UNIVERSIDADE ESTADUAL DE LONDRINA

CLEBER HENRIQUE DE OLIVEIRA

CRIPTOGRAFIA COM CAOS APLICANDO O SISTEMA DE CHUA

Londrina
2011

CLEBER HENRIQUE DE OLIVEIRA

CRIPTOGRAFIA COM CAOS APLICANDO O SISTEMA DE CHUA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual de Londrina como Parte dos Requisitos para a obtenção do Título de Mestre em Engenharia Elétrica.
Área de concentração: Sistemas Eletrônicos
Especialidade: Sistemas de Telecomunicações

Orientador: Prof. Dr. José Carlos Pizolato Junior

Londrina
2011

Catalogação elaborada pela Divisão de Processos Técnicos da Biblioteca Central da Universidade Estadual de Londrina.

Dados Internacionais de Catalogação-na-Publicação (CIP)

O48c Oliveira, Cleber Henrique de.
Criptografia com Caos aplicando o sistema de Chua / Cleber Henrique de Oliveira. – Londrina, 2011. 87 f.: il.

Orientador: José Carlos Pizolato Junior.
Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual de Londrina, Centro de Tecnologia e Urbanismo, Programa de Pós-Graduação em Engenharia Elétrica, 2011.
Inclui bibliografia, apêndice e anexos.

1. Sistemas eletrônicos de segurança – Teses. 2. Criptografia – Teses. 3. Sistema caótico de Chua – Segurança da informação – Teses. 4. Cifrador XTEA – Chaves criptográficas – Teses. I. Pizolato Junior, José Carlos. II.Universidade Estadual de Londrina. Centro de Tecnologia e Urbanismo. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU 621.391.9.05

CLEBER HENRIQUE DE OLIVEIRA

CRIPTOGRAFIA COM CAOS APLICANDO O SISTEMA DE CHUA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual de Londrina como Parte dos Requisitos para a obtenção do Título de Mestre em Engenharia Elétrica.

BANCA EXAMINADORA

_____

Orientador. Prof. Dr. José Carlos Pizolato Junior
Universidade Federal de São Carlos – UFSCar

_____

Prof. Dr. Taufik Abrão
Universidade Estadual de Londrina – UEL

_____

Prof. Dr. Mario Lemes Proença Junior
Universidade Estadual de Londrina – UEL

Londrina, 25 de fevereiro de 2011

Aos meus familiares:

Luiza, Maria Helena, Doralina, Berenice e Vanessa.

Presença constante em minha vida!

Agradecimentos

A Deus pela força, paciência e sabedoria dada por Ti.
Ao professor Doutor José Carlos Pizolato Junior, pela orientação.
Aos amigos que torceram por mim durante esta difícil caminhada.

OLIVEIRA, Cleber Henrique de. **Criptografia com caos aplicando o sistema de Chua**. 2011. 87 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual de Londrina, Londrina, 2011

RESUMO

Nos dias atuais, a necessidade de se proteger uma informação a ser armazenada ou transmitida, contra ataques de usuários que não são os verdadeiros destinatários do conteúdo ali existente, ou de pessoas cujo acesso a ela não está autorizado, prende a atenção de um número cada vez maior de pesquisadores envo lvidos na área de segurança. Por esse motivo, nas últimas duas déc adas, sistemas caóticos estão sendo aplicados na área de segurança da informação. O ponto que chama a atenção nesses sistemas, é devido ao com portamento complexo obtido, o que os torna atrati vos para aplic ações criptogr áficas. Nesse trabalho, o comportamento gerado pelo sistema caótico de Chua é us ado para c ifragem de informações, a fim de se chegar a um algoritmo de criptografia seguro. Para fins de explanação, tal algoritmo será referido, no trabalho, como “cifrador proposto”. Assim, uma chave criptográfica é usada na cifragem de cada componente da informação original. Uma característica importante deste cifrador corresponde à utilizaçã o de cada chave criptográfica apenas uma ve z. O desempenho do cifrador proposto é analisado ao ser aplicado sobre informações do tipo “texto, imagem, áudio e vídeo”. Para verificação da robustez do cifrador foram r ealizados os seguintes testes: aplic ações de funções hash, aplicaç ão da técnica de análise de freqüência, utilização da entropia de Shannon e anális e da sensibilidade do cifrador proposto para alteraçõe s das condiç ões iniciais do sistema caótico de Chua. Os desempenhos do cifrador proposto e XTEA, um algoritmo comerc ialmente utiliza do, são comparados e analisados. Os resultados obtidos nesta investigação demonstram que o cifrador proposto é funcional. Desta forma, tal cifrador poderia ser aplicado na criptografia da informação para sistemas de segurança.

Palavras-chave: Criptografia. Segurança. Sistemas caóticos.

OLIVEIRA, Cleber Henrique de. **Criptografia com caos aplicando o sistema de Chua**. 2011. 87 p. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual de Londrina, Londrina, 2011

ABSTRACT

Nowadays, the need to protect a piec e of information to be transmitted or stored away against attaches of those users w ho are not the actual receiver s of the contents, or from people whose access is not authorized, draws the attention of more and more researchers involved in this area of security. Therefore, in these two last decades, chaotic systems are applied in data security. The matter which s tands out in these s ystems is due to the comp lex behavior obtained what makes them attractive to cryptographic applications. In this essay, the behavior generated by Chua syst em is used to data encrypting in order to obtain a secure method of cryptography. For the purposes of exp lanation, such an alg orithm will b e referred, hereby, as “proposed cipher”. Thus, a cryptographic key is used in encrypting of each component of the original data. An important characteristic of this cipher corresponds to the using of each cryptographic key only once. The proposed cipher performance is analyzed as applied on data s uch as “text, image, audio and video”. To v erify the cipher’s strength the following t ests were carried out: applying of hash function, applying of frequency analysis technique, using the Shannon entropy and s ensibility analysis of the proposed cipher to alter chaotic system of Chua initial conditions. The performance of the proposed ci pher and of XTEA, an algorit hm of commerci al use, are compared and analyz ed. The res ults obtained through th is inv estigation demonstrate that the proposed cipher is functional. So, such a cipher could be applied to cryptography of data on security systems.

Keywords: Cryptography. Security. Chaotic systems.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ABNT	Associação Brasileira de Normas Técnicas
AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
CBC	Cipher Block Chaining
CERT	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
DES	Data Encryption Standard
ECB	Electronic Code Book
FEBRABAN	Federação Brasileira de Bancos
IBM	International Business Machine
INPE	Instituto Nacional de Pesquisas Espaciais
JPCS	Journal of Physics: Conference Series
MD1	Message Digest Algorithm 1
MD2	Message Digest Algorithm 2
MD3	Message Digest Algorithm 3
MD4	Message Digest Algorithm 4
MD5	Message Digest Algorithm 5
NIST	National Institute of Standards and Technology
SHA1	Secure Hash Algorithm
TEA	Tiny Encryption Algorithm
XTEA	Extended Tiny Encryption Algorithm

Lista de Símbolos

A	Parâmetro de bifurcação
A_c	Algoritmo de cifragem
A_d	Algoritmo de decifragem
a	Parâmetro do sistema de Chua
B	Bloco de bits
b	Parâmetro do sistema de Chua
C	Informação cifrada
CBC	Operação modo CBC (Cipher Block Chaining)
C_n	Componente da informação cifrada
CP	Cifrador proposto
c	Parâmetro do sistema Chua
D	Método de decifragem
E	Método de cifragem
F	Sistema de Chua no intervalo $F: X \rightarrow X$
$f(x_1)$	Função não-linear do sistema de Chua
i – *esimo*	I-ésimo caracter ASCII
K	Chave criptográfica
K_0	Chave inicial constituído de 12 dígitos inteiros no intervalo 0-9
$k1_0$	Parte da chave inicial constituído por 4 números decimais
$k2_0$	Parte da chave inicial constituído por 4 números decimais
$k3_0$	Parte da chave inicial constituído por 4 números decimais
K_c	Chave criptográfica de cifragem
$k_{C(r)}$	Chave criptográfica de cifragem (privada)
K_d	Chave criptográfica de decifragem
$K_{d(p)}$	Chave criptográfica de decifragem (pública)
K_n	Chave criptográfica originada da chave K
$L0$ e $R0$	Divisão do bloco de bits em duas partes
L_n	Valores (bits) resultantes de uma rodada
L_{n-1}	Valores (bits) resultantes da rodada anterior
P	Informação original
P_n	Componente da informação original

Q	Atacante (criptoanalista)
R_n	Valores (bits) resultantes de uma rodada
R_{n-1}	Valores (bits) resultantes da rodada anterior
S	Entropia de Shannon
t_0	Instante de tempo 0
t_{3000}	Instante de tempo 3000
t_{fc}	Instante de tempo final de cifragem
t_{fd}	Instante de tempo final de decifragem
t_{ic}	Instante de tempo inicial de cifragem
t_{id}	Instante de tempo inicial de decifragem
t_n	Instante de tempo n
TP	Operação troca de posições dos bytes
VI	Vetor de inicialização
x_1	Variável de estado do sistema de Chua
x_2	Variável de estado do sistema de Chua
x_3	Variável de estado do sistema de Chua
$x_1\,[0]$	Condição inicial da variável de estado x_1
$x_2\,[0]$	Condição inicial da variável de estado x_2
$x_3\,[0]$	Condição inicial da variável de estado x_3
X_{n+1}	Valor futuro de uma variável a partir da variável X_n
α	Parâmetro do sistema de Chua
β	Parâmetro do sistema de Chua

Sumário

1 INTRODUÇÃO

Segundo o CERT (2011a), a quantidade de indivíduos que passaram a ter acesso às facilidades da informática, se tornando usuários frequentes de computadores, aumentou sensivelmente nos últimos tempos, e neste caso, tais computadores, tanto aqueles que possuem destinação doméstica quanto os utilizados com finalidades empresariais, estão sendo direcionados para diversas tarefas, como envio e recebimento de informações via internet, armazenamento de dados, movimentações financeiras, etc.

Conforme dados levantados pela FEBRABAN (2011), os clientes com contas cujo acesso se dá pelo Internet Banking correspondem a cerca de trinta e cinco milhões no Brasil. Este quadro espantoso se deve ao crescimento do número de usuários que passaram a se utilizar da internet. Por outro lado, o volume de incidentes envolvendo sistemas de informação vem crescendo na mesma proporção. Nesse contexto, tanto os clientes quanto as instituições bancárias estão sensíveis a riscos de fraudes e preocupados com a frequência com que ocorrem. No último ano, dos incidentes virtuais reportados no Brasil, em torno de vinte e um por cento estão relacionados a fraudes com objetivo financeiro (CERT, 2011b).

Nesse contexto de segurança da informação, e a partir dos exemplos acima destacados, não há como deixar de reconhecer que a ciência criptográfica assume papel de crescente destaque e desponta como uma coadjutora dos profissionais que trabalham no ambiente computacional, garantindo a eficiência e o sucesso de suas realizações, e evitando que seus esforços sejam obstruídos por conta de atividade ilícita, a partir do momento em que não conseguem barrar o acesso ou o ataque de usuários não autorizados contra seus empreendimentos.

A criptografia pode ser definida como a ciência de ocultar a informação e, com isso, torná-la incompreensível diante de usuários não autorizados (SINGH, 2008). Em caminho inverso, e com a mesma dedicação dos profissionais anteriores, que se dedicam ao estudo dos métodos criptográficos, existem os decifradores, que são aqueles que tentam, de maneira não autorizada, decifrar informações cifradas, empregando processamento computacional.

Diante do acima exposto, vários grupos de pesquisa estão propondo métodos criptográficos com elevado grau de segurança. Seguindo este princípio, nas últimas duas décadas, pesquisadores têm-se dedicado profundamente à

investigação de sistemas caóticos e sua aplicabilidade na cripto grafia (BAPTISTA, 1998), objetivando elevar a segu rança devido ao comportamento complexo obtido . Sob essa ótica, os sistem as caóticos tornam-se uma grande alternativa par a geradores de chaves criptográficas.

1.1 MOTIVAÇÃO

Um dos sistemas caóticos mais estudados é o circuito caótico de Chua. Este sistema foi utilizado para mascarar informações de áudio e imagem num esquema de sincronização caótica (GUZMAN et al., 2008). No trabalho proposto por Rong He e Vaidya (1998), o comportamento caótico de um sistema caótico é usado para gerar chaves cr iptográficas. Nestes dois últimos resultados publicados, há a necessidade de sincronismo entre os sistemas transmissor e receptor.

Entretanto, há propostas de cifrador es caóticos que não necessit am de sincronismo entre o transmissor e o rec eptor. Alguns cifradores desse tipo foram propostos por Baptista (1998) e por O liveira e Sobottka (2008). O cifrador apresentado por Baptista (1998) propõe um modelo criptográfico baseado no comportamento caótico do mapa logíst ico (KADANOFF, 1983). Já o cifrador proposto por Oliveira e Sobottka (2008) segue o mesmo princ ípio do cifrador de Baptista (1998), mas com algumas melhor ias. Nesse caso, Oliveira e Sobottk a (2008) propõem resolver problemas encon trados no método proposto por Baptista (1998), como: não uniformidade da freqüência com que os intervalos associados aos símbolos do alfabet o utiliz ado são visita dos pela órbita esc olhida, limitação no sistema de Baptista (1998) no espaço de escolha do parâmetro para o sistema exibir o comportamento caótico, e dependênc ia do hardware devi do à sensibilidade aos métodos de arredondamento, pois conforme o número de iteraç ões aumenta, pode ocasionar perda de precisão no cálculo do mapa.

1.2 PROPOSTA

Neste trabalho, um novo cifrado r utilizando contribuições do cifrador apresentado por Rong He e Va idya (1998) é proposto. As características do cifrador de Rong He e Vaidya (1998) que foram preservadas são: o com portamento caótico

de um sistema caótico é usado para gerar chaves criptográficas, e cada chave é usada apenas uma vez.

Entretanto, os diferenciais do cifrador proposto em relação ao proposto por Rong He e Vaidya (1998) são:

i. Não necessita de sincronismo entre sistemas transmissor e receptor.
ii. Independência do alfabeto, isto é, utiliza o código ASCII como referência, e com isso eleva significativamente o grau de segurança e adaptabilidade do cifrador proposto.
iii. O comportamento obtido pelo sistema de Chua é usado para cifragem de dados.
iv. Portabilidade para diversos ambientes (desktop, laptop e diferentes canais de transmissões).

Desta forma, o cifrador proposto utiliza as órbitas geradas por um sistema caótico tridimensional, o sistema de Chua (GUZMAN et al., 2008), para cifragem de dados. O cifrador é classificado como simétrico de bloco, por cifrar blocos de bits e usar a mesma chave tanto para cifragem quanto decifragem de dados.

O trabalho é dividido em seis capítulos, que serão apresentados a seguir.

Os conceitos básicos sobre criptografia são apresentados no capítulo dois. Os sistemas caóticos e sua aplicação em criptografia serão abordados no capítulo três. O capítulo quatro apresenta o cifrador proposto. No capítulo cinco, são descritos e apresentados os testes experimentais. As conclusões a respeito do cifrador proposto estão presentes no capítulo seis.

1.3 PUBLICAÇÕES

Os resultados obtidos até o momento estão descritos em trabalhos apresentados em congressos e/ou revista da área.

1 – Artigo regular
C. H. Oliveira, J. C. Pizolato Jr., T. S. Pinto. *Cryptography with chaos.* In: Brazilian Conference on Dynamics, Control and Their Applications, 2010, Serra Negra. 9th Brazilian Conference on Dynamics, Control and Their Applications (DINCON'2010), 2010.

2 – Resumo expandido
C. H. Oliveira, J. C. Pizolato Jr. *Cryptography with chaos using Chua's attractor.* In: International Conference on Chaos and Nonlinear Dynamics, 2010, São José dos Campos. International Conference on Chaos and Nonlinear Dynamics (DDays'2010), 2010.

3 – Resumo expandido
J. C. Pizolato Jr., C. H. Oliveira, C. Gonçalves. *The electronic bouncing ball circuit in a communication system.* In: International Conference on Chaos and Nonlinear Dynamics, 2010, São José dos Campos. International Conference on Chaos and Nonlinear Dynamics (DDays'2010), 2010.

4 – Artigo regular (Aceito para publicação)
C. H. Oliveira, J. C. Pizolato Jr. *Cryptography with chaos using Chua's system.* Journal of Physics: Conference Series, Jan. 2011.

A seção 5.9 exibe observações sobre os trabalhos mencionados nesta seção. No próximo capítulo ser ão apresentados alguns conceitos básicos sobre criptografia.

2 CRIPTOGRAFIA

A segurança da informação consiste na proteção da informação contra acessos indevidos, alterações não autorizadas, tornando-a indisponível para quem o remetente assim o deseje. Para que esse objetivo seja alcançado, a segurança da informação se vale de alguns princípios básicos e fundamentais de proteção, entre os quais podem ser citados: a confidencilidade, a integridade e a disponibilidade (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2005), assim definidos da seguinte maneira:

- ✓ Confidencialidade: tem como objetivo garantir que a informação não tenha acessos não autorizados.
- ✓ Integridade: seu objetivo é proteger a informação contra alterações não autorizadas pela pessoa que deu origem a ela.
- ✓ Disponibilidade: tem por finalidade garantir a disponibilidade da informação para acessos autorizados.

O conjunto de procedimentos que envolve o conceito de segurança da informação se baseia na contínua busca de políticas de segurança destinadas à proteção da informação, evitando, sempre que possível, a ocorrência de prejuizos e incidentes desagradáveis nos mais diversos setores (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2005). Um dos mecanismos mais utilizados para a proteção da informação é a criptografia, que tem por objetivo a proteção da informação sigilosa, ocultando-a de pessoas às quais não se destina, ou que a ela não devem ter acesso.

2.1 HISTÓRIA

O termo “criptografia” tem origem no vocábulo grego “Kryptós” (que significa escondido, oculto), com a junção de “grápho”, que significa grafia, escrita. Esse mecanismo define-se como um conjunto de técnicas que permitem tornar incompreensível uma informação original, a fim de que apenas o detinatário final a decifre e a compreenda, extraindo dela o significado intentado pelo criador originário (SINGH, 2008).

A história e a evoluç ão da criptogr afia são marcadas por diversos acontecimentos, entre os quais alguns deles serão observados nesta seção.

Na antiguidade, o imperador rom ano Júlio César, tendo em mente a necessidade de cifragem de comunic ações governamentais, com vistas à preservação dos inter esses do grande im pério por ele comandado, ide alizou um método de cifragem baseado em substitu ição de caracteres. Para atingir seu objetivo, valeu-se de um método cuja lógica consistia em desviar as letras em três posições, de sorte que a inf ormação orig inária poderia s er obtida mediante substituição de uma letra pela que ocupa a terceira posição antecedente (SINGH, 2008).

A figura 2.1 exibe o processo de s ubstituição do método de cifragem idealizado por Júlio César:

Figura 2.1 – Lógica de substituição usado no método de César.

Fonte: (SINGH, 2008)

A lógica desse método de substituição é a única da antiguidade ainda em uso nos tempos modernos (SINGH, 2008).

Adentrando ao século XIX, nota- se que em 1840, Samuel M orse idealizou o que seria c onhecido até hoje como o Código Mors e, sistema por meio do qual letras, números e sinais de pontuação são representados por um sinal cifrado, enviado intermitentemente. Na v erdade, o que se constata é que esse método de transmissão de informações não pode ser chamado de código, mas sim, um alfabeto cifrado, consistente na emissão de sons curtos e longos para cada finalidade.

Caminhando para o final do séc ulo XX, em 1976, a empresa I BM idealizou o cifrador inicia lmente chamado de Lucifer, e após algumas modificações, tornou-se o cifrador DES (Data Encryption Standard) (BERENT, 2003).

Com a evolução s empre constante dos c omputadores, chegou-se a um ponto em que o cifrador DES não conseguia mais garantir a segurança e inviolabilidade das informações que eram por ele cifrados. Diante desse quadro

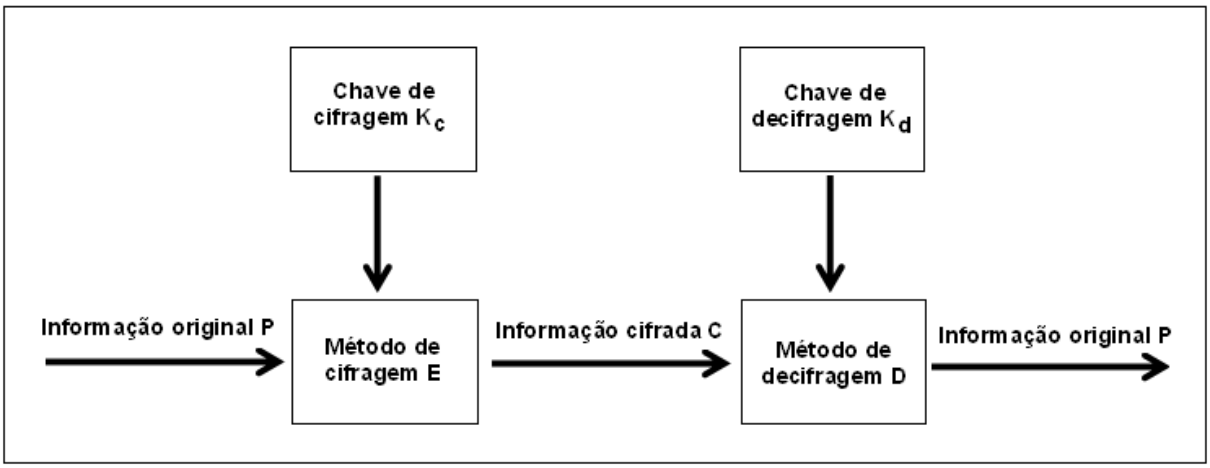
desfavorável, em 1997 foi apr esentado o su cessor do cifrador DES: trata-se do algoritmo Rijndael, q ue acabou por ser batizado de AES (Ad vanced Encryption Standard), passando a ser utilizado como padrão de cifragem do gover no norte-americano (BERENT, 2003).

