UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MODELO CRIPTOGRÁFICO BASEADO EM AUTÔMATOS CELULARES TRIDIMENSIONAIS HÍBRIDOS

DANIELLI ARAÚJO LIMA

Uberlândia - Minas Gerais
2012

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DANIELLI ARAÚJO LIMA

MODELO CRIPTOGRÁFICO BASEADO EM AUTÔMATOS CELULARES TRIDIMENSIONAIS HÍBRIDOS

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Inteligência Artificial.

Orientadora:
Prof[a]. Dr[a]. Gina Maira Barbosa de Oliveira

Uberlândia, Minas Gerais
2012

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU

L732m Lima, Danielli Araújo, 1989-
Modelo criptográfico baseado em autômatos celulares tridimensionais híbridos / Danielli Araújo Lima. - 2012.
224 f. : il.

Orientadora: Gina Maira Barbosa de Oliveira.

Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Inteligência artificial - Teses. 2. Criptografia - Teses. I. Oliveira, Gina Maira Barbosa de. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Modelo criptográfico baseado em autômatos celulares tridimensionais híbridos**" por **Danielli Araújo Lima** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 3 de Julho de 2012

Orientadora: _____

Profª. Drª. Gina Maira Barbosa de Oliveira
Universidade Federal de Uberlândia

Banca Examinadora:

_____

Prof. Dr. Bruno Augusto Nassif Travençolo
Universidade Federal de Uberlândia

_____

Prof. Dr. Nei Yoshihiro Soma
Instituto Tecnológico de Aeronáutica

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: Julho de 2012

Autor:	**Danielli Araújo Lima**
Título:	**Modelo criptográfico baseado em autômatos celulares tridimensionais híbridos**
Faculdade:	**Faculdade de Computação**
Grau:	**Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

_____

Autor

Dedicatória

Aos meus pais pelo amor incondicional, compreensão, exemplo de vida e apoio irrestrito em todos os momentos de minha vida e ao meu irmão pelo carinho e incentivo.

Agradecimentos

Agradeço...

A minha orientadora e professora Gina, pela apresentação de um tema tão interessante que são os ACs, pela confiança depositada desde a Iniciação Científica e também por sua valiosa contribuição ao longo do desenvolvimento e revisão teórica deste trabalho.

Ao PPGCC, pela oportunidade que me foi concedida para a realização deste Mestrado, disponibilizando todos os meios materiais e humanos para a sua concretização.

Sem citar nomes para não correr o risco de algum esquecimento, quero também agradecer a todos os professores da FACOM que contribuíram para a minha formação desde a graduação.

Ao funcionário Erisvaldo, pela prontidão que sempre mostrou para resolver assuntos importantes do meu mestrado.

A todos os meus amigos, pela troca de conhecimentos durante os estudos e execução de trabalhos, pela convivência e também pelos momentos de descontração.

A minha família, pelo constante apoio, amor e dedicação ao longo de minha formação pessoal e profissional, que foram essenciais para superar cada uma das etapas de minha vida. Em especial, a minha mãe que me auxiliou em etapas importantes desse trabalho.

Aos integrantes da Banca Examinadora, pelos comentários e sugestões apresentadas com o objetivo de valorizar o trabalho.

A todos que, de alguma forma, contribuíram para que este trabalho se realizasse.

Finalmente, a CAPES, pelo apoio financeiro.

"A engenhosidade humana não pode arquitetar uma escrita secreta que a própria engenhosidade humana não possa resolver."

(Edgar Allan Poe)

Resumo

A popularização de equipamentos eletrônicos que capturam imagens digitais, bem como a criação de várias aplicações que possibilitam a troca e compartilhamento das mesmas através de canais de comunicação provocaram o surgimento de alguns problemas inerentes. Dentre esses problemas estão a quebra de segurança e privacidade em sistemas de informação. Para que essas questões sejam resolvidas, a utilização de sistemas criptográficos na transmissão dessas informações se faz necessária. O principal objetivo da criptografia é a garantia de que as informações transmitidas não serão copiadas, modificadas ou falsificadas. Muitos algoritmos de criptografia clássica e moderna já foram investigados para garantir a troca segura de informações. Entretanto, quando a mensagem que estiver sendo enviada for uma imagem, os algoritmos convencionais não favorecem o arranjo nem a quantidade massiva de informações que são características estruturais das mesmas. Assim, uma nova abordagem para este tema são os autômatos celulares (ACs), que estão sendo estudados pela simplicidade de implementação e também por sua capacidade de processar dados em paralelo. Nesse trabalho um novo modelo criptográfico tridimensional baseado autômatos celulares híbridos foi elaborado e chamado de 3DHCA. Esse método utiliza o cálculo de pré-imagens, que corresponde à evolução do AC para trás, e duas regras no processo de cifragem. Uma regra principal caótica é responsável pela cifragem efetiva da mensagem e uma regra de contorno, que garante a existência da pré-imagem para qualquer reticulado possível. Esse modelo tridimensional baseia-se em um precursor chamado HCA que utiliza ACs unidimensionais. Inicialmente, uma análise da segurança do modelo unidimensional foi realizada através da Teoria dos Grafos. A definição do novo modelo de criptografia tridimensional, bem como os experimentos realizados para sua validação e refinamento, são discutidos nessa dissertação. O método resultante se mostrou seguro e passível de implementação com alto nível de paralelismo. Assim, se torna uma opção extremamente interessante para a cifragem de grandes volumes de dados, especialmente as imagens digitais coloridas.

Palavras chave: autômato celular, cálculo de pré-imagens, encriptação de imagens, criptografia simétrica, teoria dos grafos.

Abstract

The popularization of electronic devices to capture digital images, as well as the creation of several applications enabling exchange and sharing these images by communication channels has brought some inherent problems. Among these problems it can be cited the security violation and privacy in information systems. The use of cryptographic systems in such information transmission is required to solve these issues. The main goal of cryptography is to guarantee that the information will not be copied, modified or falsified. Many classic and modern encryption algorithms have been investigated to ensure secure exchange of information. However, when the message to be sent is an image, the conventional algorithms do not favor image structural features: the spatial arrangement and the massive amount of information. Thus, a new approach for this topic is the usage of cellular automata (CA) in image encryption. CA are being studied due to their implementation simplicity and also due to their capacity to process data in a fast parallel way. In this work a novel cryptographic model based on three-dimensional hybrid cellular automata was elaborated and named 3DHCA. This method employs pre-image calculus which represents CA backward evolution and two rules are used in the encryption process. The first one is called major rule, and it is responsible for the effective encryption of the message. The second one is used only in boundary lattice cells to ensure the existence of pre-image for any lattice. This three-dimensional model is based on a precursor model called HCA, which uses one-dimensional CA. Initially, an analysis of one-dimensional model safety was performed using Graph Theory. The definition of the new three-dimensional cryptographic model, as well as experiments for its validation and refinement are discussed in this dissertation. The resulting method has proved to be safe and appropriated for an implementation with a high-level of parallelism. Thus, it becomes a very interesting option for encryption using large volumes of data, especially digital color images.

Keywords: cellular automata, pre-image calculus, image encrytion, symmetric cipher, graph theory.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o aumento dos sistemas de informação conectados à rede mundial de computadores tornou-se mais frequente a troca de dados entre entidades. Na maioria das vezes, precisamos proteger as informações que estão sendo transmitidas, pois muitas dessas informações são importantes e devem receber um tratamento de sigilo. A ferramenta mais significativa para realizar esta tarefa é a criptografia.

O principal objetivo da criptografia de dados é possibilitar que duas entidades se comuniquem ao longo de um canal de transmissão, de tal forma que nenhum oponente consiga decifrar a mensagem que é enviada. A criptografia também é estudada como abordagem para minimização da vulnerabilidade de dados e recursos, bem como para a garantia de confidenciabilidade, integridade e autenticidade durante a transmissão de dados.

A transmissão de arquivos de imagens através da Internet tornou-se comum com o advento e posteriormente com a popularização dos equipamentos eletrônicos que capturam imagens digitais. Os algoritmos de criptografia clássica existentes não tratam de forma adequada a grande redundância de informações, nem a quantidade massiva de informações, que são características inerentes às imagens. Isso motivou a criação de um novo campo na criptografia que estuda algoritmos para a cifragem de imagens [Yu et al., 2010].

A teoria do caos [Goncalves et al., 2010] tem sido aplicada para explicar o funcionamento de sistemas complexos e dinâmicos e tem atraído a atenção de diversos pesquisadores por sua aplicação em sistemas criptográficos. Esses sistemas são fortemente dependentes de seus parâmetros iniciais. Pequenas diferenças iniciais levam o sistema a exibir resultados totalmente distintos, no que diz respeito à sua evolução temporal. A maior consequência é que estes sistemas são sensíveis à perturbações e ruídos, levando a evoluções de estados que são na maioria das vezes imprevisíveis. Por outro lado, sistemas criptográficos devem garantir propriedades como confusão e difusão [Zeghid et al., 2007]. A ideia subjacente a vários métodos recentes é aproveitar a aleatoriedade inerente ao comportamento caótico e aplicar em sistemas criptográficos [Goncalves et al., 2010].

Os autômatos celulares (ACs) são sistemas dinâmicos discretos no tempo, no espaço

e nas suas variáveis. Além disso, já foi provado que os ACs são modelos computacionais que possuem computabilidade universal [Oliveira, 2003]. Os ACs estão sendo estudados em diversas áreas, tais como, na biologia evolutiva [Powathil et al., 2012], na dinâmica das reações químicas [Sarkar and Abbasi, 2006], nos sistemas dinâmicos da física [Puliafito, 2007] e até mesmo no comportamento de mercados [Huang et al., 2008]. Duas das principais vantagens em se aplicar modelos baseados em ACs reside na sua simplicidade de implementação e também na sua capacidade de processar dados em paralelo. Neste trabalho, um novo modelo de criptografia para imagens baseado em ACs tridimensionais caóticos e não homogêneos é investigado. Essa técnica utiliza o cálculo de pré-imagens (evolução do AC para trás) no processo de cifragem, a exemplo de outros métodos investigados na literatura [Gutowitz, 1995], [Macedo, 2007] e [Magalhaes, 2010]. Além disso, uma versão unidimensional desse modelo, proposta anteriormente em [Macedo, 2007], será analisada através da Teoria dos Grafos.

1.1 Objetivo

Este trabalho tem por objetivo: (i) análise do modelo unidimensional HCA, proposto em [Macedo, 2007], através da teoria dos grafos e a (ii) proposição de um novo modelo de criptografia através do cálculo de pré-imagens utilizando-se autômatos celulares tridimensionais caóticos, não homogêneos e não aditivos.

A primeira fase do trabalho fundamenta-se em analisar o modelo de criptografia proposto em [Macedo, 2007] sob o ponto de vista teórico. Além disso, buscando evidências que o modelo HCA é seguro através de uma análise matemática, poderíamos estender os conceitos para os modelos de criptografia com dimensão superior à unidimensional.

A segunda tarefa desenvolvida na dissertação de mestrado consiste na concepção e avaliação de um novo modelo de criptografia baseado em ACs tridimensionais. As características principais dos modelos unidimensional e bidimensional propostos respectivamente em [Macedo, 2007] e [Magalhaes, 2010] serão preservadas. O cálculo da pré-imagem representa a base do processo da criptografia. Na cifragem, o cálculo de pré-imagem (*backward*) é repetido por vários passos de tempo, seguindo a mesma ideia do método unidimensional proposto em [Macedo, 2007]. Para que o texto final encriptado seja do mesmo tamanho do texto claro é necessário a aplicação de duas regras no processo de cifragem: uma regra principal sensível a um dos extremos da vizinhança e uma regra de contorno que possui a mesma sensitividade da regra principal. A regra de contorno é aplicada em uma porção menor das células do reticulado (somente nas bordas) associado à imagem, enquanto que a regra principal com dinâmica caótica é aplicada a todas as demais células. O processo de decifragem pode ser realizado de forma paralela aplicando-se as regras principal e de contorno no reticulado tridimensional para frente (*forward*) pelo mesmo número de passos de tempo utilizado na cifragem.

1.2 Estrutura do texto

Esta dissertação está dividida em oito capítulos com o objetivo de transmitir as informações necessárias ao entendimento, contextualização e definição do sistema criptográfico proposto, além de apresentar resultados de alguns dos testes efetuados.

No Capítulo 2 são apresentados os principais conceitos envolvendo autômatos celulares. Dentre esses conceitos são abordados a contextualização de cálculo de pré-imagens que é o principal conceito para o entendimento do sistema criptográfico aqui proposto.

O Capítulo 3 resume os principais métodos de criptografia clássica e moderna e apresenta um resumo dos modelos de criptografia para imagens já estudados. Também define a segurança de modelos criptográficos e tipos de criptoanálise abordados para a quebra da segurança.

Os principais modelos de criptografia baseados no cálculo de pré-imagens de autômatos celulares são mostrados no Capítulo 4. Algumas referências dos principais modelos utilizados para a criptografia de imagens também são apresentadas com o intuito de contextualizar com o método proposto nessa dissertação.

No Capítulo 5 uma modelagem matemática dos modelos de criptografia baseados no cálculo de pré-imagens de autômatos celulares é realizada e resultados da teoria dos grafos são utilizados para análise da segurança do método.

No Capítulo 6 o modelo 3DHCA é apresentado discutindo-se as principais variações implementadas, a fim de mostrar como o modelo final foi obtido.

O Capítulo 7 apresenta os resultados de alguns experimentos realizados com o modelo 3DHCA com o propósito de avaliar sua robustez e refinar o conjunto de chaves criptográficas que devem ser utilizadas.

Finalmente, no Capítulo 8 são relatadas as conclusões e também algumas investigações não contempladas neste trabalho que são sugeridas como continuidade dessa dissertação. Além disso, uma especificação é apresentada para a utilização do sistema criptográfico 3DHCA.

Capítulo 2

Autômatos celulares

Autômatos celulares (ACs) podem ser entendidos como um conjunto de unidades processadoras simples (células) que interagem entre si, com conectividade local, e ao longo do tempo apresentam um comportamento global. O estudo dos ACs atraiu o interesse de pesquisadores nos últimos anos, devido à sua capacidade de gerar padrões complexos, que podem ser observados na natureza, a partir da aplicação de regras simples.

Os ACs foram estudados inicialmente por Ulam e Von Neumman para projetar mecanismos artificiais de auto-reprodução [von Neumann and Burks, 1966]. Mais tarde, Jonh Conway investigou se existia um AC simples com computabilidade universal. O resultado da pesquisa foi o *Game of Life*, um AC bidimensional onde as células no estado 1 são ditas células "vivas" e as células no estado 0 são ditas células "mortas" [Berlekamp et al., 1992]. Os ACs se tornaram mais populares depois do lançamento do livro *A New Kind of Science* [Wolfram, 2002]. Um dos resultados mais importantes das pesquisas de [Wolfram, 2002] foi a estudo de comportamento de dinâmico dos ACs.

Existem diversas aplicações no uso de autômatos celulares, dentre elas, podemos citar a modelagem de fenômenos naturais, físicos ou biológicos que seriam muito difíceis de serem modelados pelas equações diferenciais, que são as mais utilizadas nesse tipo de tarefa. Alguns dos principais trabalhos conhecidos nessa linha de pesquisa foram: formação de padrões em conchas e peles de animais [Markus and Kusch, 1995], [Wolfram, 1986], [Wolfram, 2002], escoamento de fluidos [Frissh et al., 1988], propagação de incêndios em florestas [Hernandez et al., 2007], infecção do vírus da AIDS [Santos et al., 2001], evacuação emergencial [Varas et al., 2007] e formação de cristais [Wolfram, 2002].

Nesse trabalho, o foco é na aplicação de ACs como método de criptografia. As evoluções espaço-temporais mesmo utilizando-se regras simples podem dar origem a sequências de reticulados com padrões pseudo-aleatórios. Essa característica foi aproveitada na proposição de diversos métodos baseados em ACs. Nesse trabalho, utilizaremos os ACs tridimensionais para a proposição de um modelo criptográfico especialmente adequado à cifragem de imagens. O novo modelo proposto nessa dissertação é fortemente baseado nos modelos precursores propostos em [Macedo, 2007] e [Magalhaes, 2010].

