0x46	0xf57d8a1b	0xc6	0x42df8106
0x47	0x11b00ee3	0xc7	0x18d14196
0x48	0x55519987	0xc8	0x7cd190ea
0x49	0x91867eb4	0xc9	0x80da1b04
0x4a	0x841722be	0xca	0x3cb4a5fa
0x4b	0x411a5d5e	0xcb	0xaf005b6d
0x4c	0xb65190c	0xcc	0xcb15ebaf
0x4d	0x9d3bf563	0xcd	0xf5c22866
0x4e	0xca942527	0xce	0xb66aae59
0x4f	0x40efd0e9	0xcf	0x8f939a6d
0x50	0x2226f32e	0xd0	0xec88b90e
0x51	0xdcf4dfe6	0xd1	0x2ae8e836
0x52	0x88a8113f	0xd2	0x7006decd
0x53	0xef4b8e51	0xd3	0x747b9e09
0x54	0xc3fd6806	0xd4	0x45e505cf
0x55	0x14d83b91	0xd5	0x1a3d8145
0x56	0x66598dae	0xd6	0x1fdca62e
0x57	0xea8972ea	0xd7	0x1c358905
0x58	0xa126349f	0xd8	0x75f5a2c5
0x59	0x20e8b46	0xd9	0xa4bb087c
0x5a	0x67f465e0	0xda	0x38f20eea
0x5b	0xa42f5a92	0xdb	0xa0c94d8
0x5c	0x5770ebfd	0xdc	0xd7dd7f68
0x5d	0x5370ad05	0xdd	0x9377908d
0x5e	0xcb92b588	0xde	0xe43b45ee
0x5f	0x3363b3a2	0xdf	0x39467536
0x60	0x170a1c40	0xe0	0x9f043b8f
0x61	0x88110893	0xe1	0xd8374a5b

0x62	0x90ee7a17	0xe2	0xd60c2322
0x63	0x454f48b2	0xe3	0x5c763495
0x64	0xe19b8ed9	0xe4	0xe20ac498
0x65	0xa56f931d	0xe5	0xea826aef
0x66	0x374b9cb	0xe6	0x31be9208
0x67	0xe52dbb20	0xe7	0x53680c0f
0x68	0x613b5c06	0xe8	0x670db49c
0x69	0x54189004	0xe9	0xf22fe06c
0x6a	0x5d798ae0	0xea	0xaf8c057d
0x6b	0xbff64b46	0xeb	0xa3dba8d7
0x6c	0xd2738118	0xec	0xbba7aed8
0x6d	0x44660fcf	0xed	0x7fba146b
0x6e	0x3498b1f4	0xee	0x9f8a43c2
0x6f	0x5c04c719	0xef	0xbe35e825
0x70	0x9d792268	0xf0	0x69b4811
0x71	0x3dc5036c	0xf1	0xdcd2a36e
0x72	0xf3f9bb67	0xf2	0xa2df95ad
0x73	0x2a0176fb	0xf3	0xb2062738
0x74	0xf1301cb	0xf4	0x20146505
0x75	0x8070e981	0xf5	0x3975a6b5
0x76	0x97e2f5a2	0xf6	0x9887c5ea
0x77	0xbbc20d2	0xf7	0xfccbcebb
0x78	0xc2c15268	0xf8	0x8fa012c2
0x79	0xf85fdea3	0xf9	0x3de066d3
0x7a	0x7deb3a61	0xfa	0xafa0282f
0x7b	0xf60d42ba	0xfb	0x2f6dded0
0x7c	0x9132f637	0xfc	0x3be55e55
0x7d	0x7d0bcd26	0xfd	0xd10666d6
0x7e	0x9cab1178	0xfe	0x68eb568e
0x7f	0xd347b89b	0xff	0xb482befb

Tabela D.3 – Valores para o S-Box 3.