2.2 Modelo Genérico de um Cifrador

Como já mencionado anteriormente, o objetivo primordial de um cifrador é proteger a informação a ser trans mitida ou armazenada diante de usuários não autoriz ados. A segurança do cifrador é obtida através de métodos de c ifragem que empregam chav es para cifragem e decifagem, e dessa combinação é que depende a sua robus tez. O diagrama em bloc os de um cifrador está ilustrado na figura 2.2.

O caminho percorrido por um cifrador consiste em cifrar uma informação original chamada de informação P, aplicando-se um método de cifragem chamado de E, que rec ebe como entrada a própria informação P e uma chave de cifragem chamada de K_c, do que resulta uma informação cifrada, chamada de informação C. Para d ecifrar a informação ci frada utiliza-se um método de decifragem chamado de D, que recebe como entrada a informação cifrada e uma chave de decifragem chamada de K_d e fornece como saída a recuperação da informação P.

Figura 2.2 – Modelo genérico de um cifrador.



Os métodos de cifragem E e decifragem D normalmente são diferentes. Caso as chaves de cifragem K_c e decifragem K_d sejam iguais, está-se então diante de um sistema chamado de chaves simétricas. Quando, ao contrário, essas chaves são diferentes, tem-se um sistema de chaves assimétricas, também chamada de “chave pública”.

2.3 CLASSES DE ALGORITMOS

Os cifradores de chave simétrica são assim definidos por utilizar em a mesma chave, tanto no processo de cifragem, quando por acasião da decifragem. Dessa maneira, K_c e K_d são de domínio restrito (privado). Por sua vez, os cifradores simétricos são divididos em dois conjuntos: os cifradores de fluxo e os cifradores de bloco (SCHNEIER, 1996).

Os cifradores de fluxo têm como característica a cifragem bit a bit ou (byte a byte) por iteração do algoritmo. Nesse caso, o bit (ou byte) de entrada gera um bit (ou byte) diferente, na saída do algoritmo. Já os cifradores de bloco geralmente operam cifrando blocos de bits por iteração do algoritmo. Dessa forma, o bloco de bits original é diferente do bloco de bits cifrado (SCHNEIER, 1996).

Os cifradores de fluxo e os cifradores de bloco podem operar de diversos modos. Esses modos de operação têm a função de realizar um pré-tratamento dos bits de entrada do algoritmo de cifragem. Além disso, eles adicionam características de dependência entre a sequência de blocos de bits cifrados, e atuando dessa maneira, tornam o cifrador mais robusto (STALLINGS, 2002).

A seguir, são descritos dois modos de operação com o objetivo de evidenciar a influência que o modo de operação tem no resultado final da cifragem, quando combinado com as operações de criptografia.

- ECB - (Electronic Code Book)

No modo de operação ECB, cada bloco de $nbits$ original é individual e independentemente cifrado para gerar os blocos de $nbits$ cifrados (STALLINGS, 2002). Diante disso, o modo ECB é descrito pelas funções do tipo:

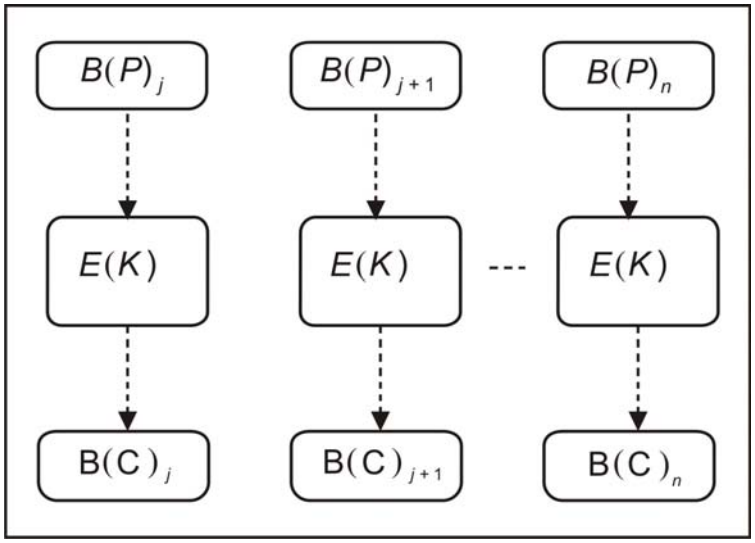
$B(C)_j = E(K(B(P)_j))$ (Cifragem) $B(P)_j = D(K(B(C)_j))$ (Decifragem), (2.1)

onde

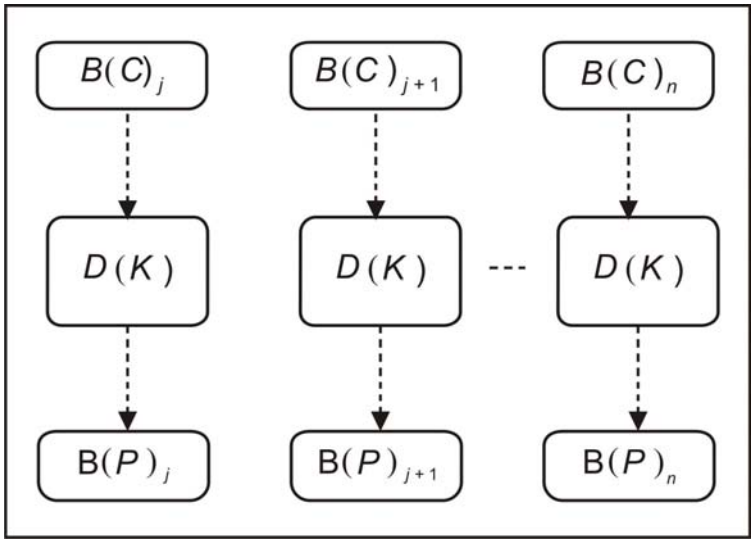
- ✓ B é o bloco de bits

A figura 2.3 ilustra o modo de operação ECB utilizado para tratamento dos dados de entrada no cifrador.

Figura 2.3 – Utilização do modo de operação ECB. (a) Processo de cifragem. (b) Processo de decifragem.



(a)



(b)

- CBC - (Cipher Block Chaining)

O modo de operação CBC atua realimentando a cifragem do bloco de bits atual com o resultado da cifragem dos blocos de bits anteriores, realizando a operação XOR entre bloco atual e o bloco anteriormente cifrado. Assim, blocos iguais serão cifrados de maneira diferente, isto é, de forma sequencial. O primeiro bloco a ser cifrado utiliza um vetor VI, denominado vetor de inicialização, contendo a mesma quantidade de bits dos blocos, de conhecimento prévio do transmissor e do receptor (STALLINGS, 2002). O modo CBC é descrito a seguir pelas funções:

(Cifragem)

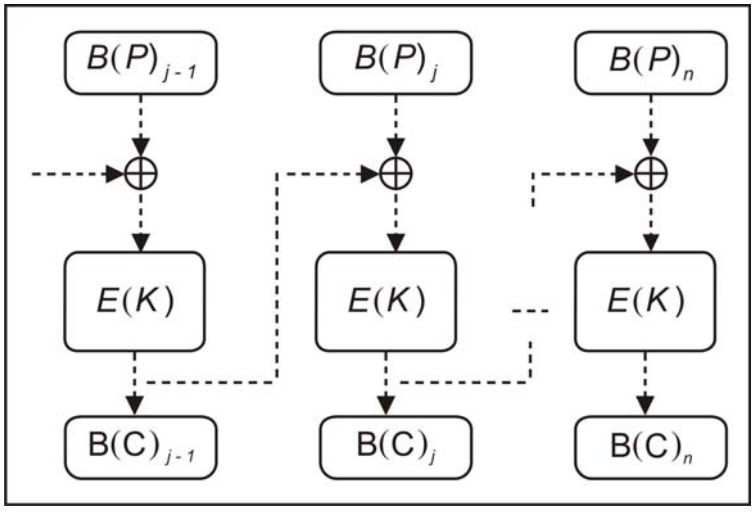
$$E_1 = VI \oplus B^1(P^1(E)) = B^1(C^1),\ E_2 = B^1(C^1) \oplus B^2(P^2(E)) = B^2(C^2),...,E_n \quad (2.2)$$

(Decifragem)

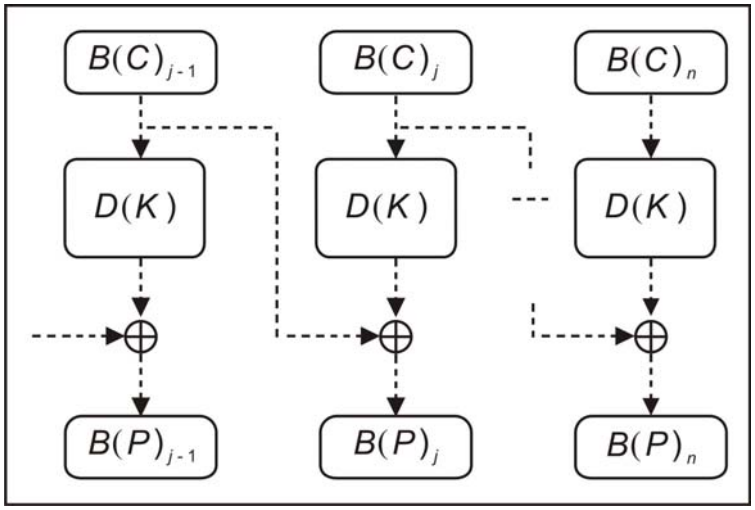
$$D_1 = B^1(C^1(D)) \oplus VI = B^1(P^1),\ D_2 = B^2(C^2(D)) \oplus B^1(C^1) = B^2(P^2),...,D_n \quad (2.3)$$

A figura 2.4 ilustra o modo de operação CBC utilizado para o tratamento dos dados de entrada no cifrador.

Figura 2.4 – Utilização do modo de operação CBC. (a) Processo de cifragem. (b) Processo de decifragem.



(a)



(b)

Adentrando agora ao exame dos cifradores de chave assimétrica ou de chave pública, tem-se que eles se caracterizam por utilizar duas chaves diferentes nos processos de cifragem e decifragem. Tais chaves são conhecidas como chave pública e chave privada, sendo que a primeira é usada no processo de cifragem, e a segunda utilizada no procedimento de decifragem. Enquanto a chave de cifragem costuma ser de domínio público, a chave de decifragem deve ser mantida em sigilo absoluto (STALLINGS, 2002).

Para isso são utilizadas as funções:

$C = E(P, K_c)$ (Cifragem) e $P = D(C, K_d)$ (Decifragem), (2.4)

Geralmente, as ações dos métodos E e D sobre P e C dependem respectivamente das chaves criptográficas K_c e K_d. O conceito de chave assimétrica foi idealizada por Whitfield Diffie e Martin Hellman em meados da década de 70, quando publicaram os primeiros artigos sobre o assunto. Deste conceito surgiu o método denominado Diffie-Hellman, um algoritmo usado para a distribuição de chaves. Tal método resolve o problema de compartilhamento de chave. Contudo, não é um algoritmo final de criptografia, pois ele não cifra informação (BURNETT; PAINE, 2002).

Dando seguimento aos conceitos utilizados por aqueles estudiosos, Ron Rivest, Adi Shamir e Len Adleman se aprofundaram no estudo do referido

método, desenvolvendo um algoritmo de criptografia com chaves assimétricas que ficou conhecido como RSA (BURNETT; PAINE, 2002).

Mediante utilização das chaves assimétricas foram resolvidos os problemas de distribuição de chaves. E outros dois obstáculos que foram, também, simultaneamente superados, são a autenticação e o não repúdio, posto que garantidos pela assinatura digital (BURNETT; PAINE, 2002).

Como já mencionado, em cifradores assimétricos a chave pública é utilizada para cifragem da informação, e assim apenas o receptor com a chave privada poderá decifrá-la. Neste caso, mudando o papel das chaves, isto é, usando a chave privada para domínio público e a chave pública com domínio restrito (privado), situação em que o processo de cifragem torna-se de domínio restrito e o processo de decifragem de domínio público, tal mudança resultará em um efeito de autenticação, isto é, assinatura da informação. Seguindo esta ótica, teremos o mecanismo denominado de assinatura digital (BURNETT; PAINE, 2002).

Para isso são utilizadas as funções:

$C = E(P, K_{c(r)})$ (Cifragem) e $P = D(C, K_{d(p)})$ (Decifragem), (2.5)

onde

- ✓ $K_{c(\tau)}$ é a chave de cifragem (privada)
- ✓ $K_{d(p)}$ é a chave de decifragem (pública)

Uma outra maneira de obter uma "assinatura," isto é, um valor que represente o conteúdo de uma informação, ocorre por meio da função *hash.* Tal função consiste em gerar um fluxo de dados de tamanho fixo denominado resumo de mensagem (message digest). Para isso, a função utiliza uma informação independente do tamanho para gerar um resumo de tamanho fixo. Existem diversos algoritmos para este propósito, tais como: (MD1, MD2, MD3, MD4, MD5 e SHA1). O tamanho do resumo de mensagem está relacionado com as características do algoritmo. Como exemplo, o algoritmo MD 5 (Message Digest Algorithm 5) gera um resumo de mensagem de tamanho fixo de 128 bits. O SHA1 (Secure Hash Algorithm) gera um resumo de 160 bits (BURNETT; PAINE, 2002; LEE, 2007).

Uma característica importante desses algoritmos é que qualquer alteração dos bits na informação resultará em valores significativamente diferentes do resumo de mensagem. Seguindo este princípio, a função *hash* é um mecanismo importante e eficiente para analisar a integridade da informação.

2.4 Estrutura dos Cifradores Modernos

Segundo Shannon (1949), a estrutura dos cifradores modernos geralmente se baseia na ótica de um criptoanalista, o que significa dizer que as características do cifrador são fundamentadas na segurança contra possíveis ataques sobre a chave ou a informação cifrada. Na esteira dessa linha de raciocínio, surgiram os princípios de difusão e confusão, aplicados nos cifradores modernos. O conceito de difusão dissipa as características estatísticas entre a informação original e a informação cifrada. A confusão elimina as características estatísticas entre a informação cifrada e a chave criptográfica (SHANNON, 1949).

Diante disso, os princípios de difusão e de confusão garantem a segurança do cifrador contra ataques refinados.

De boa medida ressaltar que existem inúmeros mecanismos utilizados pelos cifradores simétricos para garantir a difusão e confusão entre os dados.

A tabela de substituição é um importante exemplo desses mecanismos, utilizado para gerar a difusão, com objetivo de introduzir a não-linearidade no cifrador. Esta tabela geralmente é usada em cifradores de bloco, e estruturada por uma matriz $n \times n$ constituída por símbolos. No caso do algoritmo AES, utiliza-se uma matriz bidimensional $(16x16)$, composta por símbolos hexadecimais $(0-F)$. O mapeamento é realizado com N bits de entrada em N bits de saída. No AES, 32 bits de entrada, com 32 bits de saída, a substituição é realizada sobre cada byte do bloco (WEBSTER; TAVARES, 1986; DAEMEN; RIJMEN, 2001).

No processo de cifragem, o resultado da operação gera os bits correspondentes aos bits de entrada. Caminhando em sentido contrário, o processo de decifragem, utiliza-se da matriz inversa (WEBSTER; TAVARES, 1986; DAEMEN;

RIJMEN, 2001). As figuras 2. 5-6 exibem a estrutura da tabela de s ubstituição utilizada no algoritmo AES.

Figura 2.5 – Tabela de substituição (Caixa-S) utilizada no processo de cifragem.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fonte: (DAEMEN; RIJMEN, 2001)

Figura 2.6 – Tabela de substituição (Caixa- S) inversa utilizad a no proces so de decifragem.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

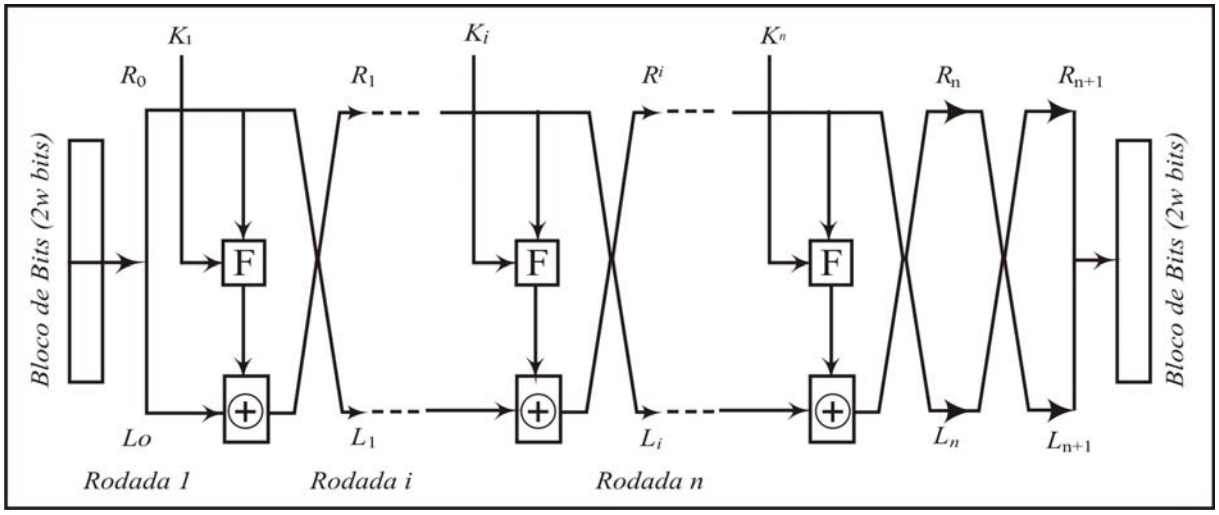
Fonte: (DAEMEN; RIJMEN, 2001)

A estrutura das redes Feistel é um outro mecanismo utilizado p ara garantir a segurança da informa ção. Ela foi um dos p rimeiros mecanismos criados

para promover a difusão entre os dados. A estrutura da rede inicia com a divisão do bloco de dados a ser cifrado em duas partes, aqui denominados $L0$ e $R0$.

Portanto, em cada rodada n, o cifrador recebe os valores resultantes da rodada anterior L_{n-1} e R_{n-1}. As rodadas são estruturadas com chaves K_n originadas da chave K. Diante disso, aplica-se a função F sobre o bloco R_{n-1}, na forma $F(R_{n-1}, K_n)$. No final da rodada aplica-se a operação XOR entre L_{n-1} e $F(R_{n-1}, K_n)$, de modo que no final da operação gera R_n e o valor L_n corresponde a R_{n-1}. Os valores resultantes L_n e R_n são equivalentes a um a rodada (STALLINGS, 2005). A figura 2.7 exibe a estrutura das redes Feistel utilizada em cifradores simétricos.

Figura 2.7 – A estrutura da rede Feistel.



Fonte: (STALLINGS, 2005)

2.5 CHAVES CRIPTOGRÁFICAS

O termo “chave criptográfica” é definido como um valor secreto utilizado em cifradores para cifragem e decifragem de informação. A chave é considerada um dos pontos fracos de um cifrador, ao lado dos algoritmos de cifragem/decifragem envolvidos. Para que um cifrador seja considerado robusto, deve ele garantir as diferenças estatísticas entre a informação original, a sua versão cifrada e a chave (SHANNON, 1949; MENEZES; VANSTONE; OORSCHOT, 1996).

Um tipo de chave utilizado em cifr adores é a chave de comprimento (bits). Neste caso, o tamanho da chave es tá relacionado ao ní vel de robustez, de modo que, quanto maior o comprimento da c have (bits), maior a robustez. Por outro lado, quant o maior a chave (bits), consequentemente maior o tempo de execuç ão dos proces sos de cifrafem/decifragem. Nest e caso, deve-se obs ervar qual o nível mais adequado de s egurança que se pode te r sem comprometer o desempenho. Seguindo este princípio, uma chave de c omprimento necessit a ser igual ou maior (bits) em relação ao bloco de dados para se obter níveis satisfatórios de s egurança (BURNETT; PAINE, 2002).