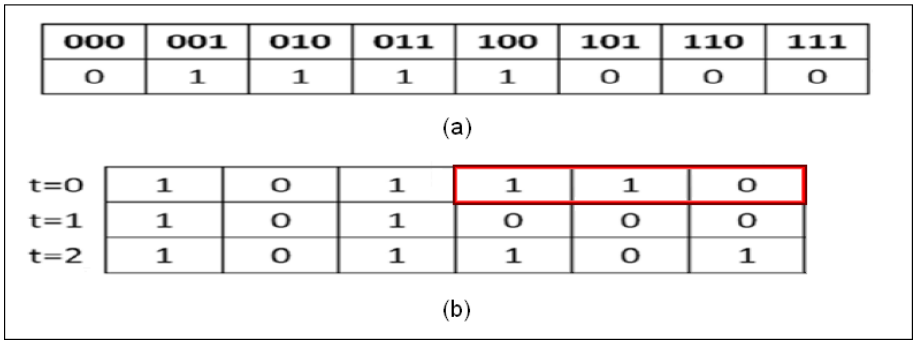
2.1 Autômatos celulares unidimensionais

Um AC é composto por um reticulado com uma dimensão d dividido em células, sendo que, cada célula contém um símbolo ou estado. As células modificam seus estados a cada passo de iteração de acordo com uma função de transição. Podemos aplicar a função de transição por T passos de tempo para obter a evolução espaço-temporal do reticulado do AC. Essa função de transição também é chamada de regra de transição. A regra de transição indica o novo símbolo a ser escrito na célula do reticulado de acordo com seu estado atual e dos estados de suas vizinhas (regra local). Em sua definição mais usual, a atualização dos estados se dá de forma síncrona, isto é, a cada passo de tempo todas as N células do reticulado são atualizadas.

A dimensão dos ACs mais estudada é a unidimensional, na qual um novo estado depende de m células vizinhas, tal que $m = (2r + 1)$ e r é o raio do AC. O reticulado de tamanho N pode ser representado, em termos computacionais, por um vetor.

A Figura 2.1 (b) apresenta um reticulado de 6 células ($N = 6$) sendo que o estado inicial de cada célula é apresentado em $t = 0$. A Figura 2.1 (a) mostra uma regra binária de raio 1 na qual a vizinhança de cada célula é formada por três elementos: a própria célula e suas duas vizinhas imediatas (à esquerda e à direita). Como esse AC é binário (2 estados possíveis), existem 8 diferentes vizinhanças, da 000 a 111. A regra em si é dada pelos 8 bits de saída associados a cada vizinhança possível: {01111000}. Na Figura 2.1 (b) vemos a atualização do reticulado por 2 passos de tempo a partir de sua configuração inicial {101110} em $t = 0$. A cada passo, cada célula do reticulado é atualizada identificando-se sua vizinhança e seu novo estado é dado pelo bit de saída correspondente na regra de transição. Por exemplo, na Figura 2.1 (b) a vizinhança da quinta célula é destacada: 110. De acordo com o bit de saída regra de transição, o estado da quinta célula deve ser atualizada para 0 ($110 \rightarrow 0$) no próximo instante de tempo. O reticulado é submetido a condições periódicas de contorno, sendo que a primeira célula é vizinha imediata da última, e vice-versa. Aplicando esse procedimento para todas as células do reticulado de forma síncrona tem-se a nova configuração do reticulado em $t = 1$. Se a regra for aplicada novamente em todas as células, obtém-se a configuração apresentada em $t = 2$.

Outra forma de apresentar a evolução espaço-temporal de um reticulado de um AC unidimensional binário é aquela em que as células no estado 0 são representadas por pixels brancos, enquanto que as células no estado 1 são representadas por pixels pretos, conforme é mostrado na Figura 2.2. Nessa figura, a regra da Figura 2.1 (a) foi aplicada em um reticulado de 16 células, partindo da configuração inicial {1011110011010110}. Observe na figura que a evolução é feita a partir da aplicação da regra no reticulado inicial, sempre inicializando essa evolução de cima para baixo. Este tipo de visualização da evolução espaço-temporal das células é largamente utilizado, pois facilita a identificação de padrões associados à dinâmica dos ACs.



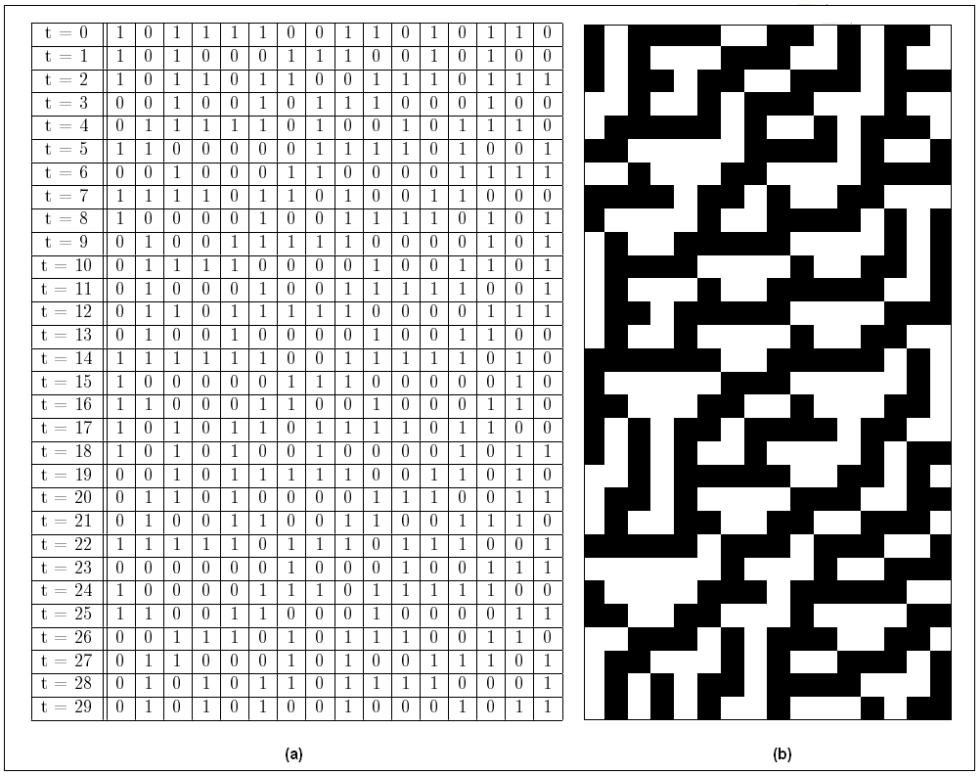
000	001	010	011	100	101	110	111
0	1	1	1	1	0	0	0

(a)

t=0	1	0	1	1	1	0
t=1	1	0	1	0	0	0
t=2	1	0	1	1	0	1

(b)

Figura 2.1: (a) Regra de transição de raio 1. (b) Reticulado unidimensional sendo evoluído por $T = 2$ passos de tempo (evolução *forward*).



t = 0	1	0	1	1	1	1	0	0	1	1	0	1	0	1	1	0
t = 1	1	0	1	0	0	0	1	1	1	0	0	1	0	1	0	0
t = 2	1	0	1	1	0	1	1	0	0	1	1	1	0	1	1	1
t = 3	0	0	1	0	0	1	0	1	1	1	0	0	0	1	0	0
t = 4	0	1	1	1	1	1	0	1	0	0	1	0	1	1	1	0
t = 5	1	1	0	0	0	0	0	1	1	1	1	0	1	0	0	1
t = 6	0	0	1	0	0	0	1	1	1	0	0	0	0	1	1	1
t = 7	1	1	1	1	0	1	1	0	1	0	0	1	1	0	0	0
t = 8	1	0	0	0	0	1	0	0	1	1	1	1	0	1	0	1
t = 9	0	1	0	0	1	1	1	1	1	0	0	0	0	1	0	1
t = 10	0	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1
t = 11	0	1	0	0	0	1	0	0	1	1	1	1	1	0	0	1
t = 12	0	1	1	0	1	1	1	1	1	0	0	0	0	1	1	1
t = 13	0	1	0	0	1	0	0	0	0	1	0	0	1	1	0	0
t = 14	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	0
t = 15	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0
t = 16	1	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0
t = 17	1	0	1	0	1	1	0	1	1	1	1	0	1	1	0	0
t = 18	1	0	1	0	1	0	0	1	0	0	0	0	1	0	1	1
t = 19	0	0	1	0	1	1	1	1	1	0	0	1	1	0	1	0
t = 20	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1
t = 21	0	1	0	0	1	1	0	0	1	1	0	0	1	1	1	0
t = 22	1	1	1	1	1	0	1	1	1	0	1	1	1	0	0	1
t = 23	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	1
t = 24	1	0	0	0	0	1	1	1	0	1	1	1	1	1	0	0
t = 25	1	1	0	0	1	1	0	0	0	1	0	0	0	0	1	1
t = 26	0	0	1	1	1	0	1	0	1	1	1	0	0	1	1	0
t = 27	0	1	1	0	0	0	1	0	1	0	0	1	1	1	0	1
t = 28	0	1	0	1	0	1	1	0	1	1	1	1	0	0	0	1
t = 29	0	1	0	1	0	1	0	0	1	0	0	0	1	0	1	1

(a) (b)

Figura 2.2: Diagrama espaço-temporal representando a evolução de um AC. (a) representação binária (b) representação por pixels preto e branco.

2.1.1 Definição formal do autômato celular unidimensional

Um AC unidimensional é definido por seu espaço celular e por sua regra de transição. O espaço celular é um conjunto de células idênticas e justapostas em um arranjo unidimensional (1D). Uma regra de transição fornece o próximo estado de cada célula. O conjunto de estados possíveis é denotado por Σ e o número de elementos desse conjunto é denotado por k. Cada célula é denotada por um índice i e seu estado, em um determinado tempo t, é denotado por S_i^t, tal que $S_i^t \in \Sigma$. O estado da célula i juntamente com o estado das outras às quais ela está conectada é denominado vizinhança da célula i, que é

denotado por η_i^t. $\Phi(\eta_i^t)$ é a regra de transição para uma dada célula i. Seja r tamanho do raio do AC 1-dimensional, η_i^t tem tamanho $m = 2r + 1$, assim, temos que Φ: $\Sigma_m \rightarrow \Sigma$. O tamanho de uma regra é dado por $n = k^m$ e o espaço de regras é dado por k^n, que representa o conjunto de todas as regras possíveis que podem ser geradas para um determinado tamanho de raio. Quanto maior for o raio de um AC maior será a quantidade de regras distintas que poderão ser geradas (cardinalidade do espaço de regras). Por exemplo, em ACs unidimensionais binários de raio 2, as regras possuem 32 bits e o espaço de regras tem cardinalidade 2^{32}.

Os autômatos celulares mais estudados, são os ACs elementares, que possuem raio 1 e alfabeto $\Sigma \in \{0, 1\}$. Dessa forma, a regra é definida por 8 bits e existem 256 regras elementares possíveis. Uma regra pode ser enumerada de forma binária, mas geralmente para ACs Elementares é comum usar a notação decimal, uma nomenclatura usual que foi proposta por Wolfram [1984]. Nessa representação, os bits de saída da regra são lidos da vizinhança 111 para a vizinhança 000 e o número binário resultante é convertido para a base 10. Por exemplo, na Figura 2.1 temos a regra 30.

A notação formal para escrever a regra na forma decimal pode ser dada através de uma fórmula que é representada pela Equação 2.1. Temos que v_i representa cada bit do conjunto $(v_0, v_1, ..., v_{m-1})$ que é uma possível vizinhança, sendo que as vizinhanças estão ordenadas lexograficamente da vizinhança 000 para a vizinhança 111. A saída para cada vizinhança é dada por b_j, tal que, $0 \leq j \leq 2^m - 1$. τ representa o valor da regra escrito na notação decimal.

$$\tau = \sum_{j=0}^{2^m-1} b_j * 2^j \tag{2.1}$$

Como exemplo, a regra elementar $\{01111000\}$ apresentada na Figura 2.1, é também chamada de Regra 30. Regras de ACs com raio maior que 1 são, geralmente, escritas na notação hexadecimal, para compactação do seu tamanho.

2.2 Autômatos celulares bidimensionais

Os ACs bidimensionais também são muito estudados na literatura de ACs [Wolfram, 2002]. O modelo mais conhecido é o jogo da vida (*Life*) que foi proposto por [Conway, 1970]. Os reticulados bidimensionais são representados no plano e em termos de estruturas computacionais em uma matriz. As vizinhanças mais conhecidas são as de Von Neumann e Moore, que podem ser vistas na Figura 2.3 (a) e Figura 2.3 (b), respectivamente. A Figura 2.4 apresenta a evolução de um AC bidimensional por 8 passos de tempo a partir da configuração inicial mostrada no instante de tempo $t = 0$, utilizando a regra do jogo da vida. A regra de transição para a evolução do jogo da vida considera a vizinhança de Moore e tem o seguinte enunciado: *"O estado da célula central será 1 (células pretas)*

no próximo instante de tempo se: o estado atual da célula é 1 e duas ou três células da vizinhança for 1 ou se a célula central é 0 e duas ou três células vizinhas forem 1. Caso contrário, a célula central assumirá valor 0 (células brancas)".

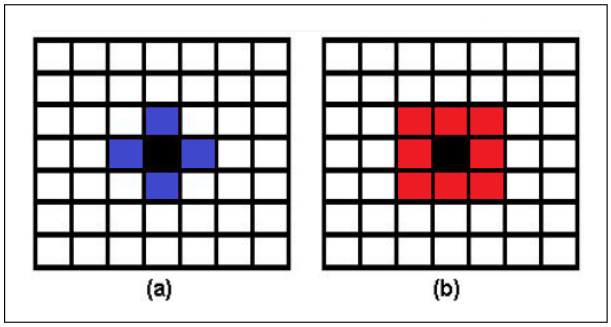


Figura 2.3: (a) Vizinhança de Von Neumann 2D. (b) Vizinhança de Moore 2D.

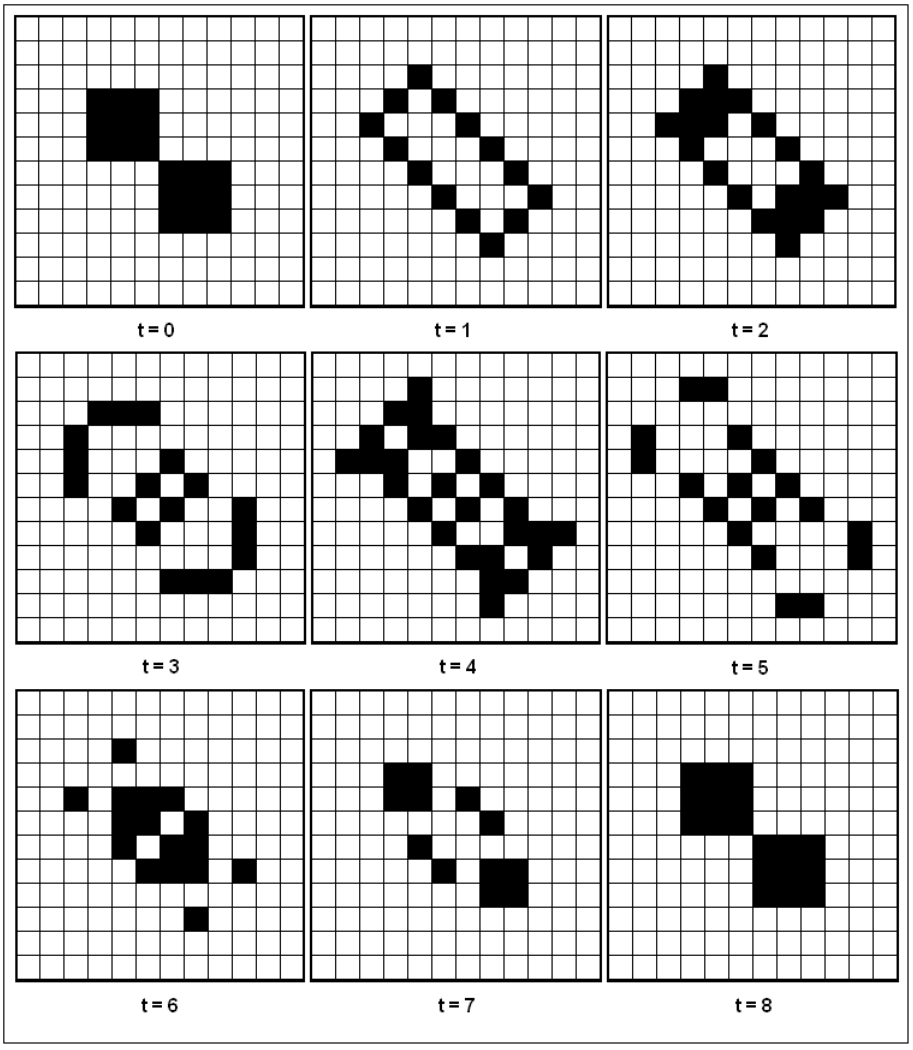


Figura 2.4: Evolução temporal do AC bidimensional por 8 passos de tempo usando a regra do *Life*.