Entrada	Saída	Entrada	Saída
0x0	0x6ab1d1a7	0x80	0xa2695a69
0x1	0xafd5074e	0x81	0xd2f50c9
0x2	0x8122f68	0x82	0xe0e54652
0x3	0x188882a9	0x83	0xfe2cc5ca
0x4	0xfae52b08	0x84	0xe4e911d0
0x5	0x143fbeb9	0x85	0xae183c66
0x6	0x32ca1bc8	0x86	0xc507ba53
0x7	0x9c1f83ae	0x87	0xa63fa8dd
0x8	0x6db08225	0x88	0x3155de84
0x9	0x4c4218d5	0x89	0x593a0eac
0xa	0x227c699e	0x8a	0x74cb43c8
0xb	0x743d86cb	0x8b	0x45c51775
0xc	0x21b0ac3	0x8c	0x46ede758
0xd	0xc4eb5dd2	0x8d	0xbaa07c34

0xe	0xa4949caf	0x8e	0x34415d65
0xf	0x81e5b042	0x8f	0x9e09a46f
0x10	0x411ccedb	0x90	0x54b55427
0x11	0x7927173a	0x91	0x51559e0f
0x12	0x3e28f986	0x92	0x75d10118
0x13	0x5bc30af2	0x93	0x9d82d72d
0x14	0x2e6c3b74	0x94	0x73ff23ad
0x15	0x462f470d	0x95	0xd1a52277
0x16	0x53dda3f	0x96	0x418bddc
0x17	0x97228705	0x97	0xd2aa1c73
0x18	0x37b04787	0x98	0xd3843d3b
0x19	0x2de7bf6f	0x99	0xeb223d52
0x1a	0x98f5aae5	0x9a	0x857c5b6e
0x1b	0xb8b82b82	0x9b	0xcc6953b9
0x1c	0xfd4337bb	0x9c	0xdf283d51
0x1d	0xd38d32be	0x9d	0xd941b267
0x1e	0x3de3f298	0x9e	0xd2ae8628
0x1f	0xf0a1d7f5	0x9f	0x70acc7e1
0x20	0xa62342da	0xa0	0xcd6ac092
0x21	0x92bc65a4	0xa1	0xf2720619
0x22	0xd7028521	0xa2	0x6adaa662
0x23	0x59bb327d	0xa3	0x3ce44e70
0x24	0xb814919e	0xa4	0xcfa14886
0x25	0x9f2f2ffc	0xa5	0xd26157f8
0x26	0x90a1cc06	0xa6	0x819ecfe0
0x27	0xd945d1a7	0xa7	0xf5dda22a
0x28	0xd320204e	0xa8	0x71329190
0x29	0x1dbb8014	0xa9	0x25f06086
0x2a	0xb0a6fe69	0xaa	0xffd1d18a
0x2b	0xeb9a74d0	0xab	0x76bafb0b
0x2c	0x326a710	0xac	0x6f0b4036
0x2d	0x7838154c	0xad	0x45473692
0x2e	0xa90ba949	0xae	0xf6e7ccb6
0x2f	0x207920d1	0xaf	0x957b9871
0x30	0x9f648920	0xb0	0xed4d76be
0x31	0xb7dd95af	0xb1	0xdeae302
0x32	0x45de8261	0xb2	0xd59d731f
0x33	0xdb4a4199	0xb3	0x972a5340
0x34	0xcae2a2ad	0xb4	0xd9b9cf2b
0x35	0x50654428	0xb5	0x2127dcdc
0x36	0xce48ef84	0xb6	0x93c6d9cf
0x37	0xe93bdbd9	0xb7	0x2c88c750
0x38	0xe74193eb	0xb8	0xa80d8d87
0x39	0x2a7734c9	0xb9	0x664b39ce
0x3a	0xca74cc34	0xba	0xaed24dc0
0x3b	0x229940b4	0xbb	0x230d6d8e
0x3c	0x5f7fa458	0xbc	0x132b49ce
0x3d	0xb5865e2b	0xbd	0x5e7923b6