Outro tipo de chave criptográfic a que vem sendo profundamente investigado por pesquisadores para sua ap licabilidade na criptogr afia é baseada nos sistemas caóticos (BAPTIST A, 1998). O que motiva essa investiga ção é o comportamento obtido por esses sistemas, que se caracterizam pela irregularidade. Com isso, não preservam os padrões repetitivos da informação original, o que faz com que essa abordagem melhore sensivelmente a criptografia. Sob esse prisma, os sistemas caóticos tornam-se uma grande alternativa para ger adores de chaves criptográficas

2.6 CRIPTOANÁLISE

A busca pela segurança lev ou as empresas e governos a criarem setores destinado ao desenvolv imento de ci fradores responsáveis por garantir a segurança das informações, cabendo a quem atua no setor inve ntar e utilizar cifradores cada vez mais segur os. Por outro lado, cresce a cada dia o número de decifradores de códigos que tentam quebrar o sigilo criado pelos cifradores, que estão sempre sob ataque (SINGH, 2008).

Os decifradores de códigos, aqui definidos como criptoanalis tas, utilizam in úmeras técnicas para tentar deduzir a forma como o cifrad or tornou ininteligível a informação, frente às pess oas não autorizadas que porventura tenham acesso a ela. Diante disso, o objetivo de um criptoanalista é det erminar uma técnica que possa deduz ir a cifragem, tornando- a novamente inteligí vel, da for ma como inicialmente idealizada (SINGH, 2008).

A segurança e o nível de sucess o de um cifrador são avaliadas pelo tipo de ataque que ele possa s ofrer sem que seja decifrado (SCHNEIER, 1996).

Logo a seguir serão analisados alguns tipos de ataques. Para esta análise são utilizadas as seguintes representações:

A_c é o algoritmo de cifragem

A_d é o algoritmo de decifragem

E é o método de cifragem

D é o método de decifragem

P é a informação original

C é a informação cifrada

K a chave criptográfica

Q é o atacante (criptoanalista)

As análises abordadas adiante pressupõem que o criptoanalista tem acesso ao A_c, porém desconhece K (SCHNEIER, 1996).

- Ataque de força bruta

Nessa modalidade, o atacante tem acesso à informação cifrada, porém, não tem conhecimento da chave criptográfica. O objetivo do criptoanalista, por conseguinte, será o de encontrar a chave criptográfica através de inúmeras tentativas.

Dados, $C = A_c = E(P, K)$

Decifragem, $Q \rightarrow K$

- Ataque do texto cifrado

Já nessa situação, o atacante tem acesso a um número elevado de informações cifradas, no entanto, não tem acesso às informações originais e às chaves criptográficas correspondentes. O princípio deste ataque consiste em encontrar as informações originais e as suas respectivas chaves.

Dados $C^1 = A_c = E(P^1, K^1), C^2 = A_c = E(P^2, K^2), ..., C^j = A_c = E(P^j, K^j)$

Decifragem, $Q \rightarrow P^1, P^2, ..., P^j$ e $K^1, K^2, ..., K^j$

- Ataque do texto conhecido

Nesse tipo, o atacante tem acesso a diversas informações cifradas e suas respectivas versões originais. Seu objetivo, agora, será o de decifrar as chaves criptográficas correspondentes.

Dados, $C^1 = A_c = E(P^1, K^1), C^2 = A_c = E(P^2, K^2), ..., C^j = A_c = E(P^j, K^j)$

Decifragem, $Q \rightarrow K^1, K^2, ..., K^j$

- Ataque adaptativo do texto escolhido

Nessa vertente, o atacante tem acesso a um número limitado de informações originais e suas respectivas versões cifradas. Nesse contexto, através das análises e resultados, o atacante terá acesso a outro grupo de informações, e assim sucessivamente. A finalidade deste ataque é encontrar as chaves correspondentes ou um método que quebre o sigilo criado.

Dados, $C^1 = A_c = E(P^1, K^1), C^2 = A_c = E(P^2, K^2), ..., C^j = A_c = E(P^j, K^j)$

Decifragem, $Q \rightarrow K^1, K^2, ..., K^j$ ou $A_c = E$

- Ataque do texto cifrado escolhido

Para essa situação, o atacante tem acesso a um número elevado de informações originais e suas respectivas versões cifradas. De posse delas, ele pode produzir uma informação cifrada, e assim ter acesso ao resultado produzido. O objetivo deste ataque é encontrar as chaves correspondentes ou um método para quebrar o sigilo criado.

Dados, $C^1 = A_c = E(P^1, K^1), C^2 = A_c = E(P^2, K^2), ..., C^j = A_c = E(P^j, K^j)$

Decifragem, $Q \rightarrow K^1, K^2, ..., K^j$ ou $A_c = E$

- Ataque da chave escolhida

Quando ocorre esse ataque, é possível para o atacante testar o cifrador através de inúmeras chaves ou convencer determinados usuários do cifrador

a utilizare m chaves específicas. A finalid ade deste a taque é en contrar as chaves correspondentes ou um método para quebrar o sigilo criado.

Dados, $C^1 = A_c = E(P^1, K^1), C^2 = A_c = E(P^2, K^2), ..., C^j = A_c = E(P^j, K^j)$

$P^1 = A_d = D(C^1, K^1), P^2 = A_d = D(C^2, K^2), ..., P^j = A_d = D(P^j, K^j)$

Decifragem, $Q \rightarrow K^1, K^2, ..., K^j$ ou $A_c = E$

Partindo das análises tratadas neste capítulo, observ ou-se que as ações dos algoritmos A_c e A_d sobre P e C dependem respectivamente das chaves criptográficas K_c e K_d, ou seja, a robustez do cifrador está intimamente relacionada com a quantidade e a complexidade das chaves usadas na cifragem da informação.

Nesse pas so, os sistemas caóticos tornam-se uma alternativ a de relevante importância para geradores de chaves criptográficas devido ao comportamento complexo obtido. No próx imo capítulo será apresentada uma breve discussão sobre s istemas caóticos. O si stema caótico de Chua, o qual será empregado no cifrador proposto, é apresentado e analisado.

3 SISTEMAS CAÓTICOS

Pesquisadores em dinâmica não–linear usam o termo “caos” para se referirem à irregularidade e ao comportament o imprevisível de sistemas dinâmicos não-lineares, os quais podem exibir um co mportamento que se estende do periódico ao caótico (KADANOFF, 1983; GLEICK, 1990).

Um exemplo de sistema dinâmico é o mapa logístico, definido como determinístico, discreto e unidimensi onal (KADANOFF, 1983). Este sistema é descrito matematicamente pela equação 3.1.

$$X_{n+1} = AX_n(1 - X_n) \tag{3.1}$$

onde: X_{n+1} representa o valor futuro de um a variável a partir da variável X_n (n inteiro e positivo) em intervalos discretos de tempo ΔT. À medida em que o parâmetro A é incrementado, a evoluç ão da sequência X_n, gerada por esta equação exibe uma extraordinária transição para c aos. O termo A é chamado parâmetro de bifurcação do sistema.

A variação do parâmetro A da equação 3.1 permite a observação de regimes dinâmicos do sistema. As figur as 3.1-2 ilustram o comportamento do sistema para determinados v alores do parâmetro de bifuração A. Quando o parâmetro A ultrapassa o valor 3 o valor de X_n oscila entre dois valores, conform e pode ser observado na figura 3.1 (a). Ne ste caso, ocorreu uma situação de bifurcação do sistema. Na condição em que o parâmetro A atinge o valor de 3,5 o valor de X_n oscila entre quatro valores, c onforme pode ser observado na figura 3.1 (b). Isto ocorre devido a uma nova bifurcação do sistema para $A = 3{,}5$. Tal sequência de duplicação de per íodo vai ac ontecendo mais rapidamente conforme o parâmetro A é increm entado at é que o número de órbi tas instáveis s eja infinito . Este comportamento é ilust rado na figura 3.1 (c). O comportamento genérico do s istema descrito pela equaç ão 3.1 pode ser suma rizado no gráfico do mapa logístico ilustrado na figura 3.2. Nessa figura é pos sível observar as situações de período 1, período 2, período 4 até a situação caótica.

Figura 3.1 – Análise do comportamento do mapa logístico. (a) Parâmetro $A = 3{,}1$ e condição inicial $x[0] = 0{,}2$. (b) Parâmetro $A = 3{,}5$ e condição inicial $x[0] = 0{,}2$. (c) Parâmetro $A = 4{,}0$ e condição inicial $x[0] = 0{,}2$.

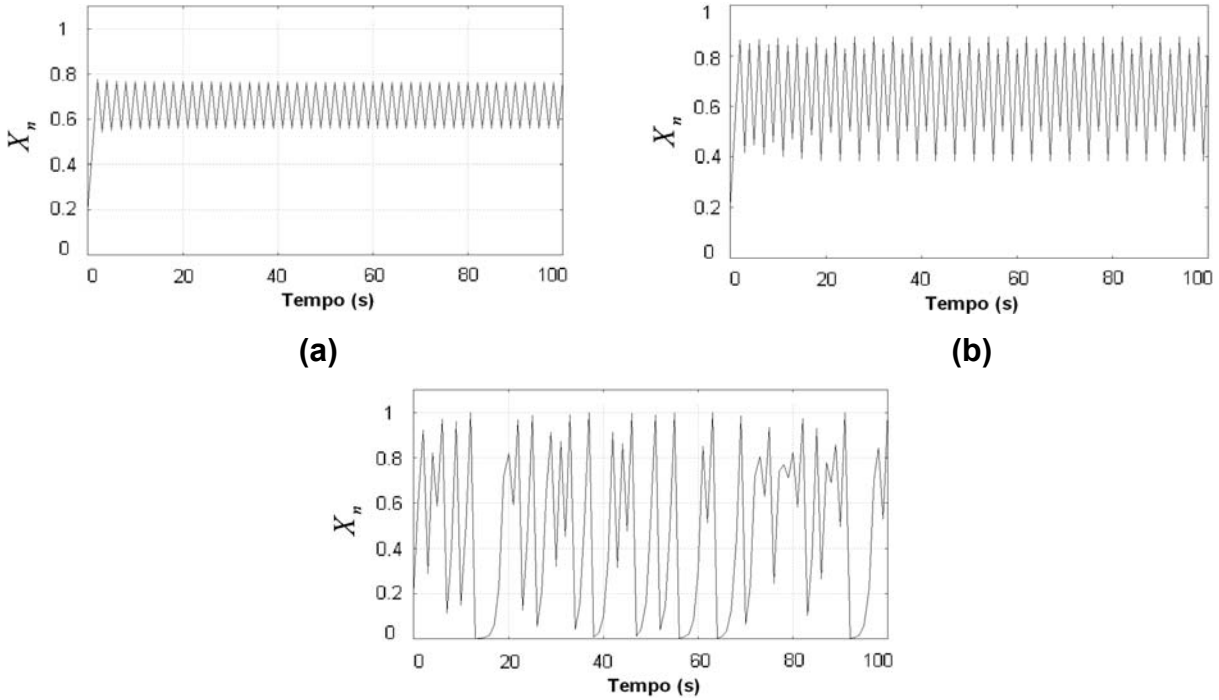
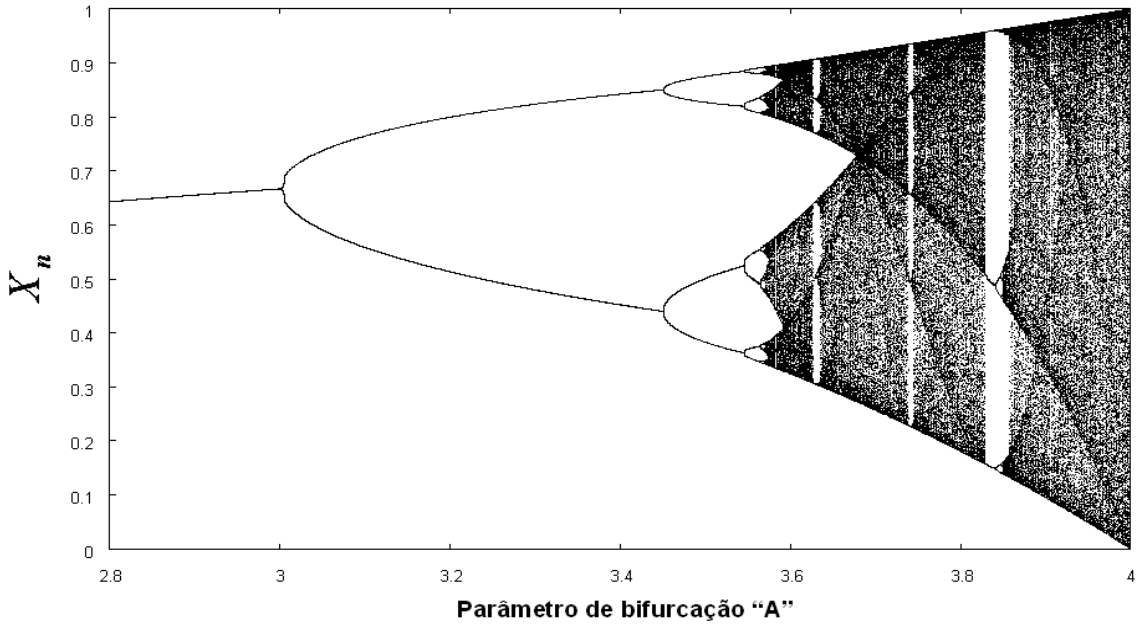


Figura 3.2 – Diagrama de bifurcações em função do parâmetro A no intervalo $2{,}8 \leq A \leq 4{,}0$ e condição inicial $x[0] = 0{,}1$. Parâmetro $A = 2{,}9$ (ponto fixo), $A = 3{,}1$ (período 2), $A = 3{,}5$ (período 4) e $A > 3{,}57$ (aperiódico(caótico)).



Conforme observado nas figuras 3.1 (c) e 3.2, para valores de $A > 3{,}57$ o comportamento da sequência X_n, ao longo do tempo, é aperiódico (caótico) e varia continuamente em intervalos de X. A evolução de X_n neste intervalo, é aleatória, mesmo sendo o mapa logístico totalmente determinístico.

Um dos sistemas dinâmicos que exibe o comportamento caótico é o sistema de Chua (GUZMAN et al., 2008). Este sistema vem sendo explorado em muitos trabalhos (GUZMAN et al., 2008; ALLIGOOD; SAUER; YORKE, 1996; MADAN, 1993) e será aplicado no cifrador proposto. A seguir será feita uma abordagem do sistema de Chua.

3.1 O SISTEMA DE CHUA

O sistema de Chua (GUZMAN et al., 2008) é descrito matematicamente pelas equações de estado 3.2-4.

$$\dot{x}_1 = \alpha(x_2 - f(x_1)) \quad (3.2)$$

$$\dot{x}_2 = x_1 - x_2 + x_3 \quad (3.3)$$

$$\dot{x}_3 = -\beta x_2 \quad (3.4)$$

onde: α e β são parâmetros do sistema, x_1, x_2 e x_3 as variáveis de estado e $f(x_1)$ é a função não-linear definida pela equação 3.5.

$$f(x_1) = bx_1 + \frac{1}{2}(a-b)(|x_1 + c| - |x_1 - c|) \quad (3.5)$$

onde: a, b e c são os parâmetros.

O comportamento do sistema de Chua foi analisado através de simulações computacionais. Aplicando o método numérico de Euler (GARCIA, 2000), obtém-se a equação discretizada 3.6.

$$x_{n+1} = x_n + hf(t_n, x_n) \quad (3.6)$$

Dado o estado inic ial do s istema $x[0]$ e aplica ndo um método previsor-corretor obtém-se a seqüência de estados x_{n+1}.

A implementação numérica e com putacional do mé todo de Euler associada ao método previsor-corretor aplicado ao sistema de Chua está descrita no Anexo A. As equações utilizadas na implementação computacional estão descritas a seguir.

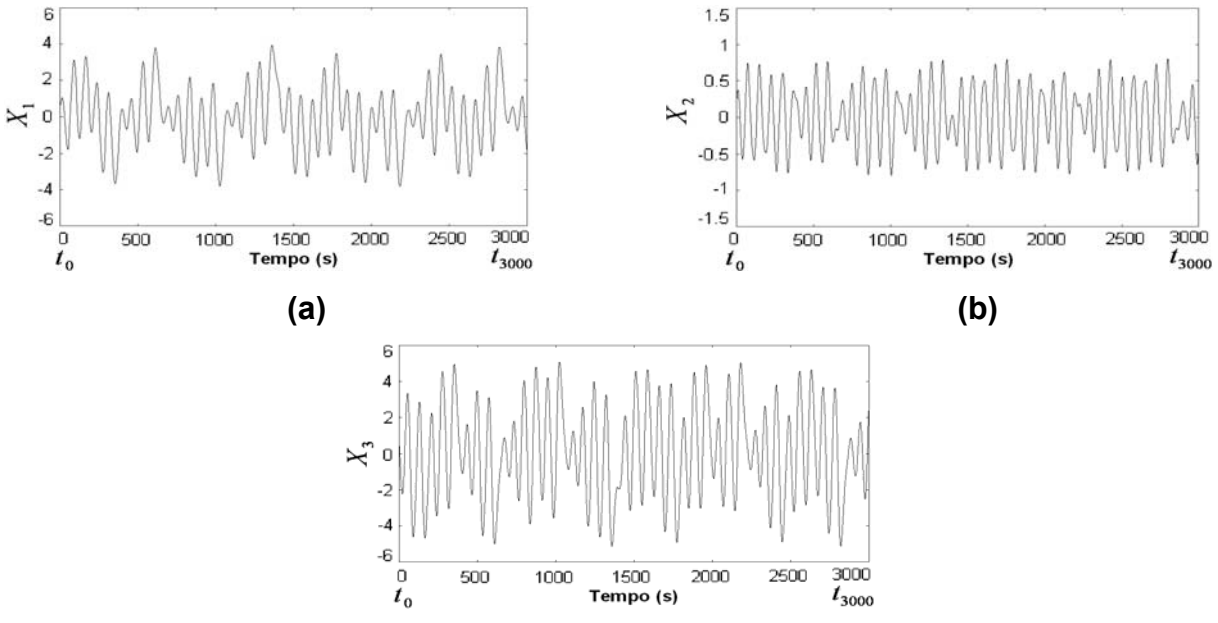
$$x_1 t_{(i+1)} = x_{1(i)} + ((\alpha(x_{2(i)} - f(x_{1(i)})))\Delta T) \quad (3.7)$$

$$x_2 t_{(i+1)} = x_{2(i)} + ((x_{1(i)} - x_{2(i)} + x_{3(i)})\Delta T) \quad (3.8)$$

$$x_3 t_{(i+1)} = x_{3(i)} + ((-\beta x_{2(i)})\Delta T) \quad (3.9)$$

O comportamento do sistema de Chua foi verificado segun do simulações ilustradas nas figuras 3.3 (a-c). A figura 3.3 exibe o comportamento do sistema de Chua através dos gráficos obtidos das variáveis de estado x_1, x_2 e x_3 em cada instante de tempo t_n.

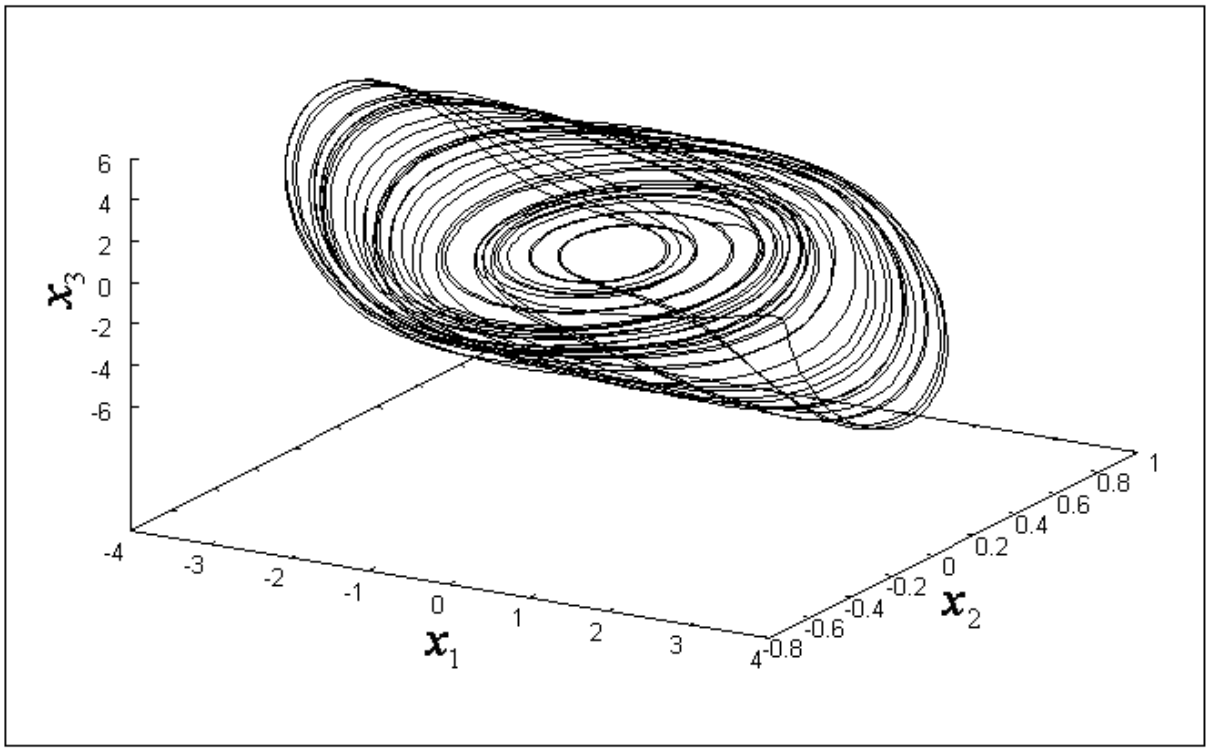
Figura 3.3 – Análise do comportamento do sistema de Chua. (a) Comportamento da variável de estado x_1 usando condiç ão inic ial $x_1[0] = 0{,}6$. (b) Comportamento da va riável de estado x_2 usando condiç ão inic ial $x_2[0] = 0{,}2$. (c) Comportamento da variável de es tado x_3 usando condição inicial $x_3[0] = 0{,}7001$.