2.3 Autômatos celulares tridimensionais

Os ACs tridimensionais que serão investigados no modelo criptográfico apresentado no Capítulo 6 utilizam um reticulado definido no espaço tridimensional e utilizam vizinhança de Von Neumann tridimensional. As vizinhanças de Moore e de Von Neumann podem ser estendidas para a terceira dimensão conforme é mostrado na Figura 2.5. O método de criptografia com ACs tridimensionais no Capítulo 6 utiliza vizinhança de Von Neumann tridimensional.

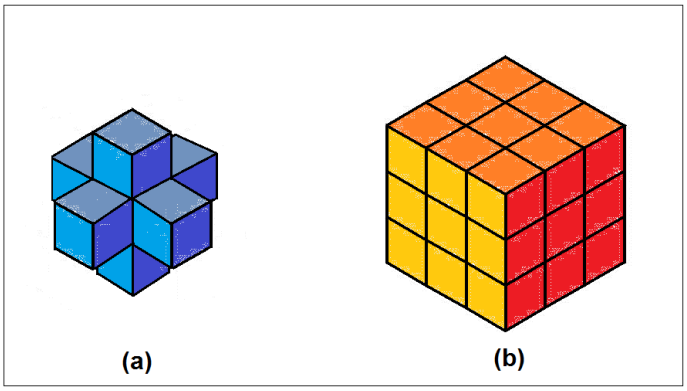


Figura 2.5: (a) Vizinhança de Von Neumann 3D. (b) Vizinhança de Moore 3D.

2.4 Classificação dinâmica

Um autômato celular pode ser classificado tendo por base seu comportamento dinâmico observado durante sua evolução temporal. Sabendo que o reticulado de um AC tem comprimento finito, então é possível determinar uma periodicidade do AC, ou seja, após aplicar a função de transição no autômato por T passos de tempo, em algum momento, nota-se que as configurações podem se repetir formando um ciclo.

Quanto mais passos de tempo forem necessários para encontrar essa repetição, maior é o ciclo da periodicidade. Dizemos periodicidade "longa", para a periodicidade que aumenta à medida que o reticulado aumenta, ou "curta" independente do tamanho do reticulado.

A classificação de um AC mais conhecida foi proposta por [Wolfram, 1984] e divide os ACs em quatro classes de acordo com a dinâmica exibida após a aplicação da regra em diversos reticulados iniciais:

- **Classe 1:** todas as células do reticulado, após um período de tempo convergem para uma configuração fixa, ou seja, para um reticulado formado somente por células no estado 0 ou reticulado formado somente por células no estado 1 (Figura 2.6 (a)).
- **Classe 2:** quase todas as células do reticulado, após um período de tempo, convergem para uma configuração de ponto fixo ou algum ciclo periódico de configurações (Figuras 2.6 (b - f).

- **Classe 3:** todas as células do reticulado, após um período de tempo, resultam em um comportamento caótico (Figura 2.6 (g)).
- **Classe 4:** todas as células do reticulado, após um período de tempo, resultam em estruturas complexas, por um espaço de tempo indefinido (Figura 2.6 (h)).

Posteriormente, pesquisadores refinaram cada uma dessas classes, sendo que um novo esquema foi proposto em [Li, 1991], que divide os ACs em seis classes distintas, conforme descritas a seguir:

- **Regras Nulas:** a configuração do reticulado após um tempo resulta em uma configuração formada tão somente por 0s ou por 1s (Figura 2.6 (a)).
- **Regras Ponto Fixo:** a configuração após um dado tempo é fixa, depois excluindo-se as configurações formadas tão somente por 0s ou por 1s, podendo apresentar um deslocamento espacial (Figuras 2.6 (b) sem deslocamento e 2.6 (c) com deslocamento espacial).
- **Regras Ciclo Duplo:** a configuração do reticulado após aplicarmos a regra do AC duas vezes é invariante, com um possível deslocamento espacial (Figuras 2.6 (d) sem deslocamento e 2.6 (e) com deslocamento espacial).
- **Regras Periódicas:** a configuração limite é invariante à aplicação da regra R (>2) vezes, com o tamanho do ciclo R ou independente ou fracamente dependente do tamanho do sistema , podendo ocorrer deslocamento espacial (Figura 2.6 (f) com deslocamento).
- **Regras Caóticas:** produzem dinâmicas não periódicas. Essas regras dependendo do tamanho do reticulado inicial apresentam uma periodicidade exponencial em relação ao tamanho desse reticulado (Figura 2.6 (g)).
- **Regras Complexas (ou Regras à Beira do Caos):** Podem produzir dinâmicas periódicas, no entanto, podem ser extremamente longas (Figura 2.6 (h)).

Para o modelo de criptografia tridimensional proposto nesse trabalho, as regras mais interessantes são aquelas que resultam em um comportamento caótico, ou seja, as que pertencem à Classe 3 [Wolfram, 1984].

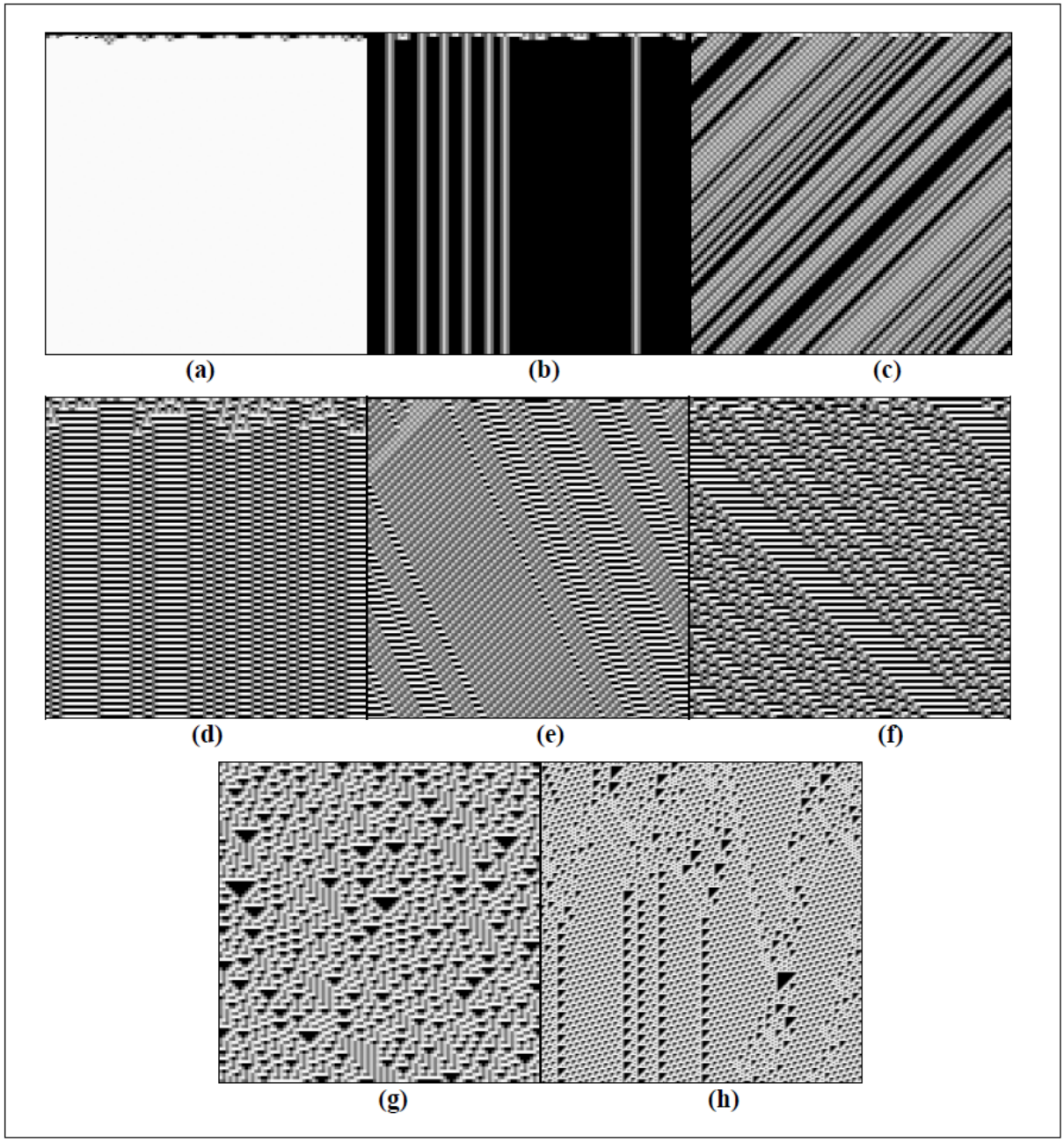


Figura 2.6: Comportamento dinâmico em ACs Elementares segundo [Li, 1991] (a) Regra 251: nula. (b) Regra 36: ponto fixo. (c) Regra 34: ponto fixo. (d) Regra 37: ciclo duplo. (e) Regra 35: ciclo duplo. (f) Regra 41: regra periódico. (g) Regra 30: caótica. (h) Regra 110: complexa [Oliveira, 2003].

2.5 Propriedade de sensitividade

As regras de ACs podem exibir propriedades específicas. Algumas dessas propriedades podem ser exploradas em criptografia, sendo uma dessas propriedades a sensitividade da regra. A sensitividade existe em uma regra quando a alteração de um bit específico da vizinhança provoca necessariamente alteração do bit de saída. No caso dos ACs unidimensionais, se a alteração for feita no extremo esquerdo da vizinhança e esta provocar

alteração nos bits de saída, temos sensitividade à esquerda. Além disso, é possível a regra ter sensitividade à direita (bit extremo direito) ou sensitividade bidirecional (bit extremo esquerdo e extremo direito ao mesmo tempo).

A Figura 2.7 apresenta exemplos de regras com sensitividade. Vemos que no caso da regra 30 os pares de vizinhança que diferem apenas no bit esquerdo (por exemplo, as vizinhanças 000 → 0 e 100 → 1) sempre levam a bits de saída complementares. Essa característica se repete para todos os pares de vizinhanças similares diferentes apenas no bit à esquerda. Por isso, essa regra é sensível à esquerda. De forma similar, a regra 85 é sensível à direita (por exemplo, 000 → 1 e 001 → 0), pois todos os pares de vizinhanças similares (diferentes apenas no bit à direita) levam a saídas complementares. No caso de regras sensíveis à direita e à esquerda temos que a sensitividade é observada nos dois extremos como na regra 90.

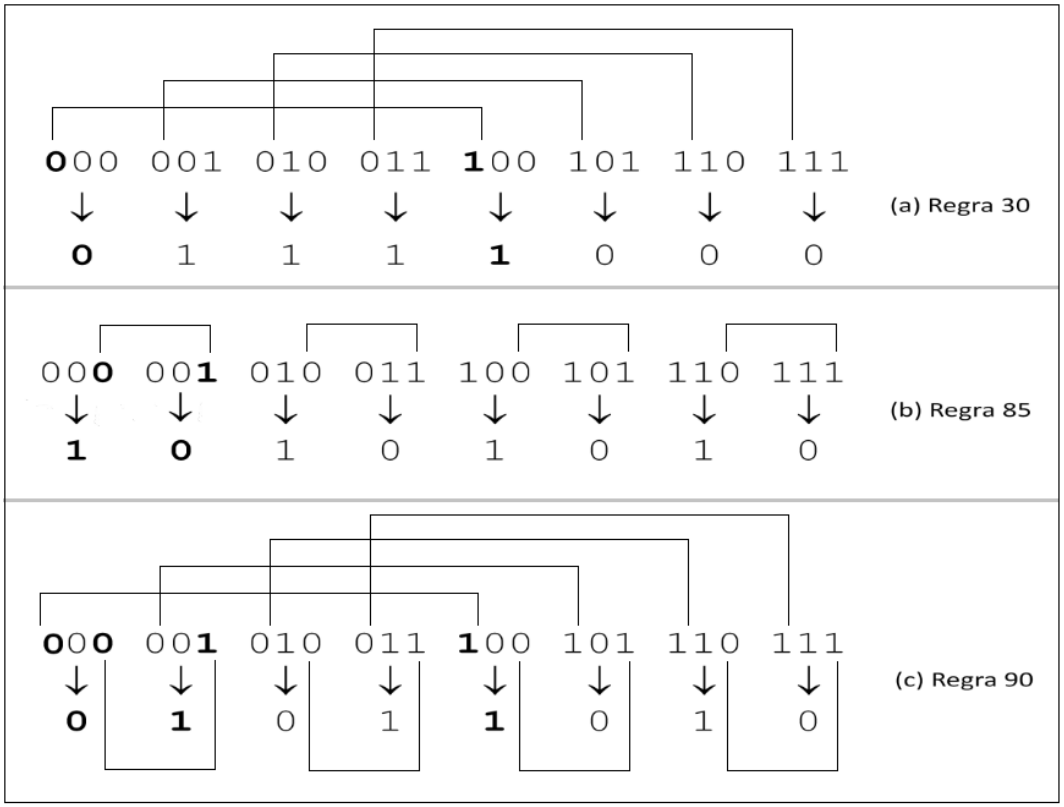


Figura 2.7: (a) Regra com sensitividade à esquerda. (b) Regra com sensitividade à direita. (c) Regra com sensitividade bidirecional.

Formalmente, podemos assim definir a sensitividade de um AC com regra de transição Φ, formada pelas vizinhanças $(v_0, v_1, \ldots, v_{m-1})$:

1. Se Φ é sensível à esquerda temos:

$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\text{Se } \Phi(v_0, v_1, \ldots, v_{m-1}) = b \text{ então } \Phi(\overline{v_0}, v_1, \ldots, v_{m-1}) = \overline{b}].$$

2. Se Φ é sensível à direita temos:

$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\text{Se } \Phi(v_0, v_1, \ldots, v_{m-1}) = b \text{ então } \Phi(v_0, v_1, \ldots, \overline{v_{m-1}}) = \overline{b}].$$

3. Se Φ é sensível à esquerda e à direita temos:

$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\text{Se } \Phi(v_0, v_1, \ldots, v_{m-1}) = b \text{ então } \Phi(\overline{v_0}, v_1, \ldots, v_{m-1}) = \overline{b} \text{ e } \Phi(v_0, v_1, \ldots, \overline{v_{m-1}}) = \overline{b}].$$

2.6 Parâmetros de previsão de comportamento dinâmico

O estudo de parâmetros estáticos é realizado com o intuito de prever o comportamento dinâmico de um AC, mas sem evoluir o reticulado por uma regra por T passos de tempo. Nesse caso, o estudo do comportamento dinâmico é realizado através de um cálculo direto nas saídas da regra de transição.

Dentre os parâmetros estudados, [Langton, 1992] propôs o Parâmetro λ que é o mais conhecido na literatura. Neste parâmetro é realizada uma somatória dos bits de saídas da regra que está em estados não quiescentes (saída igual a 1) e essa somatória é dividida pelo tamanho da regra.

Outro parâmetro importante proposto por [Wuensche and Lesser, 1992] foi o parâmetro Z. Esse parâmetro é composto por duas componentes: Z_{left} e o Z_{right}. O parâmetro Z é obtido pelo máximo entre Z_{left} e Z_{right}.

Quanto maior o valor de Z, maior é a probabilidade de a regra ser caótica [Wuensche and Lesser, 1992]. Essa característica foi aproveitada em alguns algoritmos estudados em criptografia, como veremos no Capítulo 4. Um exemplo de cálculo de Z da regra 110 é apresentado no Apêndice A.

2.7 Cálculo de pré-imagem

Uma pré-imagem de um reticulado arbitrário L_1 é um outro reticulado L_2 que quando é submetido à regra de transição resulta em L_1. Ou seja, L_2 é pré-imagem de L_1 se é um possível antecessor de L_1. O cálculo de pré-imagem é realizado a partir de uma configuração inicial do reticulado L no instante t, e tem por finalidade descobrir qual reticulado L' no instante $t-1$ pode dar origem ao reticulado L no instante t, após aplicar a regra de transição. Nesse caso, esse procedimento pode ser repetido por T passos de tempo. Essa evolução do reticulado do AC é chamada *backward* ou cálculo de pré-imagem. Alguns métodos de criptografia baseados em ACs envolvem o cálculo de pré-imagem no processo de encriptação, como no método precursor de [Gutowitz, 1995] que veremos no Capítulo 4.