0x3e	0x34fc25b7	0xbe	0xaf9a122e
0x3f	0xb4f02f30	0xbf	0x23f0de04
0x40	0xa7a87cf2	0xc0	0x5181e44
0x41	0x5302c141	0xc1	0x509f6729
0x42	0x34c80bd7	0xc2	0x811c0a82
0x43	0xabf533c4	0xc3	0xaddf30f0
0x44	0x8e3fb015	0xc4	0x1c9a6706
0x45	0x75fe2f1e	0xc5	0x4da5459f
0x46	0xf950cdad	0xc6	0xd2b8d4c3
0x47	0x153f3ace	0xc7	0x9ffc6aec
0x48	0xc5dd5edf	0xc8	0xfc6eb41c
0x49	0x5b9b751a	0xc9	0x4ecfee8c
0x4a	0x6e349d59	0xca	0x273d38fa
0x4b	0x991aa611	0xcb	0x39a4f89a
0x4c	0x952ca391	0xcc	0x188e6552
0x4d	0x7ef6f6df	0xcd	0x26cca098
0x4e	0xb696a91b	0xce	0xaca3a29
0x4f	0xfe5f5b75	0xcf	0x5912f6c8
0x50	0xd82480a0	0xd0	0xa50d0694
0x51	0x8d6152ce	0xd1	0xa4cf7682
0x52	0xe5ce1954	0xd2	0x10d0b8f1
0x53	0x38dd217b	0xd3	0xea4da82b
0x54	0x91d6e5ef	0xd4	0xc9da8127
0x55	0xc88a6463	0xd5	0x6013921a
0x56	0xd810b31f	0xd6	0x70070419
0x57	0x14858e6a	0xd7	0x5ffa9e7b
0x58	0x34300ba3	0xd8	0xbc5b4253
0x59	0xb111dc9a	0xd9	0xfaec9ea0
0x5a	0xf490d22d	0xda	0xf46b94a
0x5b	0xde7c4302	0xdb	0xe9460905
0x5c	0xba6566f7	0xdc	0x26f215da
0x5d	0x6734bc87	0xdd	0x97fd6ed3
0x5e	0x4e8b2700	0xde	0x5159f820
0x5f	0xa09741a9	0xdf	0x1601d0a1
0x60	0x20b4183e	0xe0	0xeca72875
0x61	0xb2c3e069	0xe1	0x7eada2b
0x62	0x197afe38	0xe2	0x3c53cef
0x63	0x68590eca	0xe3	0x3a1f9a48
0x64	0x6d9bf36a	0xe4	0xf293843d
0x65	0x8b586f6	0xe5	0x8c7581e9
0x66	0x5324892a	0xe6	0xee8f098e
0x67	0x2937755d	0xe7	0xb04e8ee5
0x68	0xe94c0d65	0xe8	0x47619ac
0x69	0xae2e1736	0xe9	0xb7776813
0x6a	0x1453715a	0xea	0x6998068f
0x6b	0xa911eee8	0xeb	0xb4ce2839
0x6c	0x28f32606	0xec	0x56712585
0x6d	0x798ddcfe	0xed	0x2d9b630d

0x6e	0x844e9f1a	0xee	0x5267d571
0x6f	0x788a09d9	0xef	0x99ae12c
0x70	0x49501f27	0xf0	0x8e956e2e
0x71	0x86a510d0	0xf1	0x3638f631
0x72	0x99ce0e9d	0xf2	0x5a7948ad
0x73	0xd04e848f	0xf3	0xf21525d9
0x74	0x6dee638	0xf4	0xc171a35e
0x75	0xd050158a	0xf5	0xe7e06a86
0x76	0x8a609490	0xf6	0x35ebbf86
0x77	0x1940bdad	0xf7	0xc6902c6a
0x78	0x37c5001	0xf8	0xc9565445
0x79	0xb819eb33	0xf9	0x25a6012d
0x7a	0x45f420e5	0xfa	0xec589210
0x7b	0x2e00332f	0xfb	0xa58e7a62
0x7c	0x19c7f38b	0xfc	0x76efd361
0x7d	0xe4cc9ade	0xfd	0xfad1e673
0x7e	0x42b9e694	0xfe	0x5cd38bf5
0x7f	0x184a759d	0xff	0xa7133ee

Tabela D.4 – Valores para o S-Box 4.