Conforme observado, as figuras 3.3 (a-c) exibem respectivamente os comportamentos das variáveis de estado x_1, x_2 e x_3 do sistema de Chua nos instantes de tempo $t \in [t_0; t_{3000}]$. Os parâmetros utilizados nas simulações ($\Delta T = 0{,}034, \quad b = -\frac{1}{7}, \quad a = \frac{2}{7}, \quad \alpha = 9, \quad \beta = 14{,}28$ e $c = 1$) foram escolhidos para a situação de caos do sistema de Chua (NADA, 1993; CHUA; KOMURO; MATSUMOTO, 1986).

O comportamento complexo do sistema de Chua pode ser representado pelo diagrama de fase, obtido pela combinação das variáveis de estado x_1, x_2 e x_3 durante o intervalo de tempo $t \in [t_0; t_{3000}]$. Este diagrama está ilustrado na figura 3.4.

Figura 3.4 – Diagrama de fase do sistema de Chua.



3.2 Dinâmica Caótica Aplicada à Criptografia

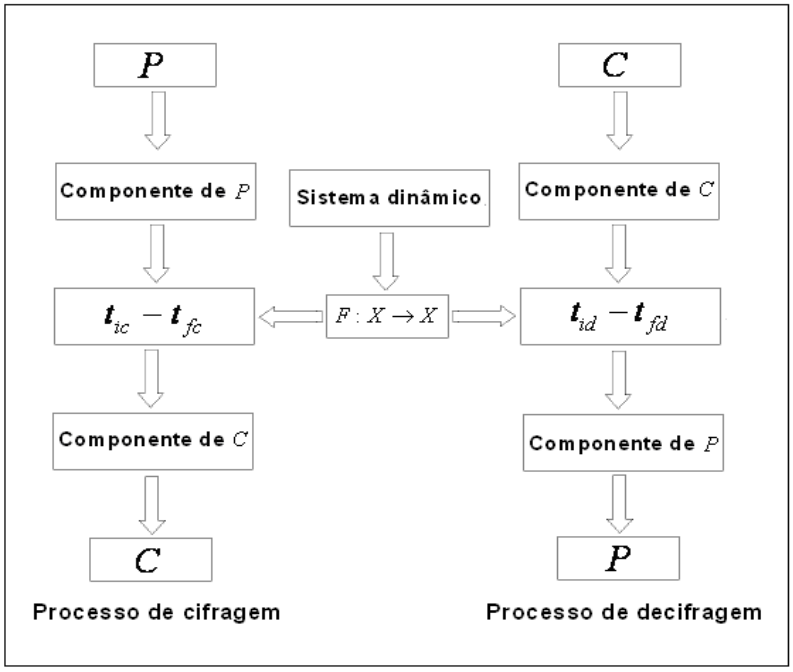
Nos últimos anos, vários pesquisadores têm aplicado o comportamento caótico dos sistemas dinâmicos em criptografia (BAPTISTA, 1998).

Esse interesse deve-se à irregularidade e ao comportamento imprevísível obtido nesses sistemas (KADANOFF, 1983; GLEICK, 1990).

Seja uma informação P que representa a informação original e $F : X \to X$ a trajetória do sistema dinâmico nos instantes de tempo t_n, geralmente desprezando alguns instantes de tempo no início para eliminar qualquer estado transitório inicial. O processo de cifragem consiste em associar cada instante de tempo t_n com um componente da informação P. Os instantes de tempo são obtidos através das órbitas do sistema, gerados a partir do estado inicial (chave) $x[0] \in t_n$, e assim utilizando o intervalo de tempo $t \in [t_{ic}; t_{fc}]$, que correspondem aos instantes inicial e final usados no processo de cifragem.

A decifragem (recuperação) da informação cifrada C é realizada utilizando o mesmo estado inicial (chave) $x[0] \in t_n$, gerando os instantes de tempo t_n obtidos através das órbitas do sistema. Associa-se cada instante t_n com um componente de C, usando assim os instantes $t \in [t_{id}; t_{fd}]$, que correspondem aos instantes inicial e final usados no processo de decifragem (recuperação). A figura 3.5 ilustra através de um diagrama em blocos o modelo criptográfico genérico usando sistemas dinâmicos.

Figura 3.5 – Modelo criptográfico genérico usando sistemas dinâmicos.

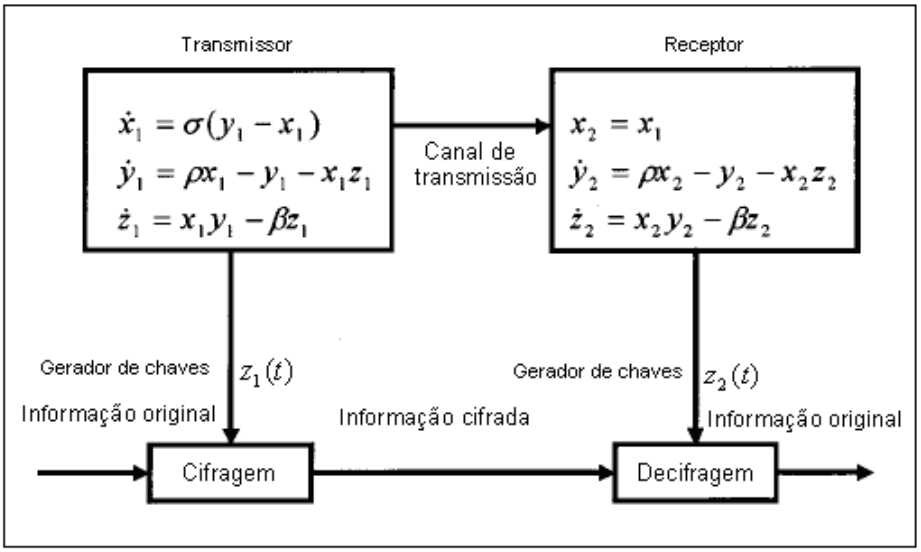


Note-se que, conforme apresentado na figura 3.5, o sistema dinâmico é o principal responsável em mascarar os componentes da informação original, de forma que a segurança do cifrador esteja associada ao comportamento irregular obtido pelo sistema dinâmico.

3.3 O CIFRADOR PROPOSTO POR RONG HE E VAIDYA

No trabalho de Rong He e Vaidya (1998), a informação original é cifrada através de chaves criptográficas geradas pelo comportamento caótico do sistema de Lorentz. A sincronização entre os sistemas transmissor e receptor utiliza a técnica de Pecora e Carroll (PECORA; CARROLL, 1990). O sistema transmissor envia um sinal $x(t)$ para o receptor. A figura 3.6 ilustra o gerador de chaves criptográficas.

Figura 3.6 – Gerador de chaves obtido pelo comportamento do sistema de Lorentz.



Na figura 3.6 é possível observar que quando o sistema escravo sincroniza com o sistema mestre $y_2(t) = y_1(t)$ e $z_2(t) = z_1(t)$. Deste modo, tanto transmissor quanto receptor têm os mesmos sinais aleatórios gerados em cada instante de tempo. Logo, esses sinais podem ser utilizados para gerar as mesmas chaves criptográficas a partir de $y_2(t)$ e $y_1(t)$ ou de $z_2(t)$ e $z_1(t)$.

3.4 Diferenciais do Cifrador Proposto em Relação ao Cifrador de Rong He e Vaidya

Um novo cifrador utilizando contribuições do cifrador apresentado por Rong He e Vaidya (1998) é proposto. As características do cifrador de Rong He e Vaidya (1998) que foram preservadas são: utilização do comportamento caótico de um sistema para gerar chaves criptográficas e utilização de cada chave uma única vez.

Entretanto, os diferenciais do cifrador proposto em relação ao proposto por Rong He e Vaidya (1998) são:

i. Não necessita de sincronismo entre os sistemas transmissor e receptor.
ii. Independência do alfabeto, isto é, utiliza o código ASCII como referência, e com isso eleva significativamente o grau de segurança e adaptabilidade do cifrador.
iii. O comportamento caótico obtido pelo sistema de Chua é usado para cifragem de dados.
iv. Portabilidade para diversos ambientes (desktop, laptop e diferentes canais de transmissões).

No próximo capítulo será apresentado o cifrador proposto, tendo como princípio utilizar as órbitas geradas pelo sistema de Chua apresentado neste capítulo.

4 CIFRADOR PROPOSTO

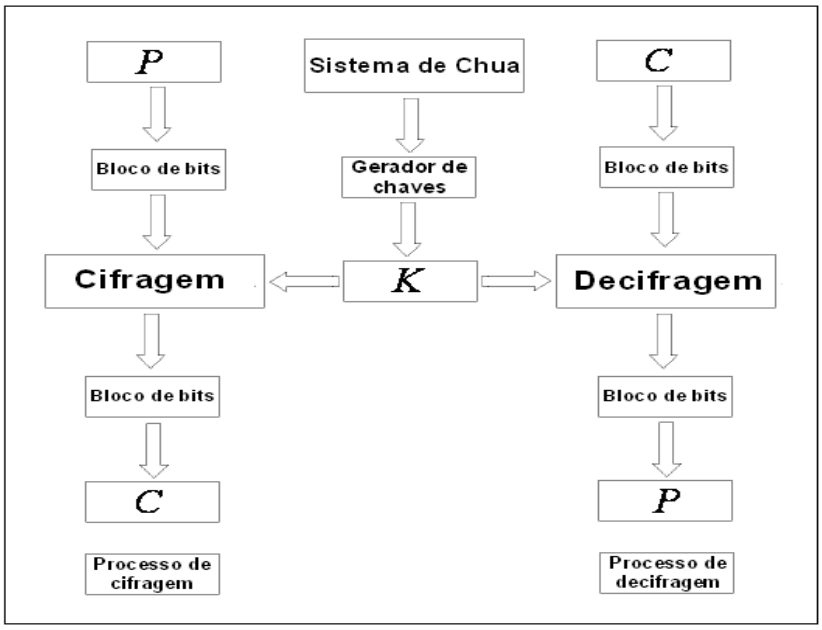
O cifrador proposto utiliza as órbitas geradas pelo sistema de Chua (GUZMAN et al., 2008) para a cifragem de informações (texto, imagem, áudio e vídeo). Uma chave criptográfica é usada na cifragem de cada componente da informação original. Uma característica importante deste cifrador corresponde à utilização de cada chave criptográfica apenas uma vez.

4.1 DEFINIÇÃO DO CIFRADOR PROPOSTO

O princípio do cifrador proposto é proteger uma informação a ser transmitida ou armazenada diante de usuários não autorizados. A segurança do cifrador é obtida através do comportamento gerado pelo sistema de Chua, que é utilizado como gerador de chaves criptográficas.

O diagrama em blocos do cifrador proposto está ilustrado na figura 4.1. Nele, a informação original é chamada de informação P, a informação irreconhecível é chamada de informação cifrada C, e a chave criptográfica é chamada de K. O processo para transformar uma a informação original em uma informação cifrada é chamado de cifragem, e o processo inverso é chamado de decifragem.

Figura 4.1 – Estrutura do cifrador proposto.



O cifrador proposto é classificado co mo cifrador de blo cos, por cifrar blocos com quatro bytes. A informação P é transformada em informação C através da função $C = E(P, K)$. Para realiz ar o processo inverso, utiliza-se a função $P = D(C, K)$. Seguindo esse camin ho, ambos os processos são parametrizados pela simetria das chaves, isto é, a mesma c have é utilizada para as t ransformações das informações P e C.

O desenv olvimento de métodos crip tográficos modernos env olve uma composição de funções matemáticas definidas como operações. Para tais operações serem utilizadas nos processo s de cifragem/decifragem, é necessário possuir características convenientes como velocidade (eficiência), facilidade na troca de chaves e altos índices de difusão e confusão (SHANNON, 1949). As operações do cifrador proposto são defin idas por operações simp les, o que permite maior eficiência no desem penho. Os princípios de difusão e confusão são garantidos através do comportamento complexo obtido pelo gerador de chaves.

O processo de cifragem corresponde na função: $C = E(P, K) = (P_n + K_n) \bmod(256)$. No processo de decifragem realiz a-se o processo inverso, sendo assim, $P = D(C, K) = C_n - (K_n \bmod(256))$, onde: P_n é um componente da informação P, K_n originado da c have simétrica K e C_n é um componente da informação C.

Os principais critérios para ava liação das operações dos cifradores são: segurança, funcionalidade e desempenho (MENEZES; VANSTONE; OORSCHOT, 1996). Com base nesse conceito , novas metodologias de matemática têm sido consideradas e consequentem ente aplic adas a novos métodos de criptografia.

Nesse contexto, os sistemas c aóticos tornam-se uma exc elente alternativa para geradores de chaves em cifradores, pelo fato de elevar a segurança. Dentre as características que destacam a aplicação dos sistemas caóticos em criptografia estão: sensibi lidade do sistema às condições inic iais e comportamento não-periódico.

4.2 OPERAÇÃO CAÓTICA

O cifrador proposto se baseia na utilização dos intervalos de órbitas do sistema de Chua (GUZMAN et al., 2008). Nessa linha, o processo de cifragem de uma informação original consiste em iterar o sistema caótico e associar cada intervalo para um componente da informação P, e, nesse passo, gerar os números de iterações. Assim, o processo de cifragem será representado pelo número de iterações utilizadas.

Por outro lado, o processo de decifragem (recuperação) de uma informação cifrada consiste em iterar o mesmo sistema caótico como no caso do processo de cifragem, e assim a partir da mesma condição inicial para o número de vezes indicado pela cifragem. O intervalo alcançado na cifragem trará a decifragem (recuperação) do componente.

A estrutura do cifrador proposto fundamenta-se na função abaixo.

$$CP = (F, K, A_c, A_d) \tag{4.1}$$

onde: CP corresponde ao cifrador proposto, F representa o sistema de Chua no intervalo $F : X \to X$, K representa a chave criptográfica, A_c é o algoritmo de cifragem e A_d é o algoritmo de decifragem.

O sistema de Chua, já apresentado na seção 3.1, é descrito pelas equações abaixo.

$$\dot{x}_1 = \alpha(x_2 - f(x_1)) \tag{4.2}$$

$$\dot{x}_2 = x_1 - x_2 + x_3 \tag{4.3}$$

$$\dot{x}_3 = -\beta x_2 \tag{4.4}$$

onde: α e β são parâmetros do sistema, x_1, x_2 e x_3 as variáveis de estado e $f(x_1)$ é a função não-linear definida pela equação 4.5.

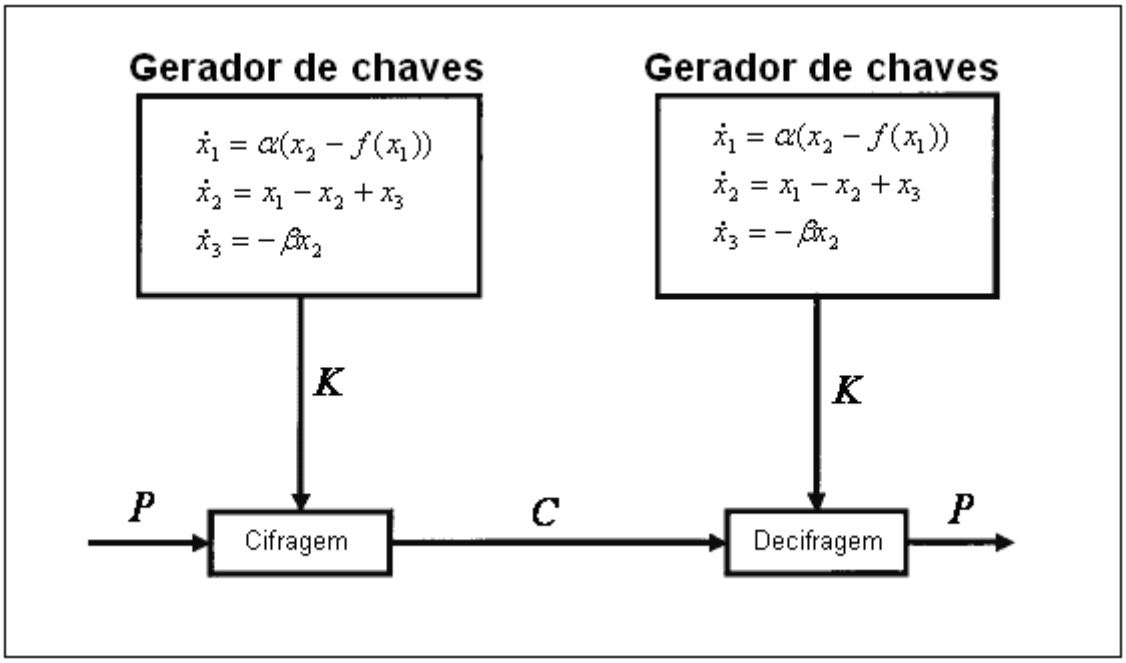
$$f(x_1) = bx_1 + \frac{1}{2}(a-b)(|x_1 + c| - |x_1 - c|) \tag{4.5}$$

onde: a, b e c são os parâmetros.

O sistema é descrito por três equações diferenciais de primeira ordem, compostas por três variáveis de estado x_1, x_2, x_3 e uma função não-linear. Para obter uma situação de caos do sistema de Chua, foram adotados os seguintes valores ($a = \frac{2}{7}$, $b = -\frac{1}{7}$, $c = 1$, $\beta = 14{,}28$ e $\alpha = 9$) para os parâmetros do sistema (NADAN, 1993; CHUA; KOMURO; MATSUMOTO, 1986).

O cifrador proposto utiliza o comportamento de um sistema dinâmico tridimensional, o sistema de Chua (GUZMAN et al., 2008), como gerador de chaves para implementar a segurança do cifrador, como ilustrado na figura 4.2.

Figura 4.2 – Gerador de chaves usando o sistema de Chua.



A figura 4.2 demonstra o gerador de chaves criptográficas.

Para cada componente de P uma chave K_n originada da chave K é usada na cifragem. Na decifragem, cada chave K_n é utilizada sobre um componente de C.

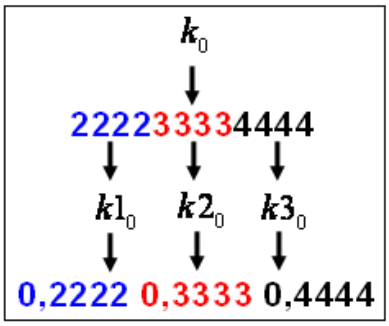
Desse modo, o comportamento caótico é combinado com a informação original para se atingir uma criptografia eficiente, procedimento este que se caracteriza por não preservar os padrões repetitivos da informação original. As

chaves criptográficas geradas são classifi cadas como simétricas, pois as mesmas chaves são usadas tanto no processo de cifragem, quanto na decifragem.

A chave in icial defin ida como k_0 é introduzida pelo usuário. Desta forma, k_0 é constituído de 12 dígitos inteiros no intervalo $0-9$. Logo a seguir, a chave k_0 é dividida respectivamente pelo c ifrador em três grupos com quatro números inteiros, de modo que $k_0 = k1_0, k2_0, k3_0$, onde s ão constit uídos por 4 números inteiros. Em seguida, os valores de $k1_0$ são transformados em valor es decimais, constituído por 4 números decim ais. O mesmo procedimento é aplicado para $k2_0$ e $k3_0$.

A figura 4.3 ilustra o processo de transformação da chave k_0.

Figura 4.3 – Transformação da chave k_0.