Um método de cálculo de pré-imagens genérico para ACs unidimensionais e com condição de contorno periódica conhecido por Algoritmo Reverso foi proposto em [Wuensche and Lesser, 1992] para possibilitar a evolução *backward* de um AC.

No início do cálculo da pré-imagem, se o AC usa uma regra de transição de raio r ($m = 2r+1$), r células são adicionadas de cada lado do reticulado e os $m-1$ primeiros bits são inicializados aleatoriamente em uma das extremidades do reticulado. Esse aumento inicial é importante para a realização do cálculo da pré-imagem, mas após completar todas as células, esses $m-1$ bits são descartados. Dessa maneira, o reticulado retorna ao tamanho original. Esse algoritmo pode ser utilizado com qualquer tipo de regra e é baseado na ideia de vizinhanças parciais, que podem surgir ao longo do reticulado.

Suponha que a pré-imagem de um reticulado será calculada da esquerda para a direita durante. Durante esse processo, podemos encontrar três tipos de vizinhança: a determinística, a ambígua e a impossível.

Na vizinhança determinística, o bit de saída tem que ser sensível a um dos extremos da vizinhança, portanto, é sempre possível encontrar o estado da célula que estiver sendo calculada, e este estado é único. A Figura 2.8 apresenta um exemplo de cálculo de 1 bit da pré-imagem com a vizinhança do tipo determinística. Observe que os bits P_0 e P_1 foram inicializados com 00. A célula demarcada por ? é a célula que deverá ser preenchida de acordo com a regra de transição. A célula ? deve ser preenchida de acordo com a vizinhança que começa por 00? e leva a célula central para 1. A única vizinhança possível para a saída 1 é a vizinhança 001 → 1. Portanto, a célula demarcada por ? deve ser preenchida por 1.

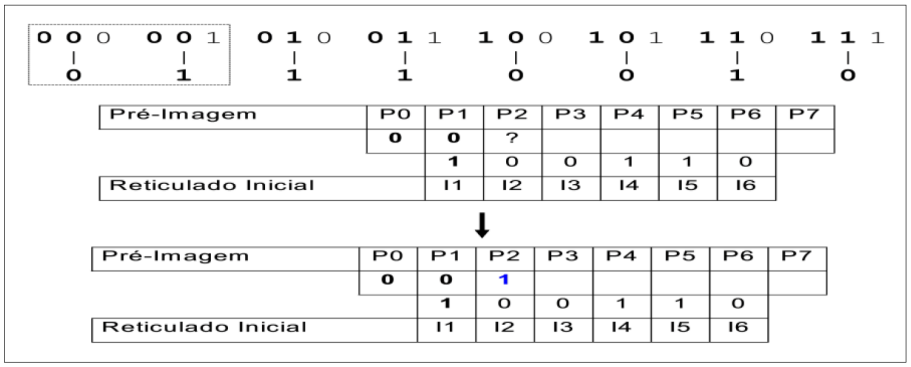


Figura 2.8: Exemplo de cálculo de pré-imagem com vizinhança determinística.

O segundo tipo é a vizinhança ambígua, na qual o bit de saída não é sensível ao extremo da vizinhança que estiver sendo analisada. Na Figura 2.9, as células P_0 e P_1 do reticulado foram inicializadas, respectivamente, com 0 e 1. A célula com o símbolo ? é a célula que deverá ser preenchida de acordo com a função de transição. Observe que a célula P_2 pode ser preenchida com dois valores de acordo com a regra de transição. Esse fato ocorre porque existem duas vizinhanças que iniciam com 01? e tem saída 1 são elas: 010 → 1 e 011 → 1. Durante o cálculo de uma pré-imagem, quando ocorre uma

vizinhança ambígua a célula mais à direita é preenchida com um dos dois valores e o outro valor é armazenado em uma pilha. Caso necessário, este bit armazenado poderá ser usado posteriormente.

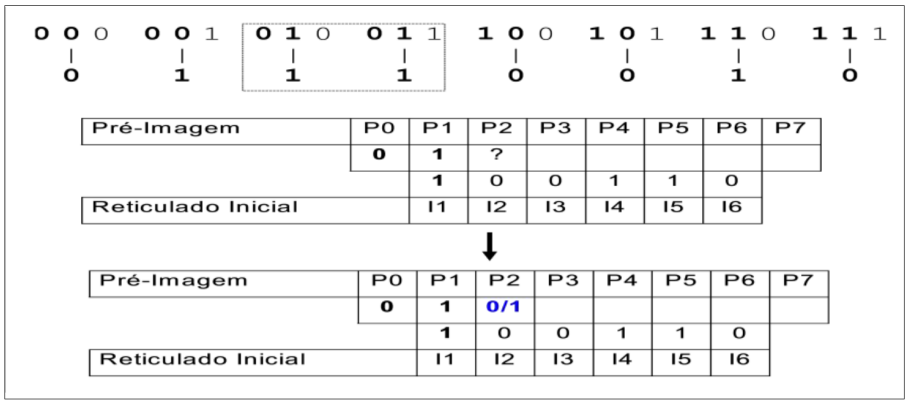


Figura 2.9: Exemplo de cálculo de pré-imagem com vizinhança ambígua.

A Figura 2.10 ilustra o caso de vizinhança parcial impossível. As células P_0 e P_1 do reticulado foram inicializadas, respectivamente, com 0 e 1. Quando a célula P_2 da pré-imagem for analisada, veremos que não existe nenhum valor possível para continuar o processamento. No caso, temos a vizinhança parcial 10? $\rightarrow$ 1 e, de acordo com a função de transição, não existe nenhuma vizinhança que começa com 01 e leva o estado da célula central para 1. Portanto, o valor de P_2 não pode ser obtido, para esses valores de P_0 e P_1. Durante o cálculo de uma pré-imagem, se ocorrer uma vizinhança impossível, o processo deve retornar à pilha e reiniciar o cálculo dos $m-1$ bits.

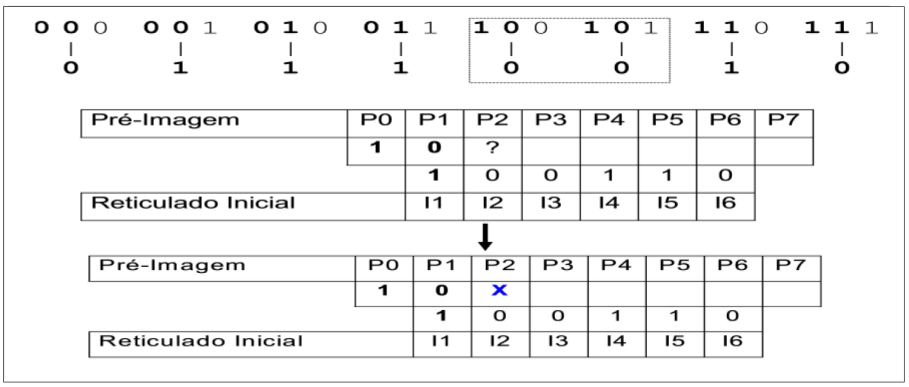


Figura 2.10: Exemplo de cálculo de pré-imagem com vizinhança impossível.

O cálculo de pré-imagem inicia com a escolha aleatória dos $m-1$ bits mais à esquerda e o cálculo sequencial Figura 2.11 (a), da esquerda para a direita, de possíveis valores para a próxima célula da direita da pré-imagem através da identificação das vizinhanças parciais. Ao chegar ao final da penúltima célula do reticulado estendido é necessário que haja uma verificação dos $m-1$ bits das extremidades direita e esquerda, Figura 2.11 (b). Como o reticulado é considerado periódico, é necessário que os $m-1$ bits finais

sejam iguais aos $m-1$ bits iniciais. Caso sejam iguais, a pré-imagem é validada e os bits extras podem ser descartados, uma vez que o reticulado possui contorno periódico, como na Figura 2.11 (c). Senão, o processo retorna até a pilha, verificando se existe alguma opção que pode ser utilizada. Se não existir nenhuma opção, ele reinicializa o reticulado com outra configuração para $m-1$ bits iniciais. O processo continua até encontrar todas as pré-imagens possíveis. Se ele realizar todas as inicializações possíveis e não conseguir finalizar/validar o cálculo em nenhuma delas, então, não existe nenhuma pré-imagem para o dado reticulado e a regra especificada. Esse reticulado é conhecido por jardim-do-Éden e só pode surgir em uma evolução temporal como configuração inicial.

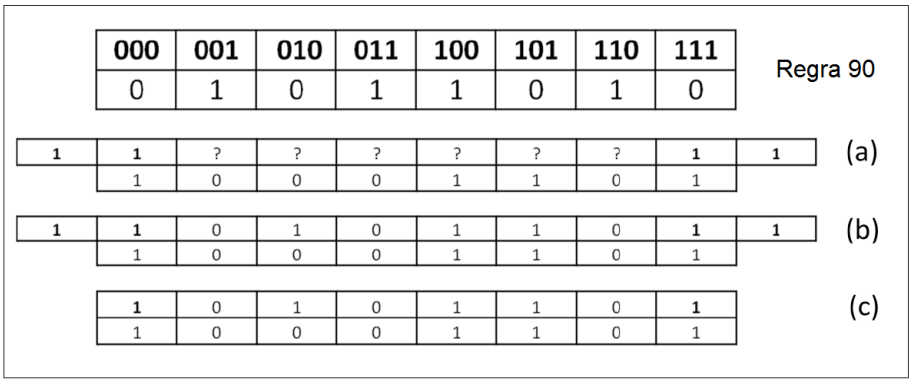


Figura 2.11: (a) Inicialização (b) Validação (c) Descarte.

2.8 Variações do autômato celular padrão

A construção dos modelos de ACs possuem algumas variações que se diferencia do modelo padrão apresentado. As principais variações que são interessantes para o entendimento do presente trabalho serão apresentadas a seguir.

A condição de contorno determina como as células da vizinhança nos extremos de um reticulado serão preenchidas. A vizinhança das células do extremo do reticulado pode ser estabelecida de diferentes formas, sendo a mais conhecida chamada de condição de contorno periódico que foi empregado na Figura 2.1. Nessa figura, temos que a primeira célula do reticulado é vizinha da última e a última célula é vizinha da primeira. Outra condição de contorno que pode ser adotada na atualização das células é chamada de condição de contorno nula, na qual a última e a primeira células do reticulado possuem células no estado zero como vizinhas.

A atualização das células do reticulado também pode ser efetuada de diferentes maneiras. Existem dois tipos de atualização de células: síncrona e sequencial. A atualização síncrona das células ocorre quando todas as células do reticulado são atualizadas ao mesmo tempo (de forma paralela), sendo esta a mais usual. A atualização pode ser realizada também de forma sequencial, na qual a atualização de uma célula depende da atualização prévia do estado da sua célula antecessora no reticulado unidimensional.

Para a evolução espaço-temporal do um AC é usual que todas as células sejam atualizadas de acordo com a mesma função de transição. Quando isso ocorre, dizemos que o autômato celular é homogêneo. Por outro lado, quando atualizamos as células de um reticulado com dois ou mais tipos de regras dizemos que o AC é heterogêneo ou híbrido. Por exemplo, em um reticulado de tamanho N, podemos definir N regras de transição diferentes, uma para cada célula do reticulado. No modelo de criptografia proposto aqui, duas regras diferentes são aplicadas ao longo do reticulado.

A dimensão de um AC também pode variar, como observado nas subseções anteriores. Um AC pode ter K dimensões, tal que $K > 0$. A estrutura do reticulado pode ser modificada ao aumentar ou reduzir a dimensão. A estrutura utilizada no presente trabalho é a dimensão 3D.

2.9 Considerações em relação ao método proposto

Nesse capitulo foram apresentadas as principais definições sobre autômatos celulares. Em relação às definições apresentadas, o método de criptografia discutido no Capítulo 6 apresenta as seguintes características: (i) a dimensão do reticulado empregado que é tridimensional, (ii) o comportamento dinâmico das regras é caótico, (iii) o AC utilizado é heterogêneo empregando duas regras de atualização, (iv) as regras devem obedecer à propriedade de sensitividade, (v) o reticulado possui condição de contorno periódica e (iv) as células devem ser atualizadas de forma de forma síncrona. O processo de cifragem equivale à evolução do AC para trás (*backward*), ou seja, o cálculo de T pré-imagens consecutivos. O processo de decifragem é feito pela evolução temporal do AC para frente (*forward*) pelo mesmo número de passos T.

Capítulo 3

Criptografia e segurança

A palavra criptografia tem sua origem no grego: *kryptos* significa oculto, envolto, escondido, secreto; *graphos* significa escrever, grafar. Portanto, criptografia significa escrita secreta ou ainda pode ser entendida como o estudo de técnicas que transformam uma mensagem para uma forma ilegível, de tal forma que nenhum oponente consiga compreender a mensagem que estiver sendo enviada. Dessa forma, só o receptor da mensagem poderá ler a informação facilmente.

Neste capítulo, será apresentada uma introdução à criptografia de dados. O principais conceitos sobre criptografia, a terminologia utilizada bem como alguns sistemas mais importantes serão detalhados a seguir.

3.1 Histórico

A criptografia tem suas origens milhares de anos atrás e remonta aos tempos da antiga Grécia (480 AC). Durante a Batalha das Termópilas, segundo a lenda, a cidade de Esparta teria sido avisada da invasão dos inimigos por uma mensagem cifrada em argila. O sistema passou a ser adotado durante outras guerras, como ocorre até hoje, mas migrou para outros setores. Os primeiros modelos de criptografia eram construídos utilizando simples recursos mecânicos. No início do século 20, o advento de novas máquinas mecânicas e eletromecânicas possibilitaram meios mais sofisticados para a criptografia. O matemático inglês Alan Turing (1912-1954), considerado o precursor da ciência da computação, foi o principal responsável pelo aperfeiçoamento da criptografia no século 20. O cientista ficou famoso ao conseguir decifrar o sistema conhecido por *Enigma*, utilizado pelos nazistas na Segunda Guerra Mundial para a transmissão de mensagens. A descoberta contribuiu para a vitória dos Aliados no confronto. Anos depois, com o advento de computadores, foi possível a criação de sistemas criptográficos com complexidade ainda maiores. Atualmente, além da clássica utilização para fins militares, a criptografia tem um importante papel na era digital. Sem a utilização desses sistemas criptográficos, seria impossível que as transações comerciais ou bancárias realizadas na internet de forma segura.

O desenvolvimento dos sistemas de cifragem ocorreu paralelamente ao desenvolvimento da criptoanálise, que é a área que investiga a "quebra" de cifras e códigos. A partir dos anos 70, a criptografia que antes era largamente utilizada para preservar o governo, passou a ser amplamente utilizada por grandes empresas do setor de computação que passaram a desenvolver seus próprios sistemas criptográficos. Um criptosistema muito importante dessa época foi o DES (*Data Encryption Standard*) [Stinson, 2006]. A concorrência entre as companhias se intensificou no início dos anos 90, e um marco na história da criptografia foi a invenção do sistema de criptografia de chave pública, como o algoritmo RSA [Rivest et al., 1978]

3.2 Terminologia

Em criptografia, considera-se a necessidade de transmitir uma mensagem x, tal que, $x \in \mathcal{P}$, entre uma entidade emissora (Alice) da informação e uma entidade receptora (Bob). A mensagem também é denominada por texto plano ou texto claro. Esta designação corresponderá a qualquer sequência de bits que se pretenda transmitir em segurança pois, não necessariamente a informação a ser transmitida é um texto. O processo de ocultar a mensagem chama-se cifragem e transforma o texto claro numa mensagem criptada y ou criptograma, tal que, $y \in \mathcal{C}$. Esse processo dificulta que um intruso (Oscar) descubra a mensagem que é enviada. O processo de recuperar o texto plano original a partir do criptograma denomina-se decifragem.

A Figura 3.1 [Stinson, 2006] ilustra o processo de cifragem de uma mensagem. Suponha que o texto plano x é cifrado utilizando um processo e_k e uma chave k e o texto cifrado é decriptado utilizando a mesma chave k e um processo d_k, assim o texto plano é obtido novamente.