Entrada	Saída	Entrada	Saída
0x0	0xc0ef85ca	0x80	0x730a039e
0x1	0xa19dd4cf	0x81	0x639cbb6f
0x2	0x9f228e62	0x82	0x2b66c66a
0x3	0x492b6dd9	0x83	0x414c2e04
0x4	0x376d5ae7	0x84	0xd79ee4de
0x5	0x216e9069	0x85	0xaf98424c
0x6	0x99cad3ba	0x86	0x36965e15
0x7	0x946fcde9	0x87	0x63711e3f
0x8	0xf95ef453	0x88	0x174fa88f
0x9	0x124b07b2	0x89	0xcbefba62
0xa	0x2564cd0f	0x8a	0x59662622
0xb	0xf3debe26	0x8b	0xf174e947
0xc	0x70ca9c9e	0x8c	0xd74444eb
0xd	0x9e27661d	0x8d	0x6da56982
0xe	0x3e813aa	0x8e	0x6e6c72b
0xf	0xeb88c572	0x8f	0xee7e4a62
0x10	0x423db6e7	0x90	0x2e691f9f
0x11	0x8995b5e0	0x91	0x78751d81
0x12	0x4dc8b29e	0x92	0xa0dc8ef5
0x13	0x1b777fcd	0x93	0x5c4f2286
0x14	0x68565f99	0x94	0x881028ad
0x15	0x24347146	0x95	0xa1cb287f
0x16	0x246a0ee	0x96	0x14aa93b8
0x17	0x56a28e20	0x97	0xba2a6ebd
0x18	0xaef9b3d8	0x98	0xfd141d69
0x19	0x345efb73	0x99	0x8590f38c

0x1a	0x6319432f	0x9a	0x499e6cdd
0x1b	0x96e28c90	0x9b	0xd2ccbc2d
0x1c	0xb9e2389d	0x9c	0x2e93adf9
0x1d	0xd769b2e6	0x9d	0x21a303c6
0x1e	0x700775ed	0x9e	0xd4183502
0x1f	0x3826b6a4	0x9f	0x4cd74b1f
0x20	0x2e6083b4	0xa0	0x86cd0abc
0x21	0x97c392ec	0xa1	0xd462cc77
0x22	0x56c051aa	0xa2	0x9c960231
0x23	0x1a377030	0xa3	0x84911a2
0x24	0x67f12859	0xa4	0xb0c17c7b
0x25	0x9c7d132c	0xa5	0x525de593
0x26	0x6ae410f9	0xa6	0x8e7d0b86
0x27	0x5e1ffb6e	0xa7	0xf2a8d002
0x28	0x499b205f	0xa8	0xdb2b5efc
0x29	0xe5a17133	0xa9	0x4515f542
0x2a	0x751b3251	0xaa	0xbc5b824e
0x2b	0xbbeb8d47	0xab	0xee12173a
0x2c	0x4d811e89	0xac	0xd3bc0fb9
0x2d	0xe01bd6b8	0xad	0x8047115a
0x2e	0xcf1e125	0xae	0x4a3e1998
0x2f	0x2ece61fb	0xaf	0xbead0bbb
0x30	0x5ad5cfb1	0xb0	0x48628c0
0x31	0x6ca5c99f	0xb1	0x31420e70
0x32	0x5f3f8d2c	0xb2	0xd2f7550e
0x33	0xa132258f	0xb3	0x954bf35f
0x34	0xbd1f664c	0xb4	0x729a3126
0x35	0x682650a	0xb5	0xc379ccbc
0x36	0x268cfad8	0xb6	0x5a6c7b42
0x37	0x55d8f7bf	0xb7	0x9b1510ad
0x38	0x54a7c4f6	0xb8	0xbaf0626f
0x39	0x2f0cf2ca	0xb9	0xefc18b1d
0x3a	0xb9c4ed23	0xba	0xcf7b8af
0x3b	0x2083afb4	0xbb	0xa4a13182
0x3c	0xd43ef910	0xbc	0x716d10d6
0x3d	0x9dbe3922	0xbd	0xdba8f8c9
0x3e	0xa5769c99	0xbe	0xc47989e9
0x3f	0x31603c8d	0xbf	0x3b7519bc
0x40	0x119261e7	0xc0	0x82b5c47d
0x41	0x33227b8d	0xc1	0x61c3e8c6
0x42	0x901765fe	0xc2	0x36b36a2a
0x43	0xc25326a	0xc3	0xf5404326
0x44	0xf814b8d8	0xc4	0xb5ceaf68
0x45	0x3473841f	0xc5	0xbd10262d
0x46	0xf83613ed	0xc6	0x3a5864bd
0x47	0x7bfa8360	0xc7	0x41c7fe06
0x48	0x698a0ec5	0xc8	0xb5ae281d
0x49	0xd2ab5ba0	0xc9	0x4de428cb