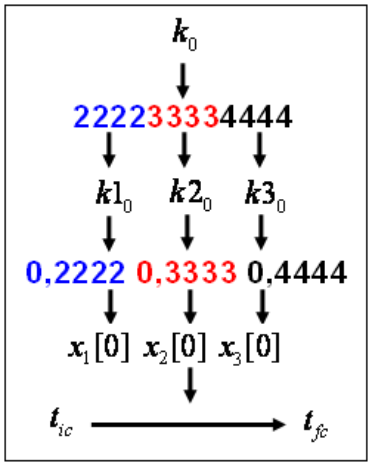
A figura 4.3 exibe a transformação realizada com a chave k_0.

A partir de $k_0 = 12$ dígitos inteiros no intervalo $0-9$, origina-s e $k1_0, k2_0$ e $k3_0$ cada qual c om quatro números inteiros, e logo após são respectivamente transformados em valores decimais , constituídos por 4 números decimais.

Em seguida, os valores de $k1_0, k2_0$ e $k3_0$ são respectivamente usados pelo sistema de Chua c omo condições iniciais das variáveis de estado $x_1[0], x_2[0]$ e $x_3[0]$ iniciando a partir do instante de tempo t_0. As órbitas do sistem a são obtidas através dos valores gerados nas iterações das variáveis de estado x_1, x_2 e x_3.

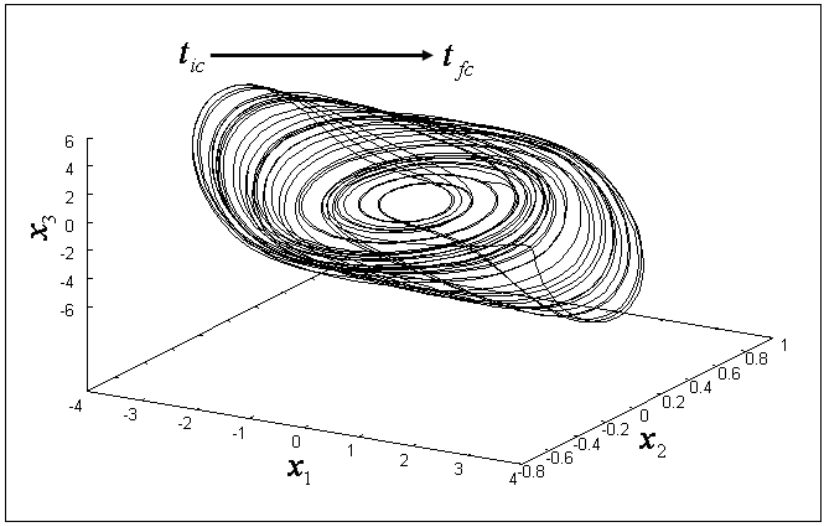
Nessa linha de raciocínio, cada iteração entre os instantes de tempo $t \in [t_{ic}; t_{fc}]$, das órbitas do sistema é combinada com um componente de P, onde t_{ic} e t_{fc} correspondem respectivamente aos instantes de tempo inicial e final usado no processo de cifragem, como ilustra a figura 4.4.

Figura 4.4 – Os instantes de tempo t_n representados pelo intervalo de tempo $t \in [t_{ic}; t_{fc}]$ gerados pelo sistema de Chua.



O comportamento complexo obtido pela combinação das variáveis de estado x_1, x_2 e x_3 no intervalo de tempo $t \in [t_{ic}; t_{fc}]$ é apresentado através da simulação do diagrama de fase gerado pelo sistema de Chua, como ilustrado na figura 4.5. Esta análise foi demonstrada utilizando como condições iniciais: $x_1[0] = 0{,}6$, $x_2[0] = 0{,}2$ e $x_3[0] = 0{,}7001$ e parâmetros: $\Delta T = 0{,}034$, $b = -\frac{1}{7}$, $a = \frac{2}{7}$, $\alpha = 9$, $\beta = 14{,}28$ e $c = 1$, que são valores do sistema que tendem a situação de caos (NADAN, 1993; CHUA; KOMURO; MATSUMOTO, 1986).

Figura 4.5 – Diagrama de fase obtido pelo sistema de Chua.



Conforme retratado na figura 4.4, cada chave K_n originada da chave K_0 é gerada a partir da iteração das variáveis de estado x_1, x_2 e x_3, onde as chaves obtidas para a cifragem são representadas pelos instantes de tempo $t \in [t_{ic}; t_{fc}]$. A chave K_n é gerada a partir do instante de tempo t_n com a adição dos valores obtidos pelas variáveis de estado x_1, x_2 e x_3. Uma representação da chave é expressa pela função 4.6.

$$K_n(t_n, x_1 + x_2 + x_3). \tag{4.6}$$

Um exemplo de geração de chaves criptográficas a partir do intervalo de tempo $t \in [t_{ic}; t_{fc}]$ é mostrada na tabela a seguir.

Tabela 4.1 – Geração de chaves criptográficas através do intervalo $t \in [t_{ic}; t_{fc}]$.

t_n	$x_1 = 0{,}610429 \; + \; x_2 = 0{,}231888 \; + \; x_3 = 0{,}609670,$ onde $k_n = 1451987$
t_{n+1}	$x_1 = 0{,}615602 \; + \; x_2 = 0{,}241770 \; + \; x_3 = 0{,}576557,$ onde $k_{n+1} = 1433929$

4.3 MODELO CRIPTOGRÁFICO DO CIFRADOR PROPOSTO

O cifrador proposto é composto pelos algoritmos A_c e A_d.

No algoritmo de cifragem, para cada componente de P é usada uma chave K_n originada de K. Dessa forma, A_c realiza o processo $C_n = E(P_n, K_n(F^n(t_n, x_1, x_2, x_3)))$ e o método de cifragem é descrito pela função: $C_n = (P_n + K_n) \bmod 256$.

A tabela 4.2 exibe o método de cifragem aplicado sobre cinco componentes da P.

Tabela 4.2 – Processo de cifragem.

Tempo	$x[t]$	Chave	Texto original	Texto cifrado
t	$x_1[t]+x_2[t]+x_3[t]$	K	P	$C=(P+K) \bmod 256$
0.1	1.451987	1451987	C (67)	22
0.2	1.433929	1433929	H (72)	145
0.3	1.414838	1414838	A (65)	247
0.4	1.394711	1394711	O (79)	102
0.5	1.373548	1373548	S (83)	191

Com base na tabela 4.2, será fornecido um exemplo do processo de cifragem aplicado sobre componentes de P. Note-se que esse processo está intimamente ligado ao comportamento irregular obtido pela função 4.6, e assim associada com o método de cifragem ilustrado na tabela 4.2.

$P_n = C = 67(ASCIIDecimal)$, $K_n = 1451987$, $C_n = (67 + 1451987) \bmod 256$, $C_n = 22(ASCIIDecimal)$.

$P_{n+1} = H = 72(ASCIIDecimal)$, $K_{n+1} = 1433929$, $C_{n+1} = (72 + 1433929) \bmod 256$, $C_{n+1} = 145(ASCIIDecimal)$.

No algoritmo de decifragem, para cada componente de C uma chave K_n é utilizada. Desse modo, A_d realiza o processo

$P_n = D(C_n, K_n(F^n(t_n, x_1, x_2, x_3)))$, e o método de decifragem é descrito pela função : $P_n = C_n - (K_n \bmod 256)$.

A tabela 4.3 exibe o processo de decifragem aplicad o sobre cinco componentes de C. Nesse processo, é necessário gerar a mesma seqüência de chaves criptográficas da cifragem obtidas a partir das iterações do sistema de Chua. Após a transformação do componente C_n em P_n é verificado se o resultado de P_n $(ASCIIDecimal)$ é menor que 0, se condição verdadeira, então $P_n + 256$.

Tabela 4.3 – Processo de decifragem.

Tempo t	$x[t]$ $x_1[t]+x_2[t]+x_3[t]$	Chave K	Texto cifrado C	Texto original $P = C - (K \bmod 256)$ Se $P < 0$, então $P + 256$
0.1	1.451987	1451987	22	C (67)
0.2	1.433929	1433929	145	H (72)
0.3	1.414838	1414838	247	A (65)
0.4	1.394711	1394711	102	O (79)
0.5	1.373548	1373548	191	S (83)

Observando a tabela 4.3 é possível verificar a aplicação do processo de decifragem sobre component es de C. Para este processo é necessário gerar a mesma seqüência (órbitas) de iterações da cifragem obtida pela função 4.6, e assim associado com o método de decifragem, conforme ilustrado na tabela 4.3.

$C_n = 22(ASCIIDecimal)$, $K_n = 1451987$, $P_n = 22 - (1451987(\bmod 256))$, $P_n = 67(ASCIIDecimal) = "C"$.

$C_{n+1} = 145(ASCIIDecimal)$, $K_{n+1} = 1433929$, $P_{n+1} = 145 - (1433929(\bmod 256))$, $P_{n+1} = 72(ASCIIDecimal) = "H"$. Se result ado de $P_n < 0$, então utiliza -se a condiçã o $P_n + 256$.

4.4 CARACTERÍSTICAS

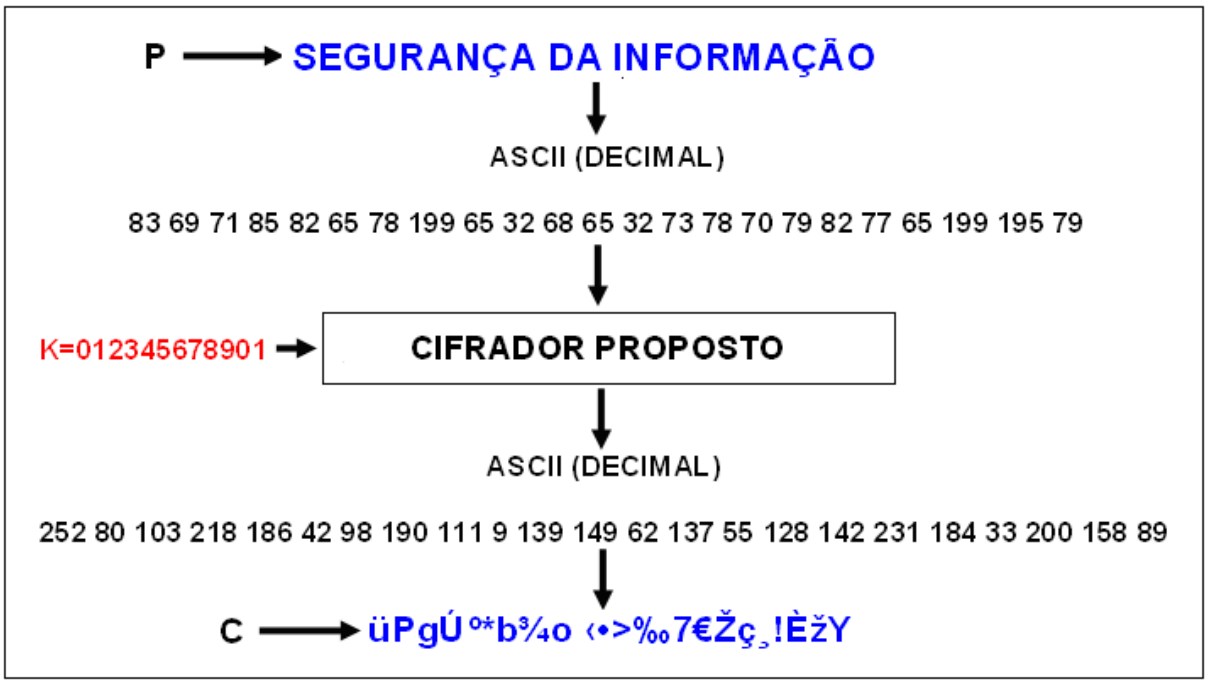
Conforme já mencionado no início deste capítulo, as operações do cifrador proposto são definidas por operações simples, o que permite maior eficiência no desempenho. Os princípios de difusão e confusão (SHANNON, 1949) são garantidos através do comportamento complexo obtido pelo gerador de chaves, isto é, a operação caótica. Os testes experimentais obtidos no capítulo 5 são gerados pelas seguintes operações:

- ✓ Operação caótica
- ✓ Método de cifragem/decifragem

Assim, a operação caótica que é descrita pela função 4.6 e associada com os métodos de cifragem e decifragem e, no sentido do exposto, referidas operações podem ser sumarizadas nas seguintes funções: $C_n = (P_n + K_n) \bmod 256$ e $P_n = C_n - (K_n \bmod 256)$.

A figura 4.6 ilustra o processo de cifragem proposto, aplicado sobre componentes de P.

Figura 4.6 – Procedimento de cifragem.



A figura 4.6 ilustra a aplicação do cifrador proposto sobre uma informação do tipo texto. Assim, é demonstrada a complexidade do comportamento obtido pela combinação da operação caótica e o método de cifragem. Além disso o cifrador proposto acrescenta duas operações que foram implementadas em Anexo B, e assim descritas por:

- ✓ Troca de posições dos bytes
- ✓ Modo CBC (Cipher Block Chaining)

As operações troca de posições dos bytes (transposição) e modo CBC (Cipher Block Chaining) foram implementadas visando otimizar a difusão e confusão (SHANNON, 1949) entre os dados.

O cifrador proposto é usado na cifragem de blocos com quatro bytes, isto é, (b_1, b_2, b_3, b_4). Diante disso, a operação troca de posições dos bytes segue a lógica da transposição. A lógica utilizada no cifrador consiste em trocar as posições dos bytes no bloco, isto é, a posição 1 com a 2 e a posição 3 com a 4.

Seguindo essa sistemática de racicínio. (b_1, b_2, b_3, b_4) corresponde a (b_2, b_1, b_4, b_3).

A seguir é descrita pelas funções a sequência da operação troca de posições dos bytes no cifrador:

$$C_n = TP \rightarrow E(P_n, K_n(F^n(t_n, x_1, x_2, x_3))) \text{ (cifragem), (4.7)}$$

$$P_n = D(C_n, K_n(F^n(t_n, x_1, x_2, x_3))) \rightarrow TP \text{ (decifragem), (4.8)}$$

onde

- ✓ TP corresponde a operação troca de posições dos bytes

A lógica aplicada no cifrador pela operação modo CBC, consiste em realizar uma operação XOR entre o bloco de bits original e o bloco anterior cifrado.

Para isso, seguem as seguintes funções:

(Cifragem)

$$E_1 = VI \oplus B^1(P^1(E)) = B^1(C^1),\ E_2 = B^1(C^1) \oplus B^2(P^2(E)) = B^2(C^2),...,E_n \quad (4.9)$$

(Decifragem)

$$D_1 = B^1(C^1(D)) \oplus VI = B^1(P^1),\ D_2 = B^2(C^2(D)) \oplus B^1(C^1) = B^2(P^2),...,D_n \quad (4.10)$$

A seguir é descrito pelas funções a sequência da operação modo CBC no cifrador:

$$C = CBC \rightarrow E(P_n, K_n(F^n(t_n, x_1, x_2, x_3))) \text{ (cifragem), (4.11)}$$

$$P = D(C_n, K_n(F^n(t_n, x_1, x_2, x_3))) \rightarrow CBC \text{ (decifragem), (4.12)}$$

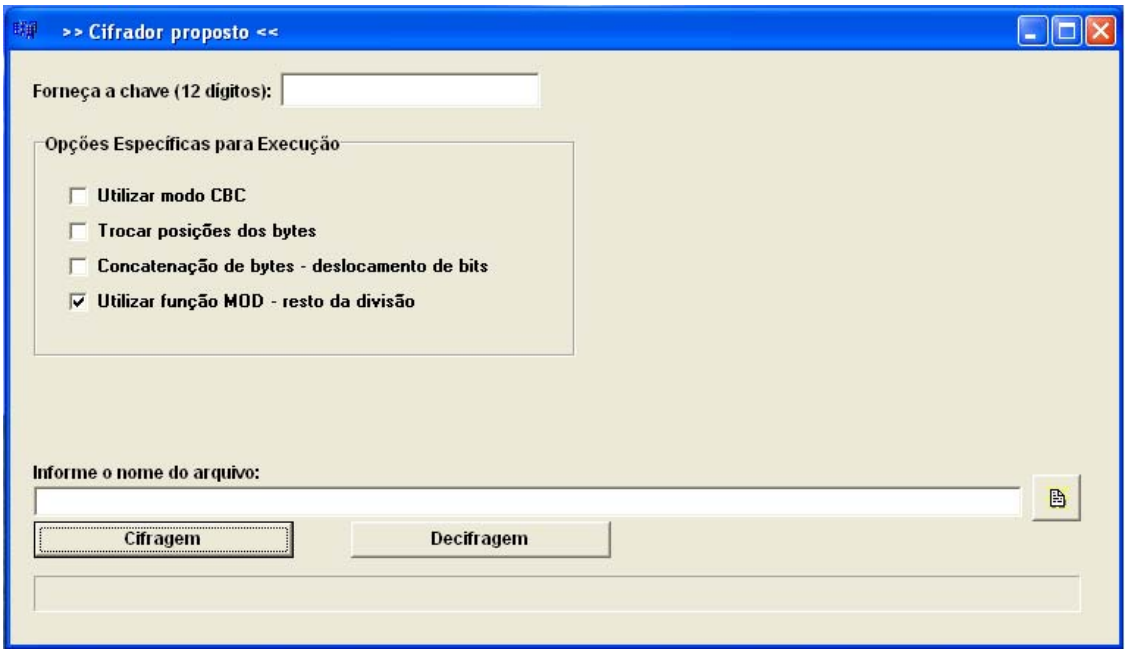
4.5 Resumo das Operações do Cifrador

As operações do cifrador correspondem a quatro operações, como assim seguem:

- ✓ Modo CBC (Cipher Block Chaining)
- ✓ Troca de posições dos bytes
- ✓ Operação caótica
- ✓ Métodos de cifragem/decifragem

A figura 4.7 ilustra as operações do cifrador em sua interface implementada em Anexo B, na linguagem de programação C++.

Figura 4.7 – Interface do cifrador.



A operação modo CBC (Cipher Block Chaining) realimenta a cifragem do bloco de bits atual com o resultado da cifragem dos blocos de bits anteriores, realizando a operação XOR entre bloco atual e o bloco anterior cifrado (STALLINGS, 2002).

A operação troca de posições dos bytes rotaciona as posições dos bytes no bloco. A operação caótica (chave) é associada com o método de cifragem para garantir a difusão e confusão entre os dados. A sequência das operações no cifrador é descrita pela função abaixo:

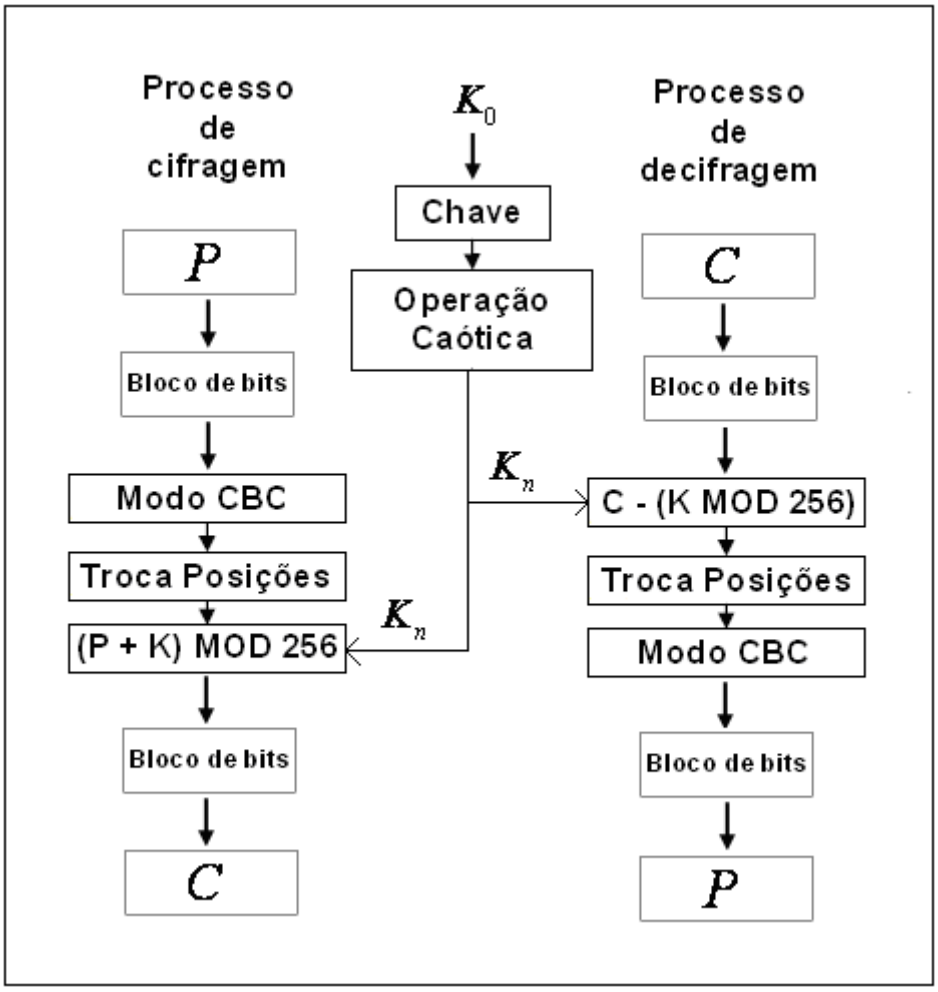
$$C = CBC \rightarrow TP \rightarrow E(P_n, K_n(F^n(t_n, x_1, x_2, x_3))) \text{ (cifragem)}, \tag{4.13}$$

Na decifragem, a operação caótica (chave) é associada com o método de decifragem. Assim, os valores resultantes são utilizados na operação troca de posições dos bytes, e finalizando com a operação modo CBC (Cipher Block Chaining). A sequência das operações no cifrador é descrita pela função abaixo:

$$P = D(C_n, K_n(F^n(t_n, x_1, x_2, x_3))) \rightarrow TP \rightarrow CBC \text{ (decifragem)}, \tag{4.14}$$

A figura 4.8 ilustra respectiv amente as operações do c ifrador proposto, tanto no processo de cifragem quanto na decifragem.