Alice e Bob vão escolher uma chave $k \in \mathcal{K}$. Isso é feito quando eles não estiverem sendo observados por Oscar, ou seja, quando for possível acessar um canal seguro. Em seguida, Alice deseja transmitir uma mensagem para Bob. Vamos supor que a mensagem seja o string $x = x_1, x_2, \cdots, x_n$ para algum $n \geq 1$, onde cada $x \in \mathcal{P}$, $1 \leq i \leq n$. Assim, x é encriptado usando o princípio de cifragem e_k especificados pela chave de cifragem $\mathcal{K}$. Posteriormente, Alice computa $y_i = e_k(x_i)$, $1 \leq i \leq n$, e o texto resultante é a string $y = y_1, y_2, \cdots, y_n$ que é enviada pelo canal inseguro. Quando Bob recebe a string cifrada, ele decripta utilizando a função d_k obtendo o texto plano novamente.

Basicamente, o criptosistema é uma 5-tupla $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \varepsilon, \mathcal{D})$ que satisfaz as seguintes condições:

- $\mathcal{P}$ é um conjunto de textos planos possíveis.
- $\mathcal{C}$ é um conjunto de textos cifrados.

- $\mathcal{K}$ é um conjunto de chaves possíveis.
- Para cada $k \in \mathcal{K}$, existe uma regra que cifra $e_k \in \varepsilon$ e uma regra que decifra $d_k \in \mathcal{D}$. Cada $e_k : \mathcal{P} \to \mathcal{C}$ e $d_k : \mathcal{C} \to \mathcal{P}$ são funções tais que $d_k(e_k(x)) = x$ para todo elemento de texto plano $x \in \mathcal{P}$.

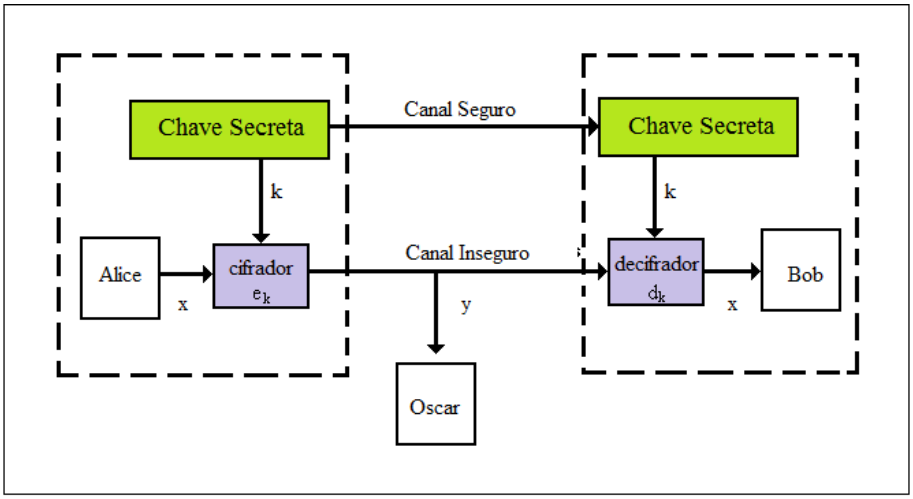


Figura 3.1: Canal de comunicação.

3.3 Métodos clássicos

Os modelos clássicos eram utilizados antes mesmo do surgimento dos computadores, esse conjunto de métodos de criptografia é também chamado de criptografia pré-computacional. O conjunto de modelos criptográficos pré-computacionais poderiam ser calculados manualmente pelo emissor e pelo destinatário da mensagem. Os principais modelos clássicos encontrados na literatura utilizam algum método de substituição ou transposição. No Apêndice B serão apresentados os principais modelos clássicos de criptografia.

3.4 Criptografia moderna

Depois das contribuições das publicações de alguns artigos de [Shannon, 1948], responsável por fundamentar um novo campo conhecido como a Teoria da Informação, e com o aumento da força computacional, vários algoritmos de criptografia foram desenvolvidos tirando vantagens de ambos [Bruen and Forcinito, 2004]. A partir desse momento surgiram trabalhos em duas novas vertentes: a criptografia simétrica e a criptografia assimétrica.

3.4.1 Criptografia simétrica

Os algoritmos de chave simétrica pertencem a uma classe de algoritmos para a criptografia que utilizam a mesma chave para cifrar e decifrar o texto. As chaves podem ser idênticas ou podem sofrer uma simples transformação entre as duas chaves. As chaves, na prática, são trocadas entre as entidades comunicantes através de um canal seguro.

Os sistemas de criptografia simétrica podem usar cifras baseadas em *streams* onde os bits da mensagem são cifrados ao longo do tempo. Ou ainda pode-se utilizar criptografia baseada por blocos (*block cipher*), onde um conjunto de bits são cifrados como uma unidade.

Dentre os algoritmos mais conhecidos e utilizados podemos destacar o DES [Stinson, 2006] e o AES [Daemen and Rijmen, 1998]. No Apêndice C, os principais algoritmos de criptografia simétrica serão apresentados com mais detalhes.

3.4.2 Criptografia assimétrica

Modelos de criptografia assimétricos possuem a característica de que o par de chaves que compõe o modelo é composto de uma chave pública e uma chave privada. Nesse modelo, a chave pública, como o próprio nome sugere, está em um diretório e é conhecida por todos, enquanto que a chave privada é conhecida apenas pelo remetente. Dessa forma, ao se utilizar a chave privada para cifrar as mensagens, há garantia de que apenas o proprietário da chave privada poderia ter cifrado a mensagem que foi decifrada com a chave pública, caracterizando assim a autenticidade. A confidencialidade é garantida quando para cifrar as mensagens é utilizada a chave pública, assim, apenas o proprietário da chave privada é capaz de decifrar as mensagens que são enviadas. O Apêndice C apresenta o principal método de criptografia assimétrica: RSA [Rivest et al., 1978]. Discorreremos a seguir sobre alguns aspectos de segurança do método RSA, pois eles serviram de base para a análise de um método baseado em ACs no Capítulo 5.

O criptosistema RSA pode ser descrito utilizando computações em $\mathbb{Z}_n$, onde n é o produto de dois primos ímpares p e q. Para esse inteiro n temos a função tociente de Euler, representada por ϕ, é definida para um número natural n como sendo igual à quantidade de números menores que n co-primos com respeito a ele. Assim, temos que, $\phi(n) = (p-1)(q-1)$. O criptosistema é descrito a seguir:

Seja $n = p \times q$, onde p e q são primos. Seja $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$, define-se:

$$\mathcal{K} = (n, p, q, a, b)\text{: } ab \equiv (1 \bmod \phi(n)) \text{ e } \gcd(b, \phi(n)) = 1.$$

ou seja, devem ser escolhidos dois inteiros a e b de forma que b e $\phi(n)$ sejam co-primos, isto é, o máximo divisor comum entre b e $\phi(n)$ deve ser igual a 1.

Uma vez definidos os parâmetros n, p, q, a, b que definem a chave $k = (n, p, q, a, b)$, o processo de cifragem e_k e de decifragem d_k é dado por:

$$e_K(x) = x^b \bmod n$$

e

$$d_K(y) = y^a \bmod n$$

De tal forma que $(x, y \in \mathbb{Z}_n)$. Os valores n e b compreendem a chave pública e os valores p, q e a formam a chave privada. Um exemplo prático e um algoritmo que define o esquema de geração dessas chaves são mostrados no Apêndice C.

Sabe-se que a segurança do RSA está baseada na dificuldade de dois problemas matemáticos: a fatoração inteiros grandes e o problema RSA. Este último pode ser definido como o problema de extrair a b-ésima raiz módulo de um inteiro composto n. Em outras palavras, dados inteiros b, n e $x^b \equiv y \pmod{n}$ como se faz para deduzir o valor de x. Atualmente, a melhor forma de resolver o problema RSA é fatorar o inteiro n. Se um atacante conseguir fatorar $n = pq$, então poderá calcular facilmente o valor de $\phi(n)$ e assim o valor de $d_K(y) = y^a \bmod n$, descobrindo a chave privada. Até o momento, não se conhece um algoritmo para fatoração de inteiros grandes, em um computador clássico, que funcione em tempo polinomial. Nem foi provado que um algoritmo deste tipo possa existir.

Matematicamente, dizemos que o RSA é seguro se para todo $y \in \mathcal{C}$ for difícil de computar x,

$$x^b \equiv y \pmod{n}$$

Pela definição, o RSA não é seguro, quando $1^b \equiv y \pmod{n}$. Atualmente, não é possível assegurar que um código com chaves de comprimento moderado é realmente seguro. Para garantir que um código não possa ser quebrado rapidamente, uma prova matemática mostrando que não é possível encontrar a chave facilmente deve ser apresentado. Ou seja, basta mostrar que precisa-se de pelo menos T passos para resolver o problema. Isso pode ser mostrado a partir da complexidade de tempo $\mathcal{T}(n)$ ou a complexidade de espaço $\mathcal{S}(n)$. Uma vez que uma chave é descoberta, verificar sua corretude pode ser realizado inspecionando-se as mensagens que foram decriptadas com ela.

3.5 Teoria da complexidade e segurança de métodos criptográficos

A Teoria da Complexidade provê uma outra maneira de ganhar evidência para a segurança de um código. Podemos mostrar que a complexidade de se quebrar um código está ligada à complexidade de algum outro problema para o qual se evidencia tal intratabilidade. A NP-Completude é usada para fornecer evidência de que os problemas são

intratáveis. No entanto, reduzir um problema NP-completo ao problema de se quebrar o código mostraria que o problema de se quebrar o código era em si próprio NP-completo. Um problema pode ser NP-completo, e mesmo assim fácil de resolver a maior parte do tempo [Sipser, 1996]. Como exemplo, sabe-se que Merkle-Hellman foi o primeiro algoritmo assimétrico divulgado [Shamir, 1982]. Esse criptosistema baseia-se no problema combinatório de se encontrar uma partição em um conjunto fixo de inteiros onde a soma de elementos dê igual ao argumento. *Merkle-Hellman Knapsack Cryptosystem* é inseguro, apesar de NP-completo, o problema no qual é baseado - a chave privada pode ser obtida em tempo polinomial a partir da pública - o torna vulnerável à ataques. Por outro lado, acredita-se que o problema da fatoração inteira seja difícil para o caso médio. O RSA foi construído em torno do problema da fatoração de modo que quebrar o código corresponde a fatorar um número. Isso constitui evidência convincente para a segurança do RSA, porque uma maneira eficiente de se quebrar tal código levaria a um algoritmo de fatoração rápido, o que seria um desenvolvimento marcante em teoria computacional dos números. Mostrar que o criptosistema RSA (C) é seguro deve-se:

1. Selecionar um problema P que é difícil de ser resolvido
2. Reduzir o problema P à segurança do criptosistema C

Uma vez que P é difícil de ser resolvido, o criptosistema C é difícil de ser quebrado. Ou seja, se for possível reduzir o problema P ao problema C usando uma função computável polinomial que converte instâncias do problema P para instâncias (I) do problema C. Se tivermos tal função de conversão, podemos resolver P com um solucionador para C. A razão é que qualquer instância de P pode ser resolvida primeiro usando a redução para converte-la para uma instancia de C e a então aplicando o solucionador para C. Dizemos $P \leq_p C$, isso significa que P pode ser reduzido polinomialmente para o problema C se:

1. Dado uma instância I_P de P é possível construir uma instância I_C de C tal que a solução de C (I_C) pode ser convertida na solução de P (I_P).
2. Ambas as construções da instâncias e a conversão das soluções devem ser feitas em tempo polinomial.

Essa redução é chamada de muitos para um. Se $P \leq_p C$, então o problema C é mais difícil de ser resolvido que o problema de P. Dizemos que $P \equiv_p C$ se $P \leq_p C$ e $C \leq_p P$.

Para mostrar que o RSA é seguro, vamos considerar $n = pq$, onde p e q são dois primos distintos. Sabemos que o criptosistema RSA consiste em calcular $x^b \equiv y \pmod{n}$. Para mostrar a segurança do RSA vamos fazer a seguinte redução considerando o problema P que consiste na fatoração de n e um problema C' que corresponde à resolver $x^2 \equiv e \bmod n$, tal que $C' \leq C$. E devemos provar que $P \equiv C'$. Para tal, primeiramente deve-se mostrar que $P \leq_p C'$, ou seja, deve-se assumir que C' pode ser resolvido em tempo polinomial.

Então, deve-se construir um algoritmo probabilístico de tempo polinomial que fatore n. O Algoritmo 1 é mostrado a seguir e o mesmo tem como entrada um inteiro n e como saída um fator desse n.

Algoritmo 1 $P \leq_p C'$

1. Randomicamente selecione um inteiro y.
2. Se $d = \gcd(y,n) > 1$ então d é um fator de n. Pare.
3. Compute $e = y^2 \bmod n$.
4. Resolva $x^2 \equiv e \pmod{n}$ para x usando o Algoritmo 2.
5. Se $x = \pm y$ então rejeite. Caso contrário, $\gcd(x = y, n)$ é um fator de n.

Para mostrar a outra direção, devemos assumir que C' pode ser resolvido em tempo polinomial, então devemos mostrar um algoritmo que resolva $x^2 \equiv e \pmod{n}$ para x em tempo polinomial. Como entrada esse Algoritmo 2 tem dois inteiros e e n e como saída o algoritmo retorna x e $x^2 \equiv e \pmod{n}$.

Algoritmo 2 $C' \leq_p P$

1. Fatore $n = pq$ usando o Algoritmo 1.
2. Resolva $x_p^2 \equiv e \pmod{p}$ para x_p.
3. Resolva $x_q^2 \equiv e \pmod{q}$ para x_q.
4. Compute x tal que $x \bmod p = x_p$ e $x \bmod q = x_q$ usando o teorema do resto chinês.

Como foi possível mostrar que $P \leq_p C'$, $C' \leq_p P$ e $C' \leq C$, onde C representa o criptosistema RSA. O problema da fatoração de um inteiro NP-completo no caso médio foi reduzido ao problema de se quebrar o código do RSA, então o problema de se quebrar o código é em si próprio NP-completo. Como não se conhece um algoritmo para fatoração de inteiros grandes, em um computador clássico, que funcione em tempo polinomial, nem foi provado que um algoritmo deste tipo possa existir. Portanto, para chaves suficientemente grandes, o RSA é seguro, levando-se em conta o conhecimento matemático atual do problema da fatoração de inteiros grandes.

3.6 Criptoanálise

A criptoanálise tem como principal objetivo descobrir o texto cifrado e/ou a lógica empregada dos métodos criptográficos. As pessoas que praticam a criptoanálise são denominadas criptoanalistas.

A criptoanálise representa o esforço de decodificar ou decifrar mensagens, mas isso não significa que o criptoanalista conseguirá acessar o texto cifrado por completo, pode

ser apenas partes dele. Existem várias técnicas para atingir tal objetivo. Essas técnicas levam em consideração o esquema criptográfico e a quantidade de informação.

Existem diversos tipos de ataques que podem ser aplicados aos sistemas criptográficos, esses ataques podem ser classificados de acordo com a disponibilidade de informações que o criptoanalista consegue obter, conforme pode ser observado na Tabela 3.1 [Conrad et al., 2012].

Tabela 3.1: Principais tipos de ataques a sistemas criptográficos.

Informação	**Tipo de Ataque**		
Texto Claro	Texto claro conhecido	Texto claro escolhido	Texto claro escolhido adaptável
Texto Cifrado	Apenas texto cifrado	Texto cifrado escolhido	Texto cifrado escolhido adaptável

Texto claro conhecido

É um tipo de ataque no qual o criptoanalista tem acesso ao texto claro e o seu texto cifrado correspondente e procura uma correlação entre os dois.

Apenas texto cifrado

É um tipo de ataque no qual o criptoanalista tem acesso ao texto cifrado, mas não tem acesso ao texto claro. Em métodos criptográficos simples como a *Cifra de Cézar*, através de uma análise de frequência é possível quebrar o sistema criptográfico.

Texto claro escolhido

É um tipo de ataque no qual o criptoanalista pode cifrar um texto claro de sua escolha e estudar o texto cifrado resultante. Este é o ataque mais conhecido contra sistemas criptográficos assimétricos no qual o criptoanalista tem acesso à chave pública.

Texto cifrado escolhido

É um tipo de ataque no qual o criptoanalista escolhe um texto cifrado e tenta encontrar um texto claro correspondente. Isto pode ser feito através de uma máquina que decifra sem expor a chave. Este modelo também é fortemente utilizado em modelos de chave pública.