0x4a	0x42ea22f4	0xca	0x1ba52c23
0x4b	0x53db9040	0xcb	0xfc8c22d9
0x4c	0x480a62d4	0xcc	0x8df0ce01
0x4d	0x54e6c8ad	0xcd	0xc282c9a8
0x4e	0x39514dfb	0xce	0x6493131c
0x4f	0xd6c35601	0xcf	0xd0a8dcd5
0x50	0xcebb3da3	0xd0	0xb2f90e94
0x51	0x518328a3	0xd1	0x9c0732f6
0x52	0x58f49eb6	0xd2	0x1803fc51
0x53	0xfa63823d	0xd3	0x9621469e
0x54	0x69172e00	0xd4	0x9ddf5cb4
0x55	0x5ccfbabd	0xd5	0xcdce7936
0x56	0x3f45d7b2	0xd6	0x6f751567
0x57	0x3c4b94c6	0xd7	0x9ed1b74b
0x58	0xa002e93e	0xd8	0x8d992942
0x59	0x88425634	0xd9	0x3bec2075
0x5a	0x32790dd2	0xda	0xa06bfb58
0x5b	0x9deeb775	0xdb	0xc6d7aa98
0x5c	0xd1cff454	0xdc	0x60927c9
0x5d	0xaf8135c0	0xdd	0xcdfe51f7
0x5e	0xd1628423	0xde	0x956580d8
0x5f	0xa12601e4	0xdf	0x1bd1dbb3
0x60	0x8051e87e	0xe0	0x628fdf57
0x61	0xd6b9e84	0xe1	0x41c18e2b
0x62	0xffeab246	0xe2	0xdf917bc5
0x63	0xbf54a236	0xe3	0xfb5c3cbf
0x64	0x18188653	0xe4	0xb026119b
0x65	0x7b3c249d	0xe5	0xabb21106
0x66	0x469d2d68	0xe6	0x93d513a0
0x67	0x47be16d3	0xe7	0x77a40ba1
0x68	0x6b21e0e1	0xe8	0xcedac1a6
0x69	0x7f1087b8	0xe9	0x6ec4cdc3
0x6a	0x8f055a70	0xea	0x57318f2
0x6b	0x5b8e2770	0xeb	0x6b49ac99
0x6c	0xff4bb280	0xec	0x25172b33
0x6d	0xbe29c7a5	0xed	0x6eb0c92c
0x6e	0xa6af1adb	0xee	0x384ec517
0x6f	0xbd89a15d	0xef	0x8ab1281f
0x70	0xdf8880d1	0xf0	0x24650882
0x71	0xcc1442b	0xf1	0x629777ca
0x72	0xee38125d	0xf2	0x1e6fb71b
0x73	0xbaddb9b8	0xf3	0x615de965
0x74	0x7630755c	0xf4	0x17cf2ad7
0x75	0x49ef87cf	0xf5	0xb8d30549
0x76	0xb618ac0b	0xf6	0x6702f8af
0x77	0x3b71e4c7	0xf7	0x2102d87e
0x78	0x6cd8305e	0xf8	0xc627b0aa
0x79	0xdec1b6bb	0xf9	0x6e4d839f

0x7a	0x4a558abb	0xfa	0x7b14e9d5
0x7b	0x9db4ad	0xfb	0x3ff2be37
0x7c	0x4bb40e93	0xfc	0xe1cccf4a
0x7d	0x6446cb48	0xfd	0xc4dba8a2
0x7e	0xae92575b	0xfe	0xca6e32c6
0x7f	0x7bfd3688	0xff	0x1a080ede