Figura 4.8 – Operações do cifrador.



A figura 4.8 demonstra as operações do cifrador.

Entretanto, a relevânc ia do trabal ho fundam enta-se em investigar a aplicação do sistema de Chua (operação caótica) na criptografia de dados.

Para tanto, os experimentos realiz ados n o capítulo 5 associam a operação caótica (chave) com os métodos de cifragem e decif ragem. Assim, as operações modo CBC e troca de posições dos bytes não são investigadas nas análises do cifrador proposto.

4.6 SEGURANÇA DO CIFRADOR

Nesta seção são descritas as características que garantem a robustez do cifrador proposto.

- ✓ Uma chave criptográfica é dita fraca para um cifrador, se todas as sub-chaves forem iguais. Com isso, o efeito para cifragem ou decifragem de uma informação é o mesmo, pois a sequência de chave K_1 a K_n é idêntica à sequência de K_n a K_1. Diante disso, a segurança do cifrador proposto é incrementada com base no comportamento complexo obtido pelo gerador de chaves criptográficas e utilizar cada chave apenas uma vez no cifrador.
- ✓ A operação caótica (chave) permite regular diretamente a sensibilidade do cifrador proposto. Desse modo, quanto maior a sensibilidade do sistema, maior a incerteza gerada. Portanto, maior a sua segurança.

4.7 LIMITAÇÕES DO CIFRADOR PROPOSTO

O cifrador proposto acrescenta inúmeras características que garantem a sua robustez. Entretanto, como todos os cifradores, existem algumas limitações.

Dessa maneira, há limites para os parâmetros de controle destinados a garantir que a órbita caótica seja suficientemente caótica, o que limita o intervalo que os valores dos parâmetros podem tomar. Os métodos de cifragem e decifragem utilizam aproximação para o número inteiro mais próximo, portanto os métodos estão limitados a operar com informações discretas por números inteiros. O desempenho na cifragem e decifragem pode ser melhorado visando a um futuro comercial. No entanto, o cifrador ainda está sob estudo, e isso implica dizer que, com o transcorrer do tempo, poderá ser desenvolvido a ponto de superar as limitações detectadas.

No próximo capítulo serão demonstrados os testes experimentais que verificam o desempenho e robustez do cifrador proposto.

5 RESULTADOS

Os resultados da investigação sobre a utilização do sistema de Chua na criptografia de dados são apresentados nes te capítulo. Alguns testes experimentais foram selecionados par a verificar o desempenho e robustez do cifrador proposto conforme segue a tabela abaixo.

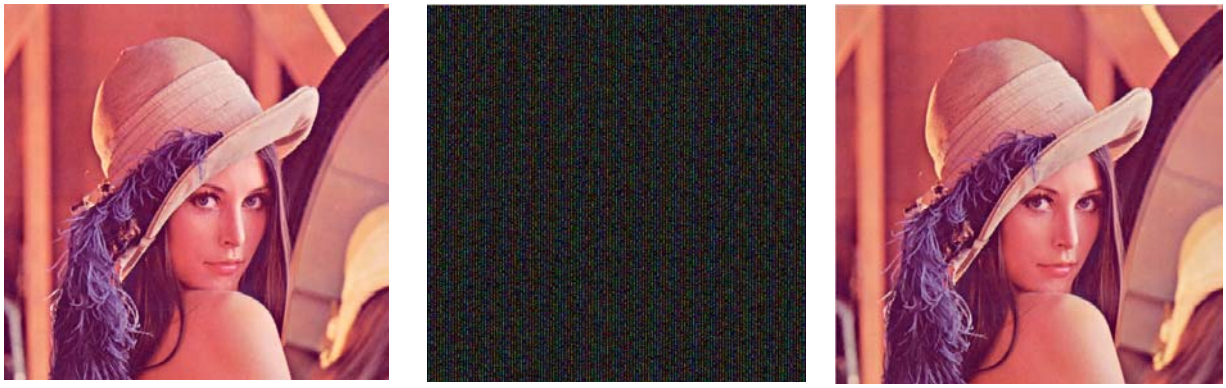
Tabela 5.1 – Testes realizados para avaliação do cifrador proposto.

Seção	Análise	Tipo da informação	Quantidade
5.1 Robustez		Imagem	1
5.2 Robustez		Imagem	1
5.3 Robustez		Texto	1
5.4 Robustez		Texto	1
5.5 Robustez		Texto	1
5.6 Robustez		Texto	1
5.7	Desempenho	Áudio e Vídeo	8

5.1 APLICAÇÃO DO MÉTODO CRIPTOGRÁFICO PROPOSTO SOBRE INFORMAÇÃO DO TIPO IMAGEM

O cifrador proposto apresentado na seção 4.3 foi aplic ado s obre uma informação do tipo imagem. A figura 5. 1 (b) exibe de maneira qualitativa as diferenças estatísticas em relação a figura 5. 1 (a). É possível notar na figura 5.1 (b) que a infor mação cifrada não m ostra vestígios visuais da informação original. Além disso, a figura 5.1 (c) demonstra qualidade no processo de recuperação a partir da informação cifrada. Diante destes resultado s, observa-se que o cifrador proposto pode ser aplicado sobre informação do tipo imagem.

Figura 5.1 – Processo criptográfico. (a) Imagem original (512 x 512 pixels) (COSTA, 2005). (b) Imagem cifrada (512 x 512 pixels) usando condições iniciais $x_1[0] = 0{,}1 \quad x_2[0] = 0{,}2 \quad x_3[0] = 0{,}3$. (c) Imagem decifrada (recuperada) (512 x 512 pixels).



(a) **(b)** **(c)**

A robustez do cifrador proposto foi analisado através de observações qualitativas dos testes experimentais realizados acima. Entretanto, para avaliar a eficiência do método criptográfico proposto é necessário avaliação quantitativa. Um mecanismo utilizado no meio técnico e científico é a função *hash* (LEE, 2007). Desta maneira, uma das alternativas para comparar a integridade das informações é através das funções *hash.*

5.2 ANÁLISE DE INTEGRIDADE ATRAVÉS DE FUNÇÕES *HASH*

O objetivo deste experimento é comparar as integridades das informações original, sua versão cifrada e decifrada (recuperada) apresentada na figura 5.1. Para isso, serão utilizadas as funções *hash MD5* e *SHA1*.

Tabela 5.2 – Análise de integridade. (a) Informação original do tipo imagem (512 x 512 pixels). (b) Informação cifrada do tipo imagem (512 x 512 pixels). (c) Informação decif rada (recuperada) do tipo imagem (512 x 512 pixels).

Tamanho	524354 bytes (Informação original)
MD5	7B6E9AB05CEBAB70B794AEE8E6D4D73A
SHA1	54704D0D05A753E6BA7EDDE64A6CFC40953596FA

(a)

Tamanho	524354 bytes (Informação cifrada)
MD5	9611BFB17E962DDD28FAB5B04A199604
SHA1	9E93C34E3CD584225764FF212C795B37368561FE

(b)

Tamanho	524354 bytes (Informação decifrada (recuperada))
MD5	7B6E9AB05CEBAB70B794AEE8E6D4D73A
SHA1	54704D0D05A753E6BA7EDDE64A6CFC40953596FA

(c)

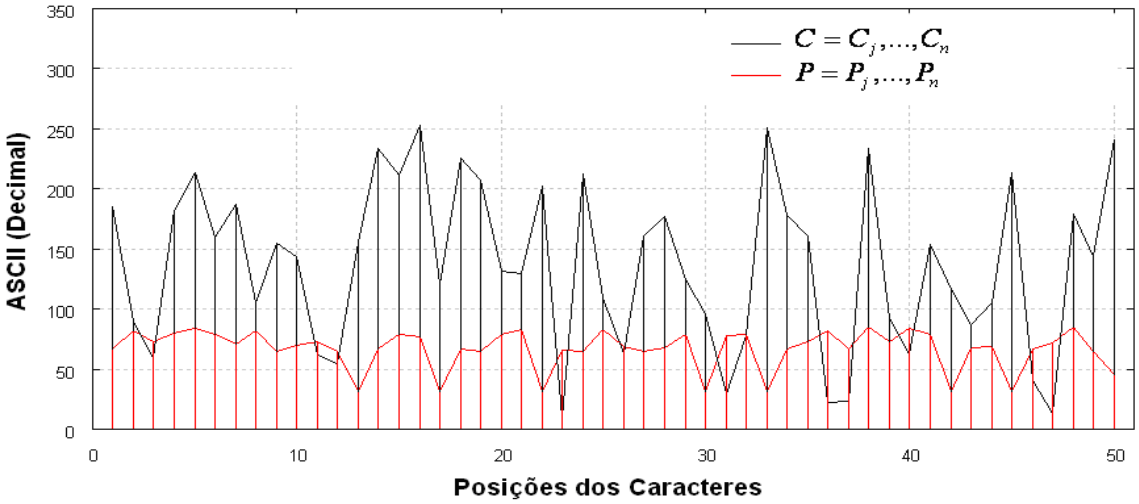
Como obs ervado na tabela 5.2, as representações geradas pelas funções *hash MD5* e *SHA1* sobre a informação original e cifrada indic am valores totalmente diferentes. Por outro lado, os valores gerados pelas funções *hash* sobre as informações original e dec ifrada (recuper ada) da tabela 5.2 são iguais, is to é, a informação decifrada (recuperada) é idêntica a infor mação original. Portanto, este experimento comprova a eficiência do cifrador no proc esso de decifragem (recuperação) dos dados.

5.3 A PLICAÇÃO DO MÉTODO CRIPTOGRÁFICO PROPOSTO SO BRE INFORMAÇÃO DO TIPO TEXTO

Esta seção analisa o grau de difus ão entre os dados, isto é , as diferenças estatísticas entre as informaç ões do tipo texto original e sua versão cifrada gerada pelo cifrador proposto. A informação original corresponde a 50 componentes de P representados respectivamente pelos seguinte s componentes: CRIPTOGRAFIA COM CAOS BASEADO NO CIRCUITO DE CHUA. A versão cifrada

também será composta por 50 componentes de C. Na figura 5.2 é ilustrado para cada posição a representação ASCII do componente P e do componente C.

Figura 5.2 – Método criptográfico aplicado sobre uma informação do tipo texto.



O comportamento irregular obtido pelo sistema de Chua e combinado com a informação original dissipa os padrões repetitivos da informação, aumentando a difusão entre os dados. Conforme ilustrado na figura 5.2 não ocorrem repetições entre os componentes de P e C para cada posição. A tabela 5.3 mostra os padrões não repetitivos das primeiras doze posições dos componentes de P e C ilustrados na figura 5.2. Portanto, os padrões obtidos neste experimento demonstram a eficiência na difusão entre os dados através do sistema de Chua aplicado na criptografia de dados.

Tabela 5.3 – Os padrões de cifragem obtidos pelo cifrador proposto.

ASCII (DECIMAL)		
	P	**C**
1	67 (C)	185
2	82 (R)	90
3	73 (I)	60
4	80 (P)	182
5	84 (T)	213
6	79 (O)	160
7	71 (G)	187
8	82 (R)	105
9	65 (A)	155
10	70 (F)	144
11	73 (I)	62
12	65 (A)	55

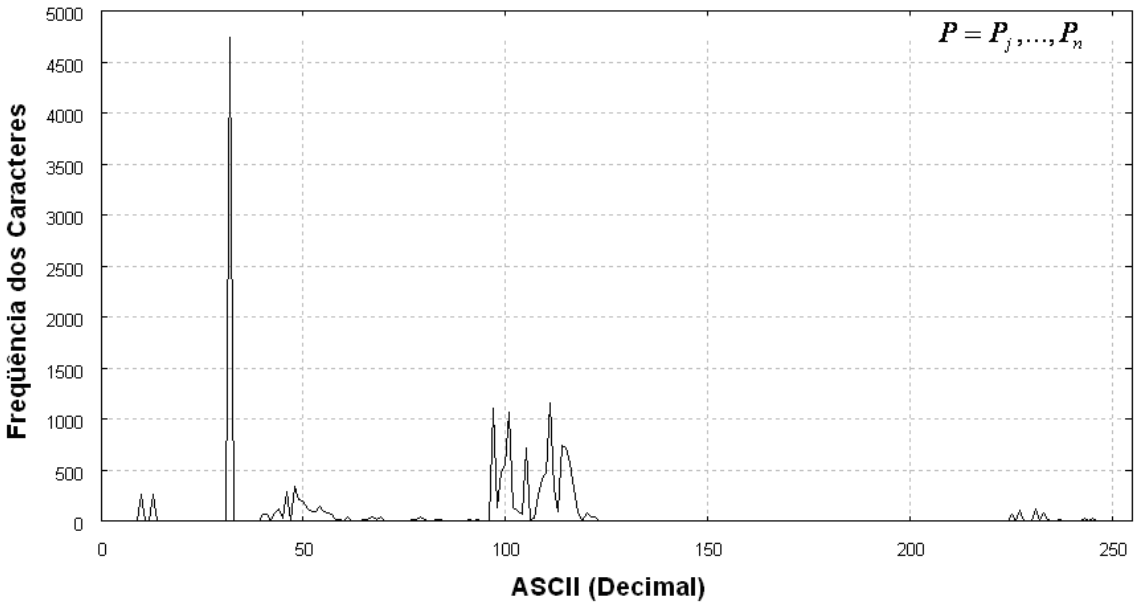
Uma técnica muito utilizada para analisar a robustez de cifrador es contra tentativas de acessos não autorizados é a análise de frequência (BRAGA, 2009). Esta técnica será apresentada na seção a seguir.

5.4 Análise de Segurança do Cifrador Através da Técnica de Análise de Frequência

A análise de frequência é uma das principais técnicas de criptoanálise, assim utilizada para decifrar (recuperar) a informação original a partir da versão cifrada. Esta técnica consiste em analisar a frequência de cada caracter da informação cifrada e comparar estas frequências com aquelas que normalmente aparecem em um determinado idioma (BRAGA, 2009).

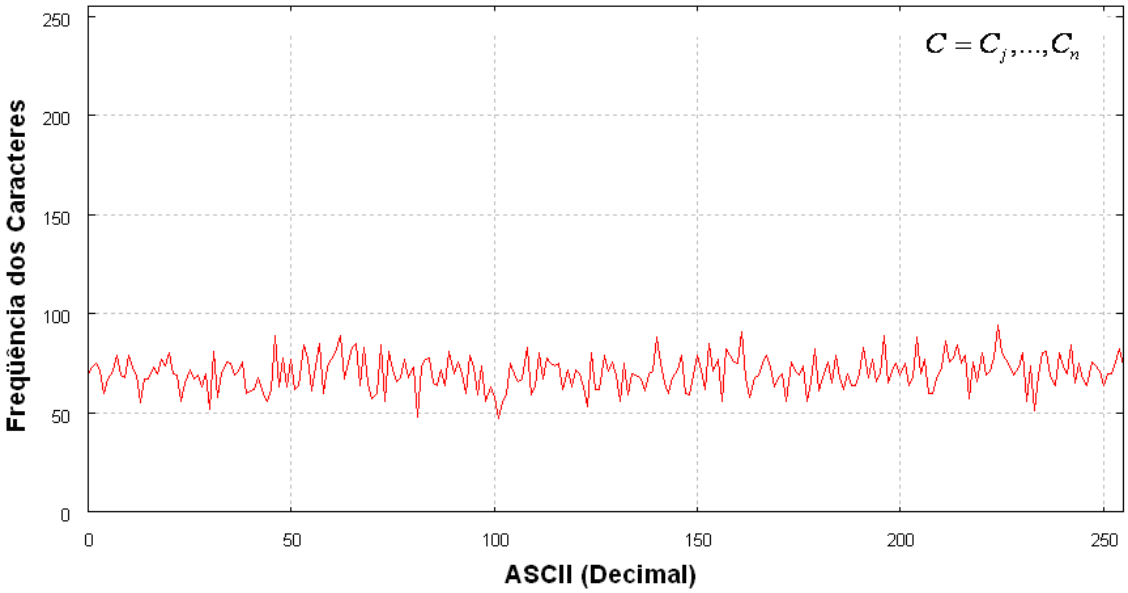
Este experimento verifica a segurança do cifrador proposto através da análise de frequência sobre uma informação do tipo texto. A informação P é composta por 18.016 componentes. A chave K (operação caótica) inicia a partir de $x_1[0] = 0{,}2$, $x_2[0] = 0{,}3$ e $x_3[0] = 0{,}1$. Na figura 5.3 está ilustrada a frequência dos componentes da informação P.

Figura 5.3 – Frequência dos caracteres originais representados pelos componentes de P.



A frequência dos caracteres da informação cifrada C está ilustrada na figura 5.4.

Figura 5.4 – Frequência dos caracteres cifrados representados pelos componentes de C.



Observando a figura 5.3 nota-se que a maior frequência para um determinado caracter é 4742 e que a menor frequência para outro caracter é 0.

Na hipótese, a diferença entre a maior e menor frequência na informação original é 4742. Já no caso da figura 5.4, a maior frequência de um caracter é 94 e a menor frequência é 47, ou seja, uma diferença de 47. Quanto mais uniforme a frequência dos componentes de C menos eficiente torna-se a análise de frequência. Dessa forma, a uniformidade alcançada na frequência dos componentes de C deve-se à eficiência da irregularidade do comportamento obtido pela operação caótica.

Outra maneira de avaliar a robustez do cifrador proposto é calcular a entropia de SHANNON (1949). Este conceito será analisado e aplicado na próxima seção.

5.5 ENTROPIA DE SHANNON

Segundo o conceito de entropia definido por Shannon (1949), a quantidade de informação pode ser associada com a incerteza de obtenção dessa informação. Mediante aplicação dessa idéia, conclui-se que a entropia está intimamente ligada ao grau de desorganização existente na fonte de informação, e quanto maior a desorganização, maior o potencial de informação dessa fonte (SHANNON, 1949; ALVAREZ et al., 2003).

Desta forma, conforme Shannon (1949) demonstrou, é possível medir a entropia para dados binários, e assim através da equação 5.1 é possível calcular a taxa de desorganização dos bits em um canal de dados. A entropia de um sistema utilizando um alfabeto com 256 simbolos é calculada segundo a formulação matemática abaixo.

$$S = -\sum_{i=0}^{256} p_i \log_2(p_i), \tag{5.1}$$

onde: p_i corresponde a freqüência dos $i-ésimo$ caracter ASCII.

Seguindo este princípio, quando $S[p] = 0$ tem-se uma certeza total na obtenção do resultado de medida, isto é, com grande previsibilidade. Entretanto, quando para uma dada distribuição $S[p] > 0$ o resultado não é totalmente previsível (SHANNON, 1949; ALVAREZ et al., 2003).

A entropia da informação cifrada apresentada na seção 5.4 e ilustrada na figura 5.4 corresponde a $S = 7{,}989579685386364$. Portanto, os resultados obtidos indicam um grau significativo de desorganização existente na informação cifrada gerada pelo cifrador proposto.

5.6 ANÁLISE DE SENSIBILIDADE

Uma das principais características dos sistemas caóticos é a sensibilidade em relação às condições iniciais (KADANOFF, 1983; GLEICK, 1990). Na criptografia, quanto maior a difusão entre os dados com a mínima mudança da chave K_0 melhor a eficiência na segurança do cifrador contra ataques de força

bruta. Como já mencionado, a operação caótica utiliza-se do comportamento obtido pelo sistema de Chua. Diante disso, este experimento analisa a sensibilidade da operação caótica (chave) em relação as condições iniciais, e assim combinada com a informação original.

Para tanto, tal análise é realizada através da variação amostral sobre quatro versões cifradas originadas de uma informação. Para isso, o cifrador proposto é aplicado sobre uma informação original, o livro eletrônico “Os Maias” com tamanho de 2,05 MB. O experimento realiza-se alterando minimamente e de maneira isolada as condições iniciais das variáveis de estado x_1, x_2 e x_3 entre as versões cifradas C_1, C_2, C_3 e C_4. A tabela 5.4 exibe os caracteres com maiores ocorrências no idioma Português e suas ocorrências entre as versões cifradas.