Texto escolhido adaptável

Este é um caso especial do ataque de texto claro escolhido. O criptoanalista não só pode escolher o texto claro a ser cifrado, mas pode ainda modificar sua escolha baseado nos resultados da encriptação anterior. Num ataque de texto claro escolhido, ele pode estar limitado à escolha de um único bloco de texto claro a ser cifrado.

Num ataque adaptativo de texto claro escolhido, ele pode escolher um bloco de texto pleno a partir do anterior, e assim por diante. Alguns métodos de criptografia como o RSA são muito vulneráveis a este ataque.

Um método que pode ser utilizado na criptoanálise é o ataque por força bruta (*brute force attack*). O ataque nesse método tem por objetivo testar todo espaço de chaves possíveis. Outros dois métodos de criptoanálise que merecem destaque são a criptoanálise diferencial e a criptoanálise linear.

3.6.1 Criptoanálise diferencial

A criptoanálise diferencial é um ataque do tipo texto claro escolhido. A principal aplicação dessa criptoanálise ocorre em métodos de cifragem por blocos. A criptoanálise diferencial estuda como as diferenças provocadas na entrada podem alterar a saída.

A ideia básica do método é utilizar pares de texto claro relacionados por uma diferença constante. A diferença pode ser definida de várias maneiras, a principal ferramenta utilizada nessa técnica é a operação binária XOR. O criptoanalista espera encontrar padrões estatísticos em sua distribuição.

3.6.2 Criptoanálise linear

A criptoanálise linear é uma forma genérica de criptoanálise, portanto a mais largamente utilizada para cifragem em blocos. O ataque dessa técnica é do tipo texto claro conhecido e foi proposta em [Matsui, 1994] a fim de quebrar o algoritmo de criptografia DES. O método tenta atacar a fraqueza do método apresentar alta probabilidade de ocorrência de uma expressão linear entre os bits do texto plano ($\mathcal{P}$), texto cifrado ($\mathcal{C}$) e a chave criptográfica (k).

Para alcançar o objetivo acima, [Matsui, 1994] elaborou um modelo estatístico linear relacionando entradas e saídas das *S-box* do DES, que em seguida é estendido para todo o algoritmo obtendo uma expressão linear não dependente de valores intermediários, que está representada na Equação 3.1 abaixo.

$$x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_u} \oplus y_{j_1} \oplus y_{j_2} \oplus \cdots \oplus y_{j_v} \tag{3.1}$$

De tal forma que x_i representa o i-ésimo bit da string de entrada $x = x_1, x_2, \cdots, x_u$ e y_j representa o j-ésimo bit da string de saída $y = y_1, y_2, \cdots, y_v$. Esta equação indica a soma do operador XOR entre os u bits da string de entrada e os v bits da string de saída.

Se o sistema criptográfico apresentar uma tendência para Equação 3.1 ocorrer com alta ou baixa probabilidade, este sistema possui grande dificuldade em tornar o texto cifrado aleatório, tornando-se vulnerável ao ataque.

3.7 Criptografia para imagens

Os algoritmos de criptografia para imagens devem contemplar as características inerentes às mesmas. Grande quantidade de dados e grande redundância das informações, são exemplos característicos das imagens. A proposição de métodos específicos para a realização da cifragem de imagens tornou-se muito importante, visto que eles tentam otimizar os aspectos não mapeados nos modelos usuais. Um breve comentário sobre alguns algoritmos de criptografia para imagens utilizando os modelos clássicos e/ou modernos ou baseado em alguma transformação desses modelos será apresentada.

Em [Zeghid et al., 2007] utilizou-se o modelo tradicional AES para adequá-lo a cifragem de imagens. Em [Socek et al., 2005], um aperfeiçoamento do CKBA (*Chaotic-Key Based Algorithm*) foi realizado. O CKBA é um algoritmo que possui um mapa caótico unidimensional, esse mapa é utilizado para realizar a cifragem. Embora alguns parâmetros como tamanho da chave, número de etapas tenham sido alterados, o modelo proposto em [Socek et al., 2005] ainda não propaga a criptografia espacialmente. Na referencia, [Chen et al., 2004], é apresentado um modelo no qual a criptografia se baseia em mapas caóticos tridimensionais. Esse método é interessante devido ao fato de que o mapeamento consegue ter uma boa propriedade de difusão ao longo de toda imagem cifrada.

3.8 Considerações em relação ao modelo de criptografia proposto

Os modelos tradicionais, bem como as definições apresentadas nesse capítulo, serão importantes para contextualizar e posteriormente comparar com o algoritmo de criptografia proposto na presente dissertação de mestrado. Um dos principais contrastes que foi encontrado é o fato de que grande parte dos algoritmos de criptografia utilizam de alguma forma operações XOR para realizar a cifragem do texto. Por outro lado, do algoritmo aqui proposto que utiliza a computação dos ACs para realização dessa cifragem. Os algoritmos que foram referenciados nesse capítulo não possuem alto grau de paralelismo, diferentemente do algoritmo proposto na dissertação. Apesar do processo de cifragem/decifragem não se assemelhar aos demais métodos tradicionais, o modelo deste trabalho pertence à classe dos algoritmos de chave simétrica que realizam cifragem por bloco. A criptoanálise diferencial contribuiu para a realização de testes a fim de verificar a validade e a segurança do modelo aqui proposto.

Capítulo 4

Criptografia baseada em autômatos celulares

Neste capítulo serão apresentados alguns dos principais métodos criptográficos baseados em ACs que já foram investigados na literatura. O primeiro trabalho conhecido envolvendo o uso de ACs para a realização da criptografia foi proposto por [Wolfram, 1986]. Nesse modelo precursor, a ideia principal é utilizar um autômato celular binário com um reticulado inicial de tamanho N e com condições de contorno periódica. A regra de transição é fixa e apresenta dinâmica caótica, sendo que uma chave binária de tamanho N é utilizada como reticulado inicial a partir do qual a regra é aplicada por um número fixo de passos.

Os valores do reticulado são atualizados de forma síncrona, em passos de tempo discretos de acordo com a regra elementar 30. [Wolfram, 1986] mostrou através de um estudo estatístico, que os estados que uma célula i assume ao longo do tempo (s_i^t), formam uma sequência binária pseudo-aleatória. Essa sequência pode ser aplicada bit a bit a um texto plano $x \in \mathcal{P}$ com a operação XOR, produzindo o texto cifrado $y \in \mathcal{C}$. O texto plano a ser cifrado é uma string de tamanho L e a sequência temporal da célula na posição i é utilizada para gerar a sequências s_i^t, para $1 \leq t \leq L$. A Equação 4.1 indica a operação que deve ser realizada entre o texto plano e a chave.

$$y = x \oplus s_i^t \tag{4.1}$$

A Figura 4.1 mostra um AC com reticulado de tamanho 16 evoluído por 15 passos de tempo com a regra 30. Os bits representados pela célula 7 ($i = 7$) estão em destaque indicando que esta é uma sequência que pode ser aplicada usando a função XOR e o texto plano $x \in \mathcal{P}$.

A força deste método se baseia na dificuldade do criptoanalista encontrar o reticulado inicial que resulte na sequência de bits usadas para cifrar o texto plano. Este método não

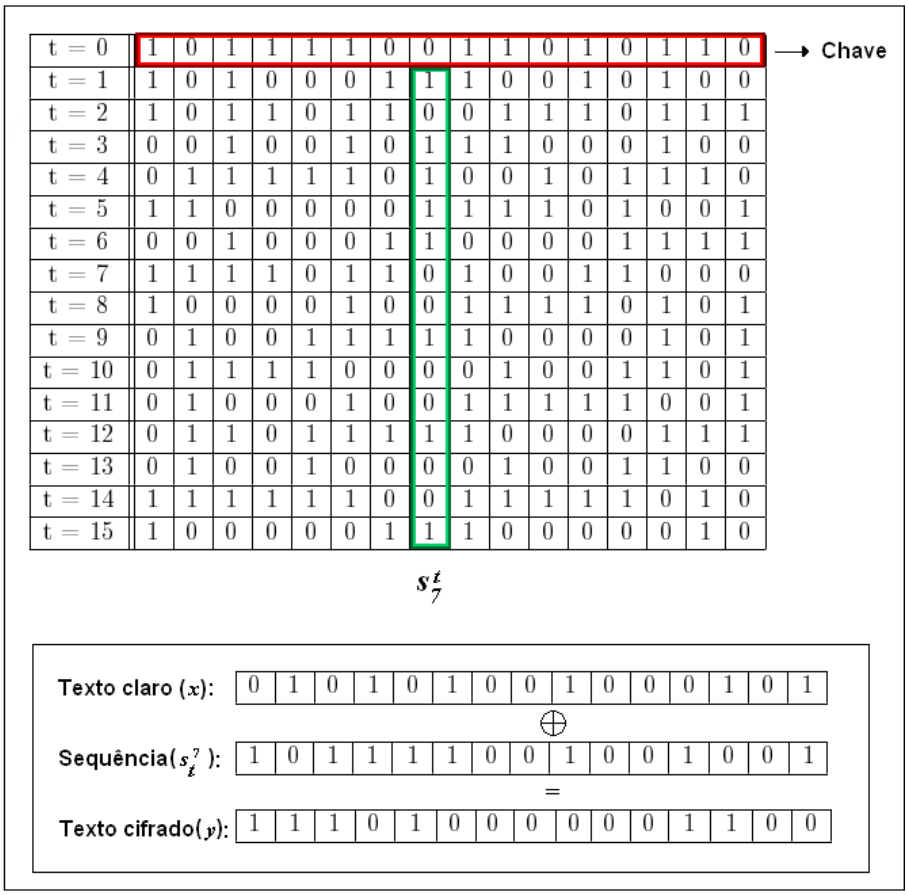


Figura 4.1: Evolução de um reticulado por 15 passos de tempo.

usa a dinâmica do AC para a realização da cifragem propriamente dita. Ele usa o AC para gerar uma sequência pseudo-aleatória e a operação XOR realiza a cifragem do texto claro.

Diversos modelos que utilizam a evolução dos ACs efetivamente no processo de cifragem/decifragem surgiram após o modelo de [Wolfram, 1986]. A maioria desses modelos são baseados nas propriedades algébricas dos ACs aditivos. O CAC é o modelo mais conhecido e utiliza ACs aditivos, não-homogêneos e reversíveis [Nandi et al., 1994]. Esse modelo foi criticado por [Blackburn et al., 1997]; segundo os autores, o modelo pode ser quebrado através de um ataque com pares do texto plano escolhido e seu texto cifrado correspondente. A principal razão para essa fraqueza do método deve-se à alta linearidade das regras aditivas. Algumas melhorias do modelo de cifragem a fim de melhorar o método CAC foram propostas posteriormente em [Ganguly et al., 2000]. Em [Sen et al., 2002] uma alteração no método de [Ganguly et al., 2000] foi proposta adicionando algumas etapas no processo de cifragem, responsável por eliminar uma fraqueza encontrada no método anterior. Bao [2004] analisou o sistema proposto por Sen [2002] e colaboradores e constatou que mesmo com a utilização de uma nova etapa o sistema criptográfico pode ser quebrado com a técnica de criptoanálise conhecida como texto claro escolhido [Bao, 2004].

Os modelos que empregam o cálculo de pré-imagem surgiram paralelamente aos modelos baseados em ACs aditivos [Gutowitz, 1995], [Oliveira et al., 2004], [Lima, 2005], [Macedo, 2007], [Oliveira et al., 2010] e [Magalhaes, 2010]. Nesses modelos, a regra de transição é obtida a partir da chave criptográfica, o texto original define o reticulado inicial e o cálculo de pré-imagens consecutivas corresponde à etapa de cifragem (evolução *backward*), enquanto a decifragem é feita utilizando-se a evolução tradicional do AC (evolução *forward*).

Nas próximas seções serão apresentados os principais modelos de criptografia baseados no cálculo de pré-imagens de ACs, relacionados ao novo modelo tridimensional proposto nessa dissertação.

4.1 Modelo de criptografia de Gutowitz

O modelo de [Gutowitz, 1995] associa um texto plano de tamanho N a um reticulado de mesmo tamanho. A cifragem nesse modelo é baseada na execução do AC para trás (*backward*) na tentativa de encontrar uma pré-imagem. Na decifragem, o receptor conhecendo a regra e o número de passos utilizados na cifragem, basta aplicar o processo inverso, ou seja, executar o AC para frente por T passos de tempo até a obtenção do texto plano novamente.

No entanto, nem todas as regras podem ser usadas para gerar uma pré-imagem de qualquer texto plano, pois não existe a garantia de existência de pelo menos uma pré-imagem. Além disso, é importante para os fins da criptografia que essa regra exiba dinâmica caótica. Para isso, Gutowitz restringiu o tipo de regra de transição que pode ser utilizada no seu modelo de criptografia. A característica imposta por Gutowitz é que as regras devem ser sensíveis a um dos bits extremos da vizinhança, ou seja, devem ser regras com sensitividade unidirecional, como foi apresentado na Seção 2.5.

Uma vez especificado o uso estrito de regras com sensitividade unidirecional, o modelo de criptografia de Gutowitz aplica um cálculo de pré-imagem específico para esse tipo de regra na etapa de cifragem. Uma das vantagens do emprego dessas regras é que esse cálculo de pré-imagem é mais simples e eficiente do que o método genérico visto na Seção 2.7. Devido à característica das regras, todas as vizinhanças são determinísticas e o cálculo de uma pré-imagem, uma vez iniciado, não precisa retornar na pilha.

O método utiliza um reticulado de tamanho N e uma regra de raio r (tamanho da vizinhança $m = 2r+1$). Para iniciar o processo de criptografia é necessário que o tamanho do reticulado inicial seja aumentado em $\delta = 2r$ em relação ao reticulado original. O próximo passo é a inicialização aleatória dos $m-1$ primeiros bits mais à esquerda ou mais à direita do reticulado em $t-1$. Essa inicialização é fortemente dependente da sensitividade da regra que estiver sendo utilizada como chave criptográfica. Se a regra for sensível à direita deve-se inicializar os $m-1$ bits mais à esquerda; caso contrário,

inicializa os bits mais à direita. A cada passo, é necessário verificar qual deve ser o bit a ser inserido na próxima célula do reticulado de acordo com a regra de transição.

A Figura 4.2 apresenta um exemplo de cálculo de pré-imagem utilizando-se um reticulado de 6 células e a regra 30 (raio 1), que é sensível à esquerda. Na Figura 4.2 (a) foi realizada uma inicialização aleatória dos bits mais à direita do reticulado: {01}. Na Figura 4.2 (b), apresenta os bits à esquerda dos iniciais, que são calculados sequencialmente, a partir da identificação das vizinhanças determinísticas. Dessa forma, independentemente do valor do bit de saída, a propriedade de sensitividade à esquerda garante que todos os bits serão obtidos deterministicamente. Diferentemente do método genérico visto na Seção 2.7, no cálculo de pré-imagem do modelo de Gutowitz as células adicionais são mantidas ao final do cálculo e os bits finais não precisam ser validados, pois não é adotada a condição de contorno periódica. Assim, sempre existe uma pré-imagem para qualquer condição inicial.

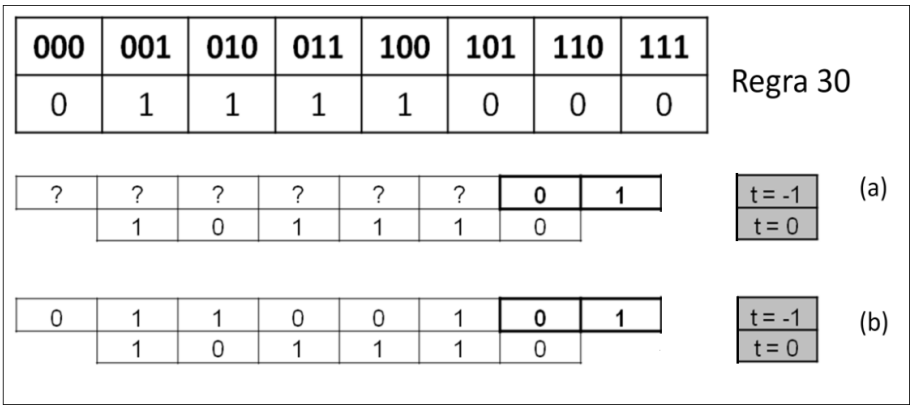


Figura 4.2: (a) Inicialização dos $m-1$ bits. (b) Cálculo da pré-imagem.

No modelo de [Gutowitz, 1995], o processo de cifragem é realizado através do cálculo de T pré-imagens consecutivas, a partir do reticulado inicial que corresponde ao texto plano. A regra de transição do AC é obtida a partir da chave criptográfica: a chave define a metade dos bits da regra (núcleo da regra), visto que, os bits da outra metade são determinados pela propriedade de sensitividade.

A quantidade de regras com sensitividade unidimensional é dada por $2^{2^{m-1}}$. A cada passo de iteração podem ser obtidas 2^{m-1} pré-imagens, uma para cada possível inicialização das $m-1$ células iniciais. Por fim, ao longo de T iterações a quantidade de pré-imagens possíveis é de $2^{T(m-1)}$.

As Figuras 4.3 e 4.4 mostram respectivamente os processos de encriptação e descriptação de um reticulado ao longo de quatro passos de iteração. Note que a criptografia nesse caso utilizou a mesma chave e a mesma quantidade de passos na cifragem e decifragem, o que garante que o processo funcione adequadamente.

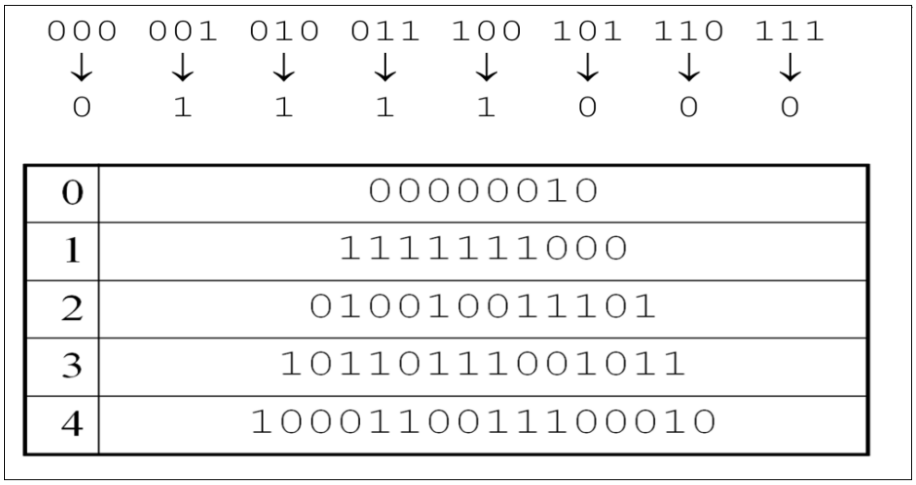


Figura 4.3: Encriptação utilizando a Regra 30 (evolução *backward*).

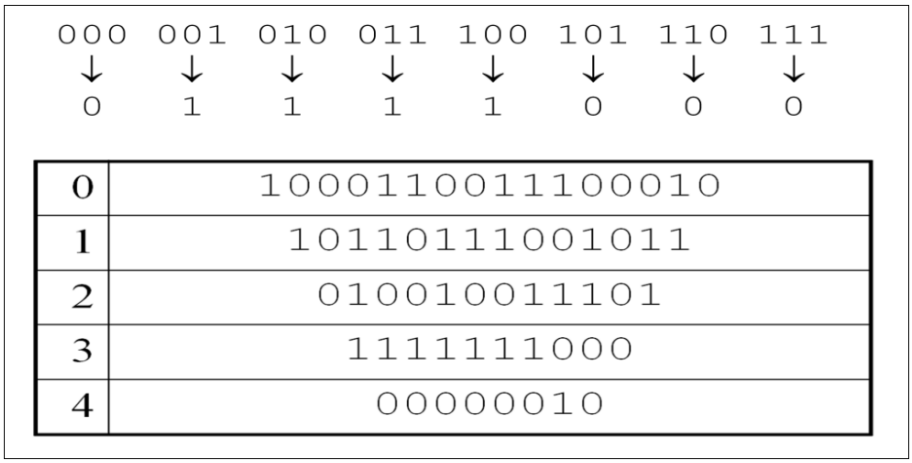


Figura 4.4: Desencriptação utilizando a Regra 30 (evolução *forward*).

4.1.1 Definição formal do modelo de Gutowitz

O criptosistema de [Gutowitz, 1995] é constituído de uma 6-tupla($\mathcal{P}$, $\mathcal{K}$, T, $\mathcal{C}$, ε, D) que satisfaz as seguintes condições:

1. $\mathcal{P}$ é um conjunto possível de reticulados iniciais de tamanho N, também denominados textos planos;
2. T representa a quantidade de passos;
3. $\mathcal{K}$ é o conjunto das chaves possíveis, sendo correspondente à metade dos bits que definem uma regra sensível a um dos extremos da vizinhança;
4. $\mathcal{C}$ é um conjunto possível de textos cifrados de tamanho L, tal que $L = N + (2r \times T)$;
5. Para cada $k \in \mathcal{K}$, existe uma função de criptográfica $e_k \in \varepsilon$, dada pelo cálculo de T pré-imagens consecutivas, e uma correspondente função de descriptografia $d_k \in D$, dada pela evolução temporal por T passos para frente. Cada e_k: $\mathcal{P} \rightarrow \mathcal{C}$ e $d_k : \mathcal{C} \rightarrow \mathcal{P}$ são funções tais que $d_k(e_k(\mathcal{P})) = \mathcal{P}$.

4.1.2 Considerações sobre o modelo de Gutowitz

Um aspecto positivo que pode ser observado no modelo é o paralelismo, pois, ao começar o cálculo da pré-imagem de um reticulado no instante t, o cálculo do reticulado $t-1$ poderá iniciar com um atraso de apenas um ciclo de *clock* do processador, equivalente ao tempo para se calcular as células iniciais da pré-imagem anterior. Na decifragem, como a evolução temporal de um AC para frente pode ser realizada de forma paralela, o modelo é totalmente paralelizado.

No entanto, esse modelo apresenta algumas desvantagens. A primeira delas é em relação ao aumento do texto cifrado. Para se ter um espaço de chaves de cardinalidade razoável (2^{1024}) , Gutowitz sugeriu o uso de regras de raio 5 que têm tamanho de 2^{11} bits, definidos por uma chave de 2^{10} bits. Assim, a cada pré-imagem, 10 bits são acrescentados no reticulado. Se forem realizados 32 passos de pré-imagem ($T = 32$), 320 bits são adicionados a cada bloco cifrado, isso gera um aumento considerável do texto encriptado.

Outro aspecto observado em relação a esse modelo, refere-se à propagação de perturbações no texto claro. Ao complementar um bit no reticulado inicial, nota-se que essa alteração se propaga somente do lado da sensitividade. A Figura 4.5 ilustra esse problema, onde o símbolo "|" representa um bit alternado e "_" representa um bit não alterado. A regra utilizada tem sensitividade à esquerda. Ao se complementar um único bit no reticulado inicial, observa-se que à medida que as pré-imagens são calculadas alguns bits se alteram, mas sempre à esquerda do bit alterado no reticulado inicial.

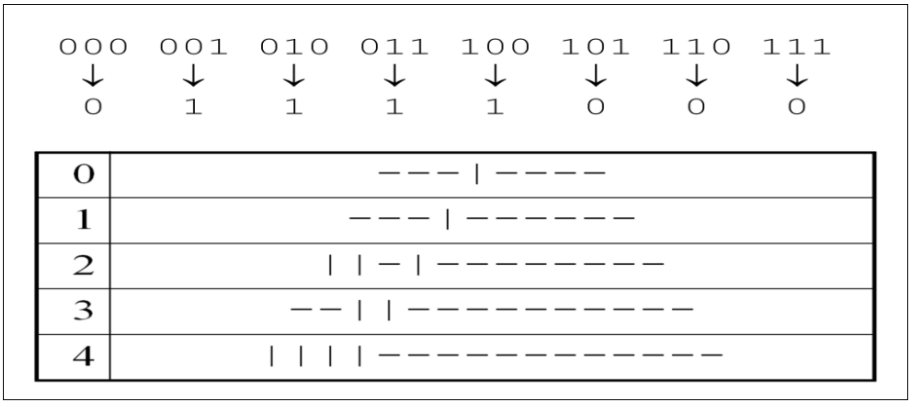


Figura 4.5: Análise da perturbação.

Para corrigir esse problema, [Gutowitz, 1995] propôs em seu trabalho a utilização de duas regras, a primeira sensível à direita e a segunda sensível à esquerda. Então, nos 32 primeiros passos a primeira regra é aplicada e nos 32 últimos passos aplica-se a segunda regra, assim o problema pode ser resolvido. As duas regras podem ser geradas a partir de uma mesma chave. A fim de evitar o uso de duas regras no processo de encriptação e reduzir o número de passos, em [Oliveira et al., 2004] foi proposto um modelo de criptografia que utiliza regras com sensitividade bidirecional.

4.2 Modelo de criptografia com sensitividade bidirecional

A ideia do modelo de criptografia com sensitividade bidirecional [Oliveira et al., 2004] é similar ao modelo de [Gutowitz, 1995], exceto pelo fato de que nesse modelo o conjunto de regras que cifram um determinado texto plano devem ser sensíveis aos dois extremos da vizinhança simultaneamente (sensitividade bidirecional).

Nesse modelo, o processo de inicialização dos $m-1$ bits pode ser dado em qualquer lugar do reticulado. Dessa maneira, o cálculo das células restantes do reticulado pode ser processado à esquerda e à direita simultaneamente.

A Figura 4.6 ilustra o cálculo de uma pré-imagem com a regra 90. Na Figura 4.6 (a), os bits {01} são inicializados aleatoriamente no centro reticulado. A Figura 4.6 (b), mostra o processo de cálculo de pré-imagem, as próximas células que serão preenchidas simultaneamente se encontram ao lado esquerdo e direito dos $m-1$ bits inicializados. Essas células devem ser preenchidas de tal forma que satisfaçam à ?01 $\rightarrow$ 1 e 01? $\rightarrow$ 1, assim as células serão preenchidas respectivamente com 0 e 1, pois 101 $\rightarrow$ 0 e 010 $\rightarrow$ 1. O processo continua até que todas as células do reticulado sejam devidamente preenchidas.

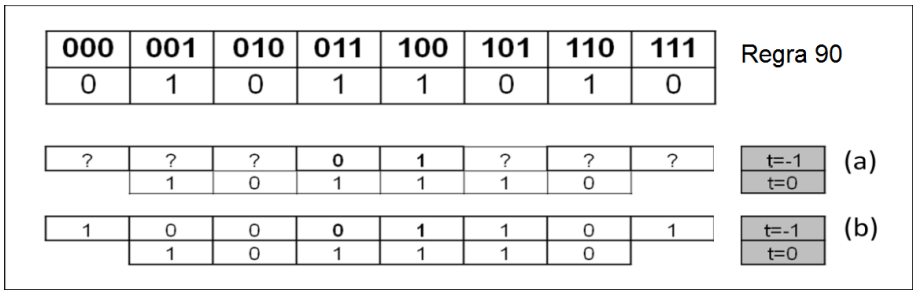


Figura 4.6: (a) Inicialização aleatória. (b) Cálculo da pré-imagem.

Na Figura 4.7 é mostrado todo o processo de encriptação que decorreu em 4 passos de tempo. Como é possível observar os bits em negrito representam as células inicializadas aleatoriamente, nesse modelo os bits iniciais podem ser escolhidos em qualquer posição do reticulado. Da mesma maneira, na Figura 4.8 podemos ver que o processo de desencriptação foi realizado.

4.2.1 Definição formal do modelo de criptografia com sensitividade bidirecional

A criptografia com regras de sensitividade bidirecional pode ser definida por uma 6-tupla($\mathcal{P}$, $\mathcal{K}$, T, $\mathcal{C}$, ε, D) que satisfaz as seguintes condições:

1. $\mathcal{P}$ é um conjunto de reticulados iniciais de tamanho N, também denominados textos planos;

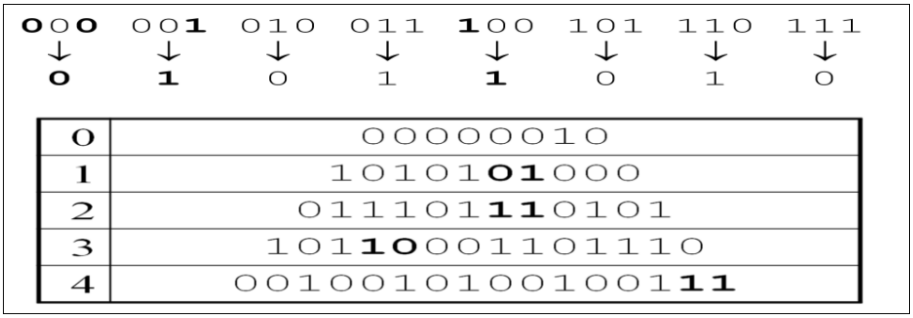


Figura 4.7: Encriptação utilizando a Regra 90.

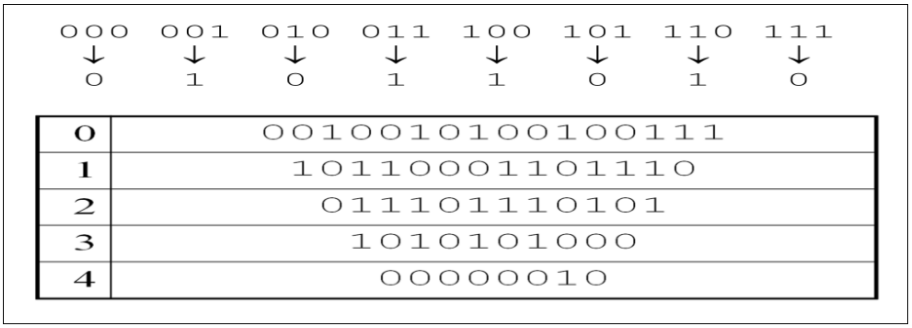


Figura 4.8: Desencriptação utilizando a Regra 90.

2. T representa a quantidade de passos;
3. $\mathcal{K}$ é o conjunto das chaves possíveis, sendo composta por 25% dos bits utilizados na geração de uma regra sensível aos dois extremos da vizinhança;
4. $\mathcal{C}$ é um conjunto possível de textos cifrados de tamanho L, tal que $L = N + (2r \times T)$;
5. Para cada $k \rightarrow \mathcal{K}$, tem uma regra de criptografia $e_k \in \varepsilon$, dada pelo cálculo de T pré-imagens consecutivas, e uma correspondente regra de descriptografia $d_k \rightarrow D$, dada pela evolução para frente do AC por T passos de tempo. Cada $e_k : \mathcal{P} \rightarrow \mathcal{C}$ e $d_k : \mathcal{C} \rightarrow \mathcal{P}$ são funções tais que $d_k(e_k(\mathcal{P})) = \mathcal{P}$.

4.2.2 Considerações sobre o modelo de sensitividade bidirecional

Uma das vantagens desse modelo é a garantia que a perturbação de um bit no reticulado inicial se espalhe ao longo de todos os passos do processo, como é mostrado na Figura 4.9. Nessa figura "#" representa um bit alterado e "_" representa um bit sem alteração. Diferentemente do que acontecia no modelo de [Gutowitz, 1995], no qual essa perturbação se propagava apenas do lado da sensitividade da regra de transição, nesse modelo percebemos que a perturbação se propaga por todo o reticulado.

Outra vantagem é em relação ao paralelismo que pode ser dado tanto na horizontal quanto na vertical. Como discutido anteriormente, na forma vertical esse paralelismo ocorre com um atraso de clock do processador, caso as células sejam inicializadas na mesma posição. Na horizontal esse paralelismo se baseia no fato de que ambas as células,

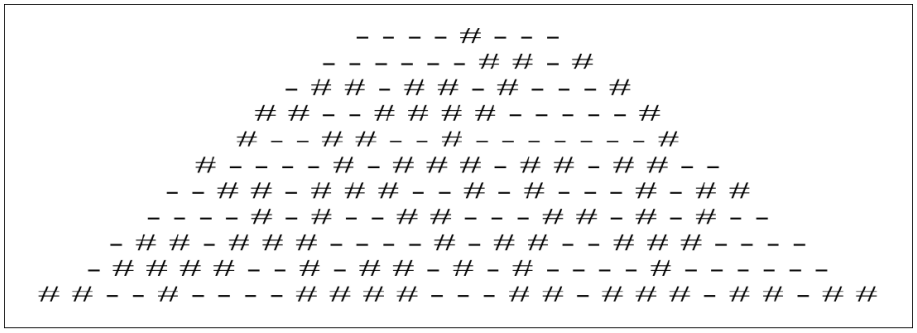


Figura 4.9: Propagação da perturbação para regras com sensitividade bidirecional.

tanto da esquerda quanto da direita, são preenchidas simultaneamente.