Tabela 5.4 – Variação amostral das versões cifradas. (a) Condições iniciais de $C_1 = x_1[0] = 0{,}1 \quad x_2[0] = 0{,}2 \quad x_3 = 0{,}3$. (b) Condições iniciais de $C_2 = x_1[0] = 0{,}11 \quad x_2[0] = 0{,}2 \quad x_3 = 0{,}3$. (c) Condições iniciais de $C_3 = x_1[0] = 0{,}1 \quad x_2[0] = 0{,}21 \quad x_3 = 0{,}3$. (d) Condições iniciais de $C_4 = x_1[0] = 0{,}1 \quad x_2[0] = 0{,}2 \quad x_3 = 0{,}31$.

ASCII (Decimal)	C_1	C_2	C_3	C_4
A (65)	12205	12360	12112	12258
E (69)	11974	12187	12324	12138
O (79)	12835	12842	12735	12956
S (83)	12964	13061	12987	12916
I (73)	12616	12455	12592	12361
R (82)	12017	12230	12176	12200

As distribuições entre os caracteres das versões cifradas demonstram diferenças. No entanto, o grau das variações será quantificado pelas medidas obtidas através do desvio padrão, conforme apresentado na tabela 5.5.

Tabela 5.5 – Análise quantitativa da variação amostral das versões cifradas.

ASCII (Decimal)	Desvio Padrão
A (65)	103,56
E (69)	144,49
O (79)	90,36
S (83)	60,40
I (73)	119,89
R (82)	95,10

Conforme apresentado na tabela 5.5, o desvio padrão dos caracteres das versões cifradas C_1, C_2, C_3 e C_4 indica o valor médio da dispersão dos dados, isto é, interpreta aproximadamente o que podemos esperar nas ocorrências dos caracteres.

Note que os valores obtidos indicam variações significativas no grau de dispersão dos dados entre as versões cifradas. Dessa forma, comprova-se a eficiência da sensibilidade da operação caótica, e assim sendo, de grande importância para a rotustez do cifrador proposto.

5.7 Análise de Desempenho

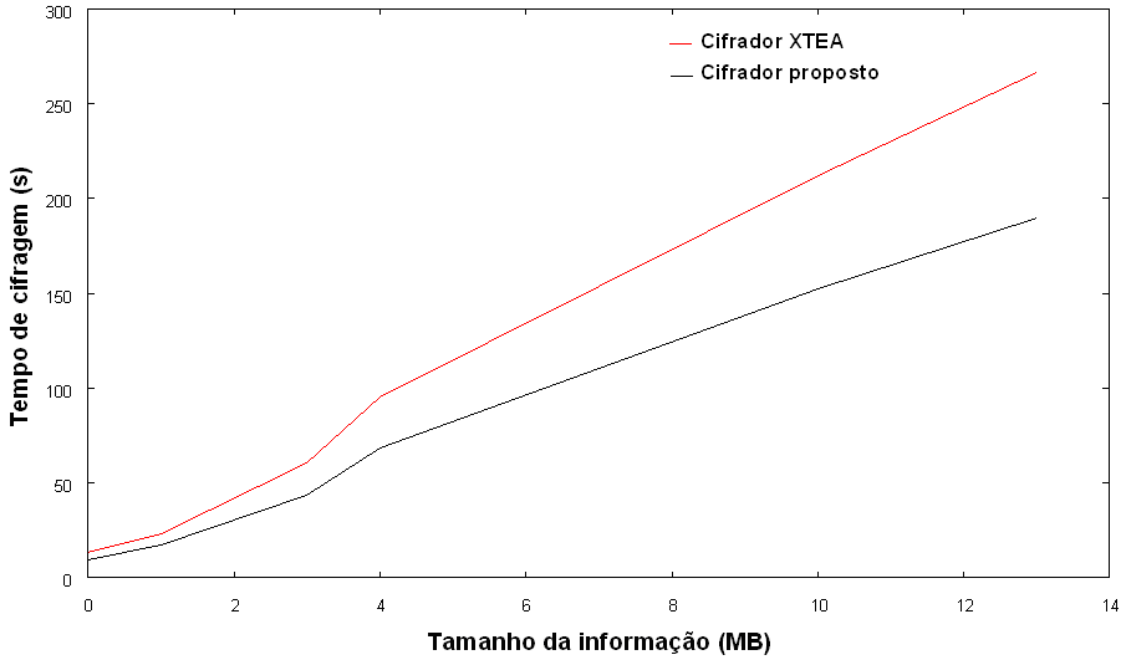
Os principais critérios para avaliação das operações dos cifradores são: segurança, funcionalidade e desempenho (MENEZES; VANSTONE; OORSCHOT, 1996).

Seguindo o exposto acima, esta seção verifica o desempenho do cifrador proposto. Para tal teste, o desempenho do cifrador é comparado com um cifrador utilizado no âmbito comercial, o XTEA (extended TEA), conforme Apêndice A. O experimento foi realizado em um microcomputador com microprocessador Pentium (R) Dual – Core E5400 2,7 GHZ e 4 GB de memória RAM.

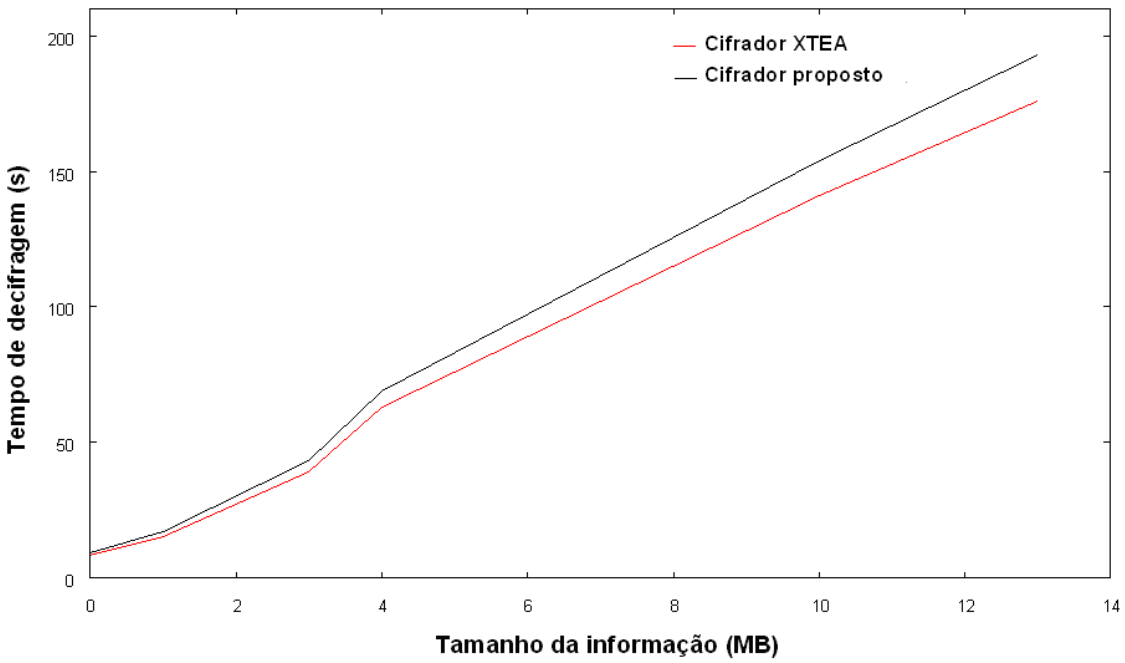
O teste observou os tempos de cifragem e decifragem gerados pelos cifradores. As informações dos tipos áudio e vídeo foram analisadas através de oito informações. O tamanho destas informações variou entre 122 KB e 13, 4 MB. A

figura 5.5 (a-b) ilustra as variações de tempo na cifragem e decifragem em relação aos tamanhos das informações.

Figura 5.5 – Comparação de desempenho na cifr agem e decifragem entre os cifradores: proposto e XTEA. (a) Variações de tempo na cifragem em relação aos tamanhos das infor mações. (b) Variações de tempo na decifragem em relação aos tamanhos das informações.



(a)



(b)

Conforme o gráfico obtido na figura 5.5 (a), o tempo na cifragem do cifrador proposto em relação as diversas informações analisadas (112 KB - 13,4MB) é menor que o tempo obtido pelo XTEA. Entretanto, o tempo na decifragem do cifrador proposto é maior em relação ao XTEA, como indica o gráfico na figura 5.5 (b). No entanto, neste teste não foram aplicados os critérios específicos utilizados na engenharia de software para comparação de desempenho. Desse modo, a partir desta análise é possível apenas garantir que o cifrador proposto seja funcional e com desempenho compatível em relação ao cifrador XTEA.

5.8 Perspectivas Futuras

Os testes experimentais realizados no capítulo 5 investigam a aplicação do sistema de Chua na criptografia de dados.

Para isso, a operação caótica é combinada com os métodos de cifragem/decifragem. Entretanto, o cifrador proposto acrescenta duas operações mencionadas nas seções 4.4-5 e implementadas em Anexo B, porém não utilizadas no capítulo 5. Na esteira do exposto, visando otimizar a robustez do cifrador, tais operações podem ser exploradas em continuidade para trabalhos futuros.

O desempenho do cifrador proposto se mostrou compatível com o do cifrador XTEA. No entanto, o cifrador proposto necessita ser otimizado visando a um futuro relacionado às atividades comerciais. Neste caso, a estrutura (algoritmo) do cifrador ainda pode ser melhorada.

5.9 Resumo das Publicações

Artigo regular: “Cryptography with Chaos”

O artigo investiga a eficiência do sistema caótico de Chua aplicado na criptografia de dados. Para isso propõe um novo algoritmo de criptografia, onde combina as órbitas geradas pelo sistema caótico com a informação original. O trabalho foi apresentado oralmente no DINCON’2010 (Brazilian Conference on Dynamics, Control and their Applications) na cidade de Serra Negra-SP, entre os dias 07 e 11 do mês de Junho de 2010.

Resumo expandido: “Cryptography with chaos using Chua’s attractor”

O resumo analisa o attrator obtido pelo sistema de Chua e sua utilização na criptografia de dados. A eficiência da aplicação é comprovada através dos resultados obtidos. Este resumo foi apresentado na *International Conference on Chaos and Nonlinear Dynamics* (Dynamics Days South America 2010), no Instituto Nacional de Pesquisas Espaciais (INPE), na cidade de São José dos Campos-SP, entre os dias 26 e 30 do mês de Julho de 2010.

Resumo expandido: “The electronic bouncing ball circuit in a communication system”

O trabalho propõe um esquema de comunicação caótica baseado no comportamento do circuito eletrônico bouncing ball utilizando sincronização entre os sistemas transmissor e receptor. Esse resumo foi apresentado na *International Conference on Chaos and Nonlinear Dynamics* (Dynamics Days South America 2010), no Instituto Nacional de Pesquisas Espaciais (INPE), na cidade de São José dos Campos-SP, entre os dias 26 e 30 do mês de Julho de 2010.

Artigo regular (Aceito para publicação): “Cryptography with chaos using Chua’s system”

Este trabalho refere-se à versão completa do resumo expandido “Cryptography with chaos using Chua’s attractor.”

6 CONCLUSÕES

Este trabalho teve por missão precípua apresentar um novo algoritmo de criptografia destinado a proteger uma informação a ser transmitida ou armazenada diante de usuários não autorizados.

Como minuciosamente detalhado, a segurança do cifrador é obtida através do comportamento gerado pelo sistema de Chua, que é utilizado como gerador de chaves criptográficas. Para tanto, uma chave criptográfica é usada na cifragem de cada componente da informação original. Característica de especial relevância desse cifrador consiste na utilização de cada chave criptográfica apenas uma vez.

O cifrador proposto se fundamenta nos seguintes critérios: não necessita de sincronismo entre sistemas transmissor e receptor; independência do alfabeto, isto é, utilização do código ASCII como referência (o que implica no aperfeiçoamento do grau de segurança e adaptabilidade do cifrador); e portabilidade para diversos ambientes (desktop, laptop e canais de transmissões), isto é, ele não se restringe apenas a canais de transmissões, mas pode ser aplicado também sobre dados armazenados.

O desempenho do cifrador proposto é analisado, e assim aplicado sobre informações do tipo texto, imagem, áudio e vídeo. Neste caso, funções *hash* são utilizadas sobre informações: original, cifrada e decifrada (recuperada). É certo que o teste indicou diferenças nos valores obtidos pelas funções *hash* entre as informações: original e cifrada. Contudo os valores gerados pelas funções *hash* sobre as informações - original e decifrada (recuperada) - são iguais, isto é, a informação decifrada (recuperada) é idêntica à informação original. Assim demonstrou-se eficiência nos processos de cifragem e decifragem (recuperação) dos dados.

A técnica de análise de freqüência verificou a segurança do cifrador proposto, e assim demonstrando um comportamento uniforme na ocorrência dos componentes cifrados na informação, o que significa dizer que a uniformidade alcançada na frequência dos componentes de C deve-se à eficiência da irregularidade do comportamento obtido pela operação caótica.

A entropia de Shannon foi utilizada para medir o grau de desorganização existente na informação. Os resultados obtidos indicam um grau signifcativo de desorganização existente na informação cifrada gerada pelo cifrador.

A sensibilidade do cifrador também foi objeto de demonstração e análise. Neste teste, os valores obtidos indicam variações significativas no grau de dispersão dos dados entre as versões cifradas de uma informação. Dessa forma, comprova-se a eficiência da sensibilidade da operação caótica, e assim sendo de grande importância para a rotustez do cifrador proposto.

Além disso, os desempenhos dos cifradores: proposto e XTEA são comparados e analisados. Desse modo, a partir desta análise é possível garantir que o desempenho do cifrador proposto é compatível com o do cifrador XTEA.

Conclui-se que o cifrador proposto necessita ser otimizado para melhorar o seu desempenho. Nesse caso, a estrutura (algoritmo) do cifrador ainda pode ser melhorada. Diante dos resuldados e conclusões acima atingidos, o cifrador proposto demonstrou ser funcional, a ponto dele poder ser aplicado na criptografia de informação para sistemas de segurança.

REFERÊNCIAS

ALLIGOOD, K.; SAUER, T. D.; YORKE, J. *A. Chaos an introduction to dynamical Systems.* New York: Addison-Wesley, 1996.

ÁLVAREZ, G. et al. Cryptanalysis of an ergotic chaotic cipher. *Physics Letters A*, n. 311, p. 172-179, 2003.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR ISO/IEC 17799:* código de prática para a gestão da segurança da informação. Rio de Janeiro, 2005.

BAPTISTA, M. S. Cryptography with Chaos. *Physics Letters A*, n. 240, p. 50-54, 1998.

BERENT, A. AES (advanced encryption standard) simplified, tech. rep. *ABI Software Development*, 2003.

BRAGA, B. *Análise de freqüências de línguas*. Disponível em: <http://www.lockabit.coppe.ufrj.br/downloads/academicos/analise_freq.pdf>. Acesso em: 25 ago. 2009.

BURNETT, S.; PAINE, S. *Criptografia e segurança* - o guia oficial RSA*.* Rio de Janeiro: Campus, 2002.

CERT - O Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. *Segurança de Computadores*. Disponível em: <http://www.cert.br/stats/incidentes/2010-jan-dec/fraude.html>. Acesso em: 05 mar. 2011a.

_____. *Incidentes reportados ao CERT.BR* – Janeiro a Dezembro de 2010. Disponível em: <http://cartilha.cert.br/ conceitos/sec1.html#subsec1.1>. Acesso em: 05 mar. 2011b.

CHEVENING, B.; HOLLOWAY, R. Related-key rectangle attack on 36 rounds of the XTEA block cipher. *International Journal of Information Security*. v. 8, n. 1, p. 1-11, Springer 2009.

CHUA, L. O.; KOMURO, M.; MATSUMOTO, T. The double scroll family. *IEEE Trans Circ Syst I,* v. 33, n. 11, p.1072-118, 1986.

COSTA, J. A. F.; NETO, A. D. D.; SOUZA, G. F. *Compressão auto-adaptativas de imagens coloridas.* Disponível em: <http://www.sbmac.org.br/eventos/cnmac/cd_xxviii_cnmac/resumos%20estendidos>. Acesso em: 21 mar. 2011.

DAEMEN, J.; RIJMEN, V. Advanced encryption standard (AES), tech. rep., National Institute of Standards and Technology NIST*. Federal Information Processing Standards Publication.* 197, 2001.

FEBRABAN. O setor bancário em números (Pesquisa). Disponível em: <http://www.febraban.org.br/Acervo1.asp?id_texto=214&id_pagina=85&palavra=>. Acesso em: 05 mar. 2011.

GARCIA, A. L. *Numerical methods for physics.* River, New Jersey: Prentice-Hall, 2000.

GLEICK, J. *Caos: A construção de uma nova ciência.* Rio de Janeiro: Campus, 1990.

GUZMAN, G. et al. Synchronization of Chua´s circuits with multi – scroll attractors: Applications to communications. *Commun NonLinear Sci Numer Simulat,* p. 2765-2775, October 2008.

HONG, H.; VAIDYA, P. G. Implementation of chaotic cryptography with chaotic synchronization. *Physical Review E*, v. 57, n. 2, p. 1532-1535, February 1998.

KADANOFF, L. P. Roads to Chaos. *Physics Today*, v. 36, issue 12, p. 46-53, December 1983.

LEE, D. Hash function vulnerability index and Hash Chain Attacks, *IEEE ICNP NPSEC*, Nov. 2007.

MADAN, R. N. *Chua's circuit:* a paradigm for chaos. Singapore: World Scientific; 1993.

MENEZES, A. J.; VANSTONE, S. A.; OORSCHOT, P. C. V. *Handbook of Apllied Cryptography.* EUA: CRC Press, 1996.

OLIVEIRA, L. P. L.; SOBOTTKA, M. Cryptography with chaotic mixing. *Chaos Solitons and Fractals,* v. 35, p. 466-471, May 2008.

PECORA, L. M.; CARROLL, T. L. Synchronization in chaotic systems. *Phys. Rev. Lett.,* v. 64, p. 821-824, Feb. 1990.

SCHNEIER, B. *Applied cryptography.* 2. ed. [S.l.]: John Wiley and Sons, 1996.

SHANNON, C. Communication theory of secrecy systems. *Bells Systems Technical Journal*, v. 28, n. 4, p. 656-715, 1949.

SINGH, S. *O livro dos códigos. 7. ed.* Rio de Janeiro: Record, 2008.

STALLINGS, W. *Cryptography and Network Security*: principles and practice*.* 2. ed. New Jersey: Prentice-Hall, 2002.

STALLINGS, W. *Cryptography and Network Security:* principles and practices*.* 4. ed. [S.l.:s.n.], 2005.

WEBSTER, A. F.; TAVARES, S. E. On the design of S-boxes. *Lecture Notes in Computer Science*, v. 218, p. 523-534, 1986.

APÊNDICES

APÊNDICE A – Algoritmo XTEA

O XTEA (extended TEA) é um cifrador utilizado no âmbito comercial. Tal cifrador foi desenvolvido por David Wheeler e Roger Needham em 1997. O XTEA é definido como um cifrador de blocos porque ele cifra blocos de 64 bits, e, durante este processo, utiliza chaves criptográficas de 128 bits, assim tornando-os resistentes às diversas técnicas de criptoanálise. A difusão é garantida através das diversas rodadas realizadas sobre suas operações. Como o cifrador XTEA não é patenteado, tem-se que ele pertence ao domínio público, podendo ser livremente utilizado por qualquer um que por ele se interesse (CHEVENING; HOLLOWAY, 2009).

O cifrador XTEA foi desenvolvido devido às limitações de segurança do cifrador TEA (Tiny Encryption Algorithm). Essas limitações estão relacionadas a ataques de chaves equivalentes e chaves relacionadas. Deste modo, o XTEA segue os mesmos princípios de operações do TEA. No entanto, com objetivo de prevenção de ataques baseados nas chaves, as quatro sub-chaves são misturadas de maneira mais irregular e com freqüência menor (CHEVENING; HOLLOWAY, 2009). A implementação do algoritmo XTEA na linguagem de programação C++ segue abaixo.

Código Fonte: Algoritmo XTEA

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#include <stdio.h>
#include <time.h>
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
```

```
//---------------------------------------------------------------------------
void TForm1::encipher(unsigned int num_rounds, uint32_t v[2], uint32_t const k[4])
{
   unsigned int i;
   uint32_t v0=v[0], v1=v[1], sum=0, delta=0x9E3779B9;
   for (i=0; i < num_rounds; i++) {
      v0 += (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + k[sum & 3]);
      sum += delta;
      v1 += (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + k[(sum>>11) & 3]);
   }
   v[0]=v0; v[1]=v1;
}
void TForm1::decipher(unsigned int num_rounds, uint32_t v[2], uint32_t const k[4])
{
   unsigned int i;
   uint32_t v0=v[0], v1=v[1], delta=0x9E3779B9, sum=delta*num_rounds;
   for (i=0; i < num_rounds; i++) {
      v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + k[(sum>>11) & 3]);
      sum -= delta;
      v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + k[sum & 3]);
   }
   v[0]=v0; v[1]=v1;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   double START, END;
   START = clock();
   TempInicio->Text = 0;
   uint32_t v[2], chave[4];
   uint32_t tamaarq;
   FILE *origem, *destino;
   char buffer[2];
   int q;
   tamaarq = FileSizeByName(nomearq->Text);
   pb1->Max = tamaarq;
   pb1->Position = 0;
   tamaarq = 0;
   chave[0] = edtch->Text[1];
   chave[1] = edtch->Text[2];
   chave[2] = edtch->Text[3];
   chave[3] = edtch->Text[4];
   if((origem=fopen(nomearq->Text.c_str(),"rb"))!=NULL) {
     if((destino=fopen(DESTINO,"wb"))!=NULL) {
       do {
         q = fread(buffer,sizeof(char),2,origem);
         if(q>0) {
           v[0] = buffer[0]; v[1] = buffer[1];
           encipher(QDEPASSOS,v,chave);
           fwrite(v,sizeof(uint32_t),2,destino);
         }
         tamaarq += q; pb1->Position = tamaarq;
         Application->ProcessMessages();
       } while(q==2);
       fclose(destino);
     }
     fclose(origem);
   }
   DeleteFile(nomearq->Text);
   RenameFile(DESTINO,nomearq->Text);
   pb1->Position = 0;
   END = clock();
   TempInicio->Text = (END-START);
   MessageBox(Form1->Handle,"Operação de cifragem
```

```
                        concluída","Atenção",MB_ICONINFORMATION);
                        }
                        //---------------------------------------------------------------------------
                        void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
                        {
                             if(od1->Execute()) {
                             nomearq->Text = od1->FileName;
                           }
                        }
                        //---------------------------------------------------------------------------
                        void __fastcall TForm1::Button2Click(TObject *Sender)
                        {
                           double START, END;
                           START = clock();
                           TempInicio->Text = 0;
                           uint32_t v[2], chave[4];
                           uint32_t tamaarq;
                           FILE *origem, *destino;
                           int q;
                           char buffer[2];
                           tamaarq = FileSizeByName(nomearq->Text);
                           pb1->Max = tamaarq;
                           pb1->Position = 0;
                           tamaarq = 0;
                           chave[0] = edtch->Text[1];
                           chave[1] = edtch->Text[2];
                           chave[2] = edtch->Text[3];
                           chave[3] = edtch->Text[4];
                           if((origem=fopen(nomearq->Text.c_str(),"rb"))!=NULL) {
                           if((destino=fopen(DESTINO,"wb"))!=NULL) {
                              do {
                                q = fread(v,sizeof(uint32_t),2,origem);
                                if(q>0) {
                                  decipher(QDEPASSOS,v,chave);
                                  buffer[0] = v[0]; buffer[1] = v[1];
                                  fwrite(buffer,sizeof(char),2,destino);
                                }
                                tamaarq += q; pb1->Position = tamaarq;
                                Application->ProcessMessages();
                              } while(q==2);
                              fclose(destino);
                            }
                            fclose(origem);
                           }
                           DeleteFile(nomearq->Text);
                           RenameFile(DESTINO,nomearq->Text);
                           pb1->Position = 0;
                           END = clock();
                           TempInicio->Text = (END-START);
                           MessageBox(Form1->Handle,"Operação de decifragem
concluída","Atenção",MB_ICONINFORMATION);
                        }
```

ANEXOS

ANEXO A – Resolução numérica do sistema de Chua

Código Fonte: Resolução numérica do sistema de Chua

Linguagem de Programação: C++

```
#include<stdio.h>
#include<math.h>
#define ite 3000
double ret[ite][3];
double f(double x1){
 dou             ble a,b,c,ret;
 a=             2.0/7.0;
 b=-1.0/7.0;
 c=             1.0;
 ret=(b*x1)+((0.5*(a-             b))*(sqrt((x1+c)*(x1+c))-sqrt((x1-c)*(x1-c))));
 return              ret;
}
void chua(double x1,double x2,double x3){
 dou             ble x1t,x2t,x3t,alfa,beta,dt;
            int cont;
 alfa=9.0;
 beta=             14.28;
 dt=             0.034;
 for(cont=             0;cont<ite;cont++){
            x1t=x1+((alfa*(x2-f(x1)))*dt);
                        x2t=x2+((x1-x2+x3)*dt);
                        x3t=x3+((-beta*x2)*dt);
                        x1=x1t;
                        x2=x2t;
                        x3=x3t;
                        ret[cont][0]=x1;
                        ret[cont][1]=x2;
                        ret[cont][2]=x3;
 }
}
int main(void){
 int             cont;
            chua(0.6,0.2,0.70001);
            FILE *fp = fopen("C:\\chua2.txt", "w");
   for(cont=0;cont<ite;cont++){
 fprintf(fp,"%lf             %lf %lf\n",ret[cont][0],ret[cont][1],ret[cont][2]);
                        }
 getchar();
            return 0;
}
```

ANEXO B – Cifrador Proposto

Código Fonte: Cifrador proposto

Linguagem de Programação: C++

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#include <stdio.h>
#include <time.h>
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   Encriptar();
}
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
   if(od1->Execute()) {
     nomearq->Text = od1->FileName;
   }
}
void __fastcall TForm1::Button2Click(TObject *Sender)
{
   Desencriptar();
}
//---------------------------------------------------------------------------
void TForm1::Encriptar(void)
{
   double START, END;
   TempInicio->Text = 0;
   START = clock();
   FILE *origem, *destino;
   char buffer[MAXBUFFER], s[40], *pt, saida[MAXBUFFER];
   uint32_t tamaarq, v[MAXBUFFER], anter[MAXBUFFER], aux;
   uint32_t saidaINT;
   int i,q,q1,xch[4];
   double e[3], fator;
   int modulo,i1;
   float d1;
   double modINT, modREST;
   limpar_buffer(v, anter);
   tamaarq = FileSizeByName(nomearq->Text);
   pb1->Max = tamaarq;
   pb1->Position = 0;
   tamaarq = 0;
   xch[0] = edtCH1->Text[1];
   xch[1] = edtCH1->Text[2];
   xch[2] = edtCH1->Text[3];
   xch[3] = edtCH1->Text[4];
   e[0] = ((x    ch[0] * 1000000) + (x       ch[1] * 10    000) +   (xch[  2] * 1    00) +     xch[3]) /
(double)100000000;
   xch[0] = edtCH1->Text[5];
   xch[1] = edtCH1->Text[6];
```

```
                        xch[2] = edtCH1->Text[7];
                        xch[3] = edtCH1->Text[8];
                        e[1] = ((x    ch[0] * 1000000) + (x       ch[1] * 10    000) +   (xch[  2] * 1    00) +     xch[3]) /
(double)100000000;
                        xch[0] = edtCH1->Text[9];
                        xch[1] = edtCH1->Text[10];
                        xch[2] = edtCH1->Text[11];
                        xch[3] = edtCH1->Text[12];
                        e[2] = ((x    ch[0] * 1000000) + (x       ch[1] * 10    000) +   (xch[  2] * 1    00) +     xch[3]) /
(double)100000000;
                        euler2(e,QDEPASSOS);
                        if((origem=fopen(nomearq->Text.c_str(),"rb"))!=NULL) {
                          if((destino=fopen(DESTINO,"wb"))!=NULL) {
                            do {
                              strcpy(buffer,"");
                              q = fread(buffer,sizeof(char),MAXBUFFER,origem);
                              if(q>0) {
                                q1 = q;
                                for(i=0;i<q;i++) {
                                  v[i] = buffer[i];
                                  modREST = modf(((double)v[i]) / ((double)256), &modINT) * ((double)256);
                                  modREST = modf(modREST, &modINT);
                                  if(modREST<-0.5) modINT--;
                                  else if(modREST>0.5) modINT++;
                                  v[i] = modINT;
                                }
//------------> Modo CBC
                                if(chkCBC->Checked)
                                  for(i=0;i<q;i++) {
                                    v[i] += anter[i];
                                    anter[i] = v[i];
                                  }
//------------> Trocando posições dos bytes
                                if(chkTROCA->Checked && q==4) {
                                  aux = v[0]; v[0] = v[1]; v[1] = aux;
                                  aux = v[2]; v[2] = v[3]; v[3] = aux;
                                }
 //-------------------------------------------------------------------------------------
                                for(i=0;i<q;i++) {
                                  euler2(e, 1);
                                    fator = e[0] + e[1] + e[2];
                                    sprintf(s,"%15.15lf",fator); fator = strtod(s,&pt);
                                    fator = fator * PRECISAO;
                                    d1 = v[i] + fator;
                                    if(chkMOD->Checked) {
                                    modREST = modf(((double)d1) / ((double)256), &modINT) * ((double)256);
                                    modREST = modf(modREST, &modINT);
                                    if(modREST<-0.5) modINT--;
                                    else if(modREST>0.5) modINT++;
                                    v[i] = modINT;
                                  }
                                }
                                if(chkCONCAT->Checked)
                                {
  //------------> Concatenação de bytes
                                   saida[0] = (char)v[0];
                                   saida[1] = (char)v[1];
                                   saida[2] = (char)v[2];
                                   saida[3] = (char)v[3];
                                   fwrite(saida,sizeof(char), q, destino);
                                }
                                else
                                {
                                   v[0] =
                                   ((v[0] & 192) << 24) +
                                   ((v[0] & 48) << 16) +
                                   ((v[0] & 12) << 8) +
```

```
                                    (v[0] & 3);
                                    v[1] =
                                    ((v[1] & 192) << 24) +
                                    ((v[1] & 48) << 16) +
                                    ((v[1] & 12) << 8) +
                                    (v[1] & 3);
                                    v[2] =
                                    ((v[2] & 192) << 24) +
                                    ((v[2] & 48) << 16) +
                                    ((v[2] & 12) << 8) +
                                    (v[2] & 3);
                                    v[3] =
                                    ((v[3] & 192) << 24) +
                                    ((v[3] & 48) << 16) +
                                    ((v[3] & 12) << 8) +
                                    (v[3] & 3);
                                    fwrite(v,sizeof(uint32_t), q, destino);
                                  }
                                  tamaarq += q1; pb1->Position = tamaarq;
                                  Application->ProcessMessages();
                                 }
                                } while(q>0);
                                fclose(destino);
                               }
                               fclose(origem);
                              }
                              DeleteFile(nomearq->Text);
                              RenameFile(DESTINO,nomearq->Text);
                              pb1->Position = 0;
                              MessageBox(Form1->Handle,"Operação de cifragem
concluída","Atenção",MB_ICONINFORMATION);
                             }
                             void TForm1::Desencriptar(void)
                             {
                                double START, END;
                                TempInicio->Text = 0;
                                START = clock();
                                FILE *origem, *destino;
                                char buffer[MAXBUFFER], s[40], *pt;
                                uint32_t  tamaarq,  v[MAXBUFFER],  anter[MAXBUFFER], aux, t    mp[MAXBUFFER],
bufferUINT32[MAXBUFFER];
                                uint32_t bufferINT;
                                int i,q,q1,xch[4];
                                double e[3], fator;
                                int modulo,d1,i1;
                                double modREST, modINT;
                                limpar_buffer(v, anter);
                                tamaarq = FileSizeByName(nomearq->Text);
                                pb1->Max = tamaarq;
                                pb1->Position = 0;
                                tamaarq = 0;
                                xch[0] = edtCH1->Text[1];
                                xch[1] = edtCH1->Text[2];
                                xch[2] = edtCH1->Text[3];
                                xch[3] = edtCH1->Text[4];
                                e[0] = ((x    ch[0] * 1000000) + (x       ch[1] * 10    000) +   (xch[  2] * 1    00) +    xch[3]) /
(double)100000000;
                                xch[0] = edtCH1->Text[5];
                                xch[1] = edtCH1->Text[6];
                                xch[2] = edtCH1->Text[7];
                                xch[3] = edtCH1->Text[8];
                                e[1] = ((x    ch[0] * 1000000) + (x       ch[1] * 10    000) +   (xch[  2] * 1    00) +    xch[3]) /
(double)100000000;
                                xch[0] = edtCH1->Text[9];
                                xch[1] = edtCH1->Text[10];
                                xch[2] = edtCH1->Text[11];
                                xch[3] = edtCH1->Text[12];
```

```
                        e[2] = ((x    ch[0] * 1000000) + (x       ch[1] * 10    000) +   (xch[  2] * 1    00) +     xch[3]) /
(double)100000000;
                        euler2(e,QDEPASSOS);
                        if((origem=fopen(nomearq->Text.c_str(),"rb"))!=NULL) {
                          if((destino=fopen(DESTINO,"wb"))!=NULL) {
                            do {
                              if(chkCONCAT->Checked)
                              {

//------------> Concatenação de bytes
                                q = fread(buffer, sizeof(char), MAXBUFFER, origem);
                                q1 = q;
                                }
                                else
                                {
                                     q = fread(bufferUINT32, sizeof(uint32_t), MAXBUFFER, origem);
                                     bufferUINT32[0] =
                                     ((bufferUINT32[0] & 3221225472) >> 24) +
                                     ((bufferUINT32[0] & 3145728) >> 16) +
                                     ((bufferUINT32[0] & 3072) >> 8) +
                                     (bufferUINT32[0] & 3);
                                      bufferUINT32[1] =
                                     ((bufferUINT32[1] & 3221225472) >> 24) +
                                     ((bufferUINT32[1] & 3145728) >> 16) +
                                     ((bufferUINT32[1] & 3072) >> 8) +
                                     (bufferUINT32[1] & 3);
                                     bufferUINT32[2] =
                                     ((bufferUINT32[2] & 3221225472) >> 24) +
                                     ((bufferUINT32[2] & 3145728) >> 16) +
                                     ((bufferUINT32[2] & 3072) >> 8) +
                                     (bufferUINT32[2] & 3);
                                     bufferUINT32[3] =
                                     ((bufferUINT32[3] & 3221225472) >> 24) +
                                     ((bufferUINT32[3] & 3145728) >> 16) +
                                     ((bufferUINT32[3] & 3072) >> 8) +
                                     (bufferUINT32[3] & 3);
                                     buffer[0] = bufferUINT32[0];
                                     buffer[1] = bufferUINT32[1];
                                     buffer[2] = bufferUINT32[2];
                                     buffer[3] = bufferUINT32[3];
                                     q1 = q * sizeof(uint32_t);
                                }
                                for(i=0;i<q;i++)
                                {
                                 modREST = modf(((double)buffer[i]) / ((double)256), &modINT) * ((double)256);
                                  modREST = modf(modREST, &modINT);
                                  if(modREST<-0.5) modINT--;
                                  else if(modREST>0.5) modINT++;
                                  v[i] = modINT;
                                }
                                if(q>0) {
                                  for(i=0;i<q;i++) {
                                    euler2(e,1);
                                      fator = e[0] + e[1] + e[2];
                                      sprintf(s,"%15.15lf",fator);
                                      fator = strtod(s,&pt);
                                      fator = fator * PRECISAO;
                                      if(chkMOD->Checked)
                                      {
                                       modREST = modf(((double)fator) / ((double)256), &modINT) * ((double)256);
                                       modREST = modf(modREST, &modINT);
                                       if(modREST<-0.5) modINT--;
                                       else if(modREST>0.5) modINT++;
                                       fator = modINT;
                                       d1 = ((int)v[i]) - ((int)(fator));              }
                                       if(d1 < 0) d1 += 256;
                                       tmp[i] = d1;
```

```
                                }
//------------> Trocando posições dos bytes
                                  if(chkTROCA->Checked && q==4) {
                                  aux = tmp[0]; tmp[0] = tmp[1]; tmp[1] = aux;
                                  aux = tmp[2]; tmp[2] = tmp[3]; tmp[3] = aux;
//------------> Modo CBC
                                  if(chkCBC->Checked)
                                  for(i=0;i<q;i++) {
                                  tmp[i] -= anter[i];
                                  anter[i] += tmp[i];
//-------------------------------------------------------------------------------------
                                  for(i=0;i<q;i++) buffer[i] = tmp[i];
                                  fwrite(buffer,sizeof(char),q,destino);
                                  tamaarq += q1; pb1->Position = tamaarq;
                                  Application->ProcessMessages();
                                 }
                                } while(q>0);
                                fclose(destino);
                               }
                               fclose(origem);
                              }
                              DeleteFile(nomearq->Text);
                              RenameFile(DESTINO,nomearq->Text);
                              pb1->Position = 0;
                              MessageBox(Form1->Handle,"Operação de decifragem
concluída","Atenção",MB_ICONINFORMATION);
                            }
                            void TForm1::euler2(double x[3], uint32_t passos)
                            {
                               const double alfa=9.0,beta=14.28,dt=0.034;
                               double x1t,x2t,x3t;
                               uint32_t cont;
                               char s[20], *pt;
                                  for(cont=0;cont < passos;cont++){
                                        x1t = x[0] + (x[1] - f_chua(x[0])) *(alfa * dt);
                                        x2t = x[1] + (x[0] - x[1] + x[2]) * dt;
                                        x3t = x[2] + x[1] * (-beta * dt);
                                        x[0] = x1t;
                                        x[1] = x2t;
                                        x[2] = x3t;
                                  }
                               sprintf(s,"%8.6lf",x[0]); x[0] = strtod(s,&pt);
                               sprintf(s,"%8.6lf",x[1]); x[1] = strtod(s,&pt);
                               sprintf(s,"%8.6lf",x[2]); x[2] = strtod(s,&pt);
                            }
                            //---------------------------------------------------------------------------
                            double TForm1::f_chua(double x1)
                            {
                               double a,b,c,ret;
                               a = 2.0/7.0;
                               b = -1.0/7.0;
                               c = 1.0;
                               ret = (b*x1)+((0.5*(a-b))*(x1+c)-abs(x1-c));
                             return ret;
                            }
                            {
                            uint32_t TForm1::fnCONCAT(uint32_t m[MAXBUFFER])
                            {
                               uint32_t res;
                               unsigned short n;
                               n = m[0]; res = n; res <<= 8;
                               n = m[1]; res = res | n; res <<= 8;
                               n = m[2]; res = res | n; res <<= 8;
                               n = m[3]; res = res | n;
                               return res;
                            }
```

```
void TForm1::fnDESCONCAT(uint32_t m[MAXBUFFER], uint32_t n)
{
   m[0] = (n & 0xFF000000) >> 24;
   m[1] = (n & 0x00FF0000) >> 16;
   m[2] = (n & 0x0000FF00) >> 8;
   m[3] = (n & 0x000000FF);
}
void TForm1::limpar_buffer(uint32_t b1[MAXBUFFER], uint32_t b2[MAXBUFFER])
{
   int i;
   for(i=0;i<MAXBUFFER;i++) b1[i] = b2[i] = 0;
}
```

ANEXO C – Resolução Numérica do Mapa Logístico

Código Fonte: Resolução numérica do mapa logístico

Linguagem de Programação: C++

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
FILE *output;
int main(){
 int i;
 double x=0.2,xn,xnn;
 double r=4.0;
 output=fopen("c:\\logistico.xls","w");
 for(i=0;i<=100;i++){
        xn=r*x*(1-x);
            xnn=r*xn*(1-xn);
    fprintf(output,"%d\t%f\t%f\t%f\n",i,x,xn,xnn);
    x=xn;
  }
    fclose(output);
  return 0;
}
```

ANEXO D – Diagrama de bifurcações do mapa logístico

Código Fonte: Diagrama de bifurcações do mapa logístico

Linguagem de Programação: C++

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
FILE *bifurca;
int main(){
  int i;
  double x,xn;
  double r;
  bifurca=fopen("c:\\bifurca.txt","w");
  x=0.1;
  for(r=2.6;r<=4;r+=0.001){
    for(i=1;i<=100;i++){
      xn=r*x*(1-x);
      fprintf(bifurca,"%f\t%f\n",r,x);
      x=xn;
    }
  }
  fclose(bifurca);
  return 0;
}
```

ANEXO E – Análise de frequência dos caracteres

Código Fonte: Análise de frequência dos caracteres

Linguagem de Programação: C++

```
#include <stdio.h>
int main(int argc, char *argv[]){
  char arqt[]="c:\\analise.txt";
  FILE *arq=fopen(arqt,"r+b");
  unsigned char carac;
  unsigned int vet[256];
  int cont;
  long tam;
  fseek (arq, 0, SEEK_END);
  tam = ftell (arq);
  fseek (arq, 0, SEEK_SET);
  for(cont=0;cont<256;cont++){
   vet[cont]=0;
  }
  for(cont=0;cont<tam;cont++){
   fread(&carac, sizeof(char), 1, arq);
   vet[(int)carac]=vet[(int)carac]+1;
  }
    FILE *fp = fopen("C:\\analise.txt", "w");
    for(cont=0;cont<256;cont++){
    fprintf(fp,"%d      %d\n",cont,vet[cont]);
    }
  fclose(arq);
  getchar();
 return 0;
}
```