Por outro lado, a quantidade de regras que podem ser usadas nesse modelo, tendo em vista que são ainda mais restritas, decai pela metade em relação ao modelo de [Gutowitz, 1995]. Temos que o número de regras possíveis em [Gutowitz, 1995] é de $2^{2^{m-1}}$, no modelo bidirecional apresentado esse número é de $2^{2^{m-2}}$.

Uma alternativa para solucionar esse problema é o aumento do raio da regra utilizada, porém o tempo de processamento a cada passo fica maior. O tamanho do reticulado final também aumenta, similar ao que ocorre no modelo de [Gutowitz, 1995], o que dificulta a transmissão da mensagem ao longo de um canal.

Para evitar que o tamanho do reticulado cresça ao longo do processo de cifragem, novos métodos de criptografia foram propostos. Um deles utiliza o algoritmo reverso [Wuensche and Lesser, 1992] que será analisado na Seção 4.3.

4.3 Modelo de criptografia baseado no algoritmo reverso

Os algoritmos para cálculo de pré-imagens utilizados nos modelos anteriores implicam em um aumento do reticulado a cada passo. Por outro lado, o Algoritmo Reverso proposto por Wuensche e Lesser [1992] visto na Seção 2.7, pode ser aplicado em qualquer tipo de regra de transição e também garante que o reticulado não aumenta a cada pré-imagem calculada. Assim, ele seria um candidato natural ao uso em criptografia e foi investigado em [Lima, 2005] e [Oliveira et al., 2008].

Nesses trabalhos, a cifragem se baseia no Algoritmo Reverso, sendo que a cifragem de um texto plano é realizada calculando-se T pré-imagens consecutivas a partir do texto plano. O processo acrescenta para o cálculo de pré-imagem $m-1$ bits e após a validação esses bits são descartados. Portanto, o texto plano possui tamanho igual ao texto final (texto encriptado).

A utilização do Algoritmo Reverso tem como vantagens o fato da pré-imagem ter o mesmo tamanho do reticulado original e o fato desse método poder ser aplicado com

qualquer regra de transição não apenas as que apresentam sensitividade. Por outro lado, o método não garante que sempre exista uma pré-imagem para qualquer par reticulado e regra.

Em [Oliveira et al., 2008], as condições para que uma regra apresentasse uma pré-imagem foram investigadas, sendo que, a utilização do parâmetro Z foi de extrema importância. O parâmetro Z conforme foi descrito na Seção 2.6, pode ser dividido em duas partes, sendo elas Z_{left} e Z_{right}. Para atingir tal objetivo, regras com valor de Z igual a 1, foram utilizadas pois, de acordo com [Wuensche, 2004], regras com essa caracterização aumentam a possibilidade de encontrar uma pré-imagem válida para um reticulado arbitrário. Dessa forma, a probabilidade que uma pré-imagem seja encontrada para um texto plano qualquer aumenta. Entretanto, foi constatado que é importante termos $Z_{left} \neq Z_{right}$ para que a regra tenha alta probabilidade de existência de pré-imagem para qualquer reticulado possível. Mesmo com a utilização do Z o problema da garantia da existência da pré-imagem não foi totalmente eliminado, apenas minimizado.

4.3.1 Definição formal do modelo de criptografia baseado no algoritmo reverso em [Oliveira et al., 2008]

A criptografia com regras de sensitividade bidirecional pode ser definida por uma 6-tupla($\mathcal{P}$, $\mathcal{K}$, T, $\mathcal{C}$, ε, D) que satisfaz as seguintes condições:

1. $\mathcal{P}$ é um conjunto de reticulados iniciais de tamanho N, também denominados textos planos;
2. T representa a quantidade de passos;
3. $\mathcal{K}$ é o conjunto das chaves possíveis, sendo correspondente à regras que possuem Z_{left} ou Z_{right} igual a 1 e $Z_{left} \neq Z_{right}$.
4. $\mathcal{C}$ é um conjunto possível de textos cifrados de tamanho L tal que $L = N$;
5. Para cada $k \in \mathcal{K}$, tem uma regra de criptografia $e_k \in \varepsilon$, dado pelo pelo cálculo de T pré-imagens consecutivas, e uma correspondente regra de descriptografia $d_k \in D$, dada pela evolução do AC por T passos de tempo. Sendo e_k a função parcial, então $\neg(\forall\ x)$ significa que $(\exists\ e_k(x))$ tal que $x \in \mathcal{P}$.

4.3.2 Considerações sobre o modelo de criptografia de cálculo de pré-imagens baseado no algoritmo reverso

No modelo de criptografia proposto em [Oliveira et al., 2008] a partir dos estudos iniciados em [Lima, 2005] não existe a garantia de que qualquer texto plano possa ser cifrado ao longo de T pré-imagens. A utilização do parâmetro Z para definir as regras de transição, quando Z_{left} ou Z_{right} assumem valor 1 (vizinhança determinística) aumenta

a probabilidade de encontrar uma pré-imagem válida. Mesmo com a utilização desse parâmetro, esse problema não foi totalmente eliminado. Isso significa que a adoção do algoritmo para o cálculo de pré-imagens genérico proposto por [Wuensche and Lesser, 1992], avaliado como um possível método de criptografia, não garante a eficiência do processo pois pode ocorrer um erro na cifragem de um texto plano se uma pré-imagem não for possível de ser calculada. Com o emprego de parâmetros adicionais, que serão discutidos na Seção 4.5, em [Oliveira et al., 2008] foi possível especificar regras com baixa probabilidade de ocorrência dessa falha na cifragem. Entretanto, foi observado também que as únicas regras de ACs que asseguram a existência de 100% de pré-imagens não são apropriadas para a cifragem. Essas regras realizam um simples deslocamento do reticulado inicial e não são capazes de inserir entropia no texto cifrado. Para resolver este problema duas abordagens foram propostas. A primeira será apresentada na Seção 4.4 e utiliza dois algoritmos diferentes no processo de cifragem resultando em um texto com tamanho variável. A segunda abordagem será apresentada na Seção 4.5 e utiliza um AC não-homogêneo com duas regras no processo de cifragem: uma regra principal que atualiza a maioria das células do reticulado e uma regra de contorno que define os bits da borda do reticulado.

Uma investigação paralela sobre o uso do algoritmo reverso foi realizada pelo próprio Wuensche [2008]. O autor chegou a uma conclusão similar de que regras com $Z = 1$ e $Z_{left} \neq Z_{right}$ deveriam ser aplicadas. Entretanto, o autor não se preocupou com o fato de ser possível a ocorrência de falhas durante o cálculo de pré-imagem e sugeriu uma simples mudança de chave (regra) quando a cifragem falhasse. Obviamente em um sistema de criptografia real, tal mudança de chave é impraticável.

4.4 Modelo de Criptografia com texto cifrado de tamanho variável (VLE)

A conclusão principal da análise do método proposto em [Oliveira et al., 2008] é que a utilização do método do algoritmo reverso não é suficiente para a garantia que seja possível cifrar qualquer texto plano. Um modelo de criptografia com texto de tamanho variável foi proposto em [Oliveira et al., 2010] e é fortemente baseado nos algoritmos de criptografia de [Gutowitz, 1995] e [Oliveira et al., 2008]. A utilização de diversos parâmetros na das regras de transição retorna baixa probabilidade de falhas ao longo do processo de cifragem, entretanto, esses parâmetros ainda não garantem 100% de pré-imagens encontradas. Assim, um procedimento alternativo foi adotado em [Oliveira et al., 2010]: adicionar bits extras apenas quando a pré-imagem não for possível de ser calculada, similar ao que é feito no modelo de [Gutowitz, 1995]. Para aumentar a velocidade da cifragem, apenas regras com sensitividade unidirecional foram utilizadas nesse modelo

por serem compostas por vizinhanças deterministas.

O processo de cifragem é definido pelo cálculo consecutivo de T pré-imagens começando de um reticulado de tamanho N correspondente ao texto original. A regra de transição utilizada possui raio r e especificação baseada em três parâmetros Z_{left}, Z_{right} e sensitividade [Oliveira et al., 2001]. O algoritmo começa a cifragem utilizando o algoritmo reverso e uma chave, que atenda a especificação imposta pelos parâmetros. Caso ocorra uma falha no cálculo de uma pré-imagem em um instante de tempo β qualquer tal que $\beta \leq T$, o processo de cifragem utiliza o algoritmo reverso modificado com bits extras, similar ao algoritmo utilizado no modelo de Gutowitz [1995] na Seção 4.1, para calcular a pré-imagem em β. Assim a pré-imagem no instante de tempo β terá $N + 2r$ células. O processo de cifragem volta a utilizar novamente o algoritmo reverso para o cálculo das pré-imagens restantes. Se nenhuma outra falha ocorrer no restante do processo o texto final encriptado possuirá tamanho igual a $N+2r$. Caso alguma outra falha seja detectada, um aumento adicional de $2r$ no tamanho de $N+2r$ será realizado. Assim, o tamanho final do texto cifrado é dado por $N + 2r \times \mathcal{F}$, sendo $\mathcal{F}$ o número de falhas ocorridas no cálculo das T pré-imagens consecutivas ($0 \leq \mathcal{F} \leq T$). Portanto, o tamanho do texto cifrado pode variar entre N e $N + 2r \times T$.

4.4.1 Definição formal do modelo de criptografia com texto de tamanho variável

A criptografia com regras de sensitividade bidirecional pode ser definida por uma 6-tupla($\mathcal{P}$, $\mathcal{K}$, T, $\mathcal{C}$, ε, D) que satisfaz as seguintes condições:

1. $\mathcal{P}$ é um conjunto possível de reticulados iniciais de tamanho N, também denominados textos planos;
2. T representa a quantidade de passos;
3. $\mathcal{K}$ é o conjunto das chaves possíveis, sendo composta por metade dos bits que definem regras que possuem Z_{left} ou Z_{right} igual a 1 e $Z_{left} \neq Z_{right}$ com sensitividade unidirecional e que atenda à especificação de parâmetros adicionais [Oliveira et al., 2001].
4. $\mathcal{C}$ é um conjunto possível de textos cifrados tal que $N \leq |L| \leq N + (2r \times T)$;
5. Para cada $k \in \mathcal{K}$, tem uma regra de criptografia $e_k \in \varepsilon$, dado pelo pelo cálculo de T pré-imagens consecutivas utilizando o algoritmo reverso com condições periódicas para o cálculo de pré-imagens e o algoritmo reverso com condições não periódicas, e uma correspondente regra de descriptografia $d_k \in D$, dada pela evolução do AC por T passos de tempo. Cada $e_k : \mathcal{P} \rightarrow \mathcal{C}$ e $d_k : \mathcal{C} \rightarrow \mathcal{P}$ são funções tais que $d_k(e_k(\mathcal{P})) = \mathcal{P}$.

4.4.2 Considerações do modelo de criptografia com texto de tamanho variável

O modelo de criptografia proposto em [Oliveira et al., 2010] corrigiu a falha no uso do algoritmo reverso identificada em [Oliveira et al., 2008], visto que neste há sempre a garantia que o texto plano possui um texto cifrado correspondente. Apesar da garantia da existência de 100% pré-imagem para qualquer texto plano, o tamanho do texto final, mesmo que mais próximo do tamanho do texto plano, pode ser maior que o tamanho do texto original. Embora o tamanho final tenda a ser bem menor que o utilizado em [Gutowitz, 1995], ainda sim foi observado que na maioria das cifragens um pequeno aumento de texto é esperado.

4.5 Modelo de criptografia baseado em autômatos celulares unidimensionais (HCA)

Em [Macedo, 2007] foi proposto um modelo de criptografia baseado no cálculo de pré-imagens de autômatos celulares unidimensionais caóticos e não homogêneos. Esse método foi chamado de HCA (*Hybrid Cellular Automata*) e foi realizado um pedido de patente no INPI para o mesmo [Oliveira and Macedo, 2007].

A partir dos estudos discutidos em [Oliveira et al., 2008] observou-se que o Algoritmo Reverso proposto por [Wuensche and Lesser, 1992] não conseguia, a partir de qualquer regra sensível, encontrar uma pré-imagem para qualquer reticulado inicial. Além dissom identificou-se que regras elementares que encontram pré-imagens para qualquer texto plano são: 15, 51, 85, 170, 204, 240. Essas regras elementares, além de terem sensitividade unidirecional, têm a propriedade do bit sensível da vizinhança definir de forma única o bit de saída da regra de transição. As regras 15 e 240 são sensíveis à esquerda. Nesse caso, como visto na Seção 2.5 , para uma regra Φ composta de vizinhanças (v_0, v_1, v_{m-1}) temos que:

$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\text{Se } \Phi(v_0, v_1, \ldots, v_{m-1}) = b \text{ então } \Phi(\overline{v_0}, v_1, \ldots, v_{m-1}) = \overline{b}].$$

Adicionalmente, essas regras têm a propriedade:

$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\Phi(v_0, v_1, \ldots, v_{m-1}) = v_0]$$
$$\text{ou}$$
$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\Phi(v_0, v_1, \ldots, v_{m-1}) = \overline{v_0}].$$

De forma similar, as regras 85 e 70 são sensíveis à direita onde temos:

$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\text{Se } \Phi(v_0, v_1, \ldots, v_{m-1}) = b \text{ então } \Phi(v_0, v_1, \ldots, \overline{v_{m-1}}) = \overline{b}].$$

Adicionalmente, essas regras têm a propriedade:

$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\Phi(v_0, v_1, \ldots, v_{m-1}) = v_{m-1}]$$
$$\text{ou}$$
$$\forall\ (v_0, v_1, \ldots, v_{m-1})\ [\Phi(v_0, v_1, \ldots, v_{m-1}) = \overline{v_{m-1}}].$$

As regras 51 e 204 são sensíveis ao bit central. Por isso não podem ser aproveitadas no método de cálculo de pré-imagem e não serão detalhadas aqui. Assim, as regras 15, 85, 170 e 240 estariam aptas a formar um conjunto de regras de transição de raio que sempre encontram uma pré-imagem quando o algoritmo reverso é aplicado para qualquer reticulado inicial. Elas são apresentadas na Figura 4.10. Entretanto, essas regras possuem um comportamento dinâmico que não as tornam interessantes para comporem o conjunto de chaves de um sistema criptográfico. Essas regras apenas fazem deslocamentos à direita ou esquerda (sensitividade à esquerda ou à direita), embora o parâmetro Z das mesmas seja igual a 1. Portanto, elas não exibem o comportamento caótico necessário à cifragem do texto plano, embora a dinâmica mais provável de uma regra $Z = 1$ seja a caótica [Wuensche, 2004].

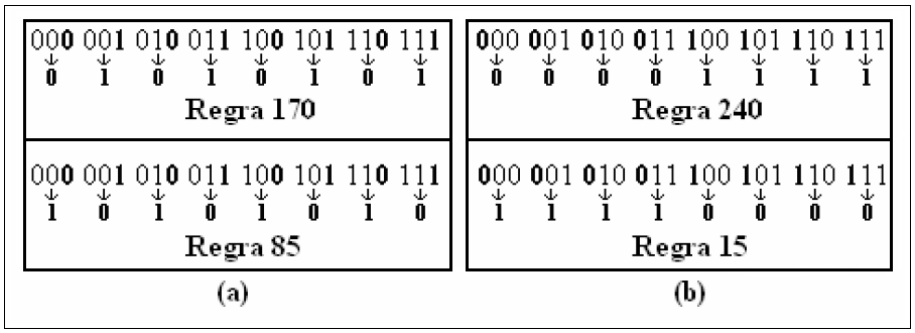


Figura 4.10: Regras capazes de resolver a condição imposta pela região de contorno do cálculo de pré-imagem proposto por Wuensche e Lesser. (a) Regras sensíveis à direita. (b) Regras sensíveis à esquerda [Macedo, 2007].

Para atender às duas propriedades necessárias ao modelo - existência de pré-imagem e dinâmica caótica - a solução encontrada em [Macedo, 2007] foi empregar duas regras diferentes no reticulado, uma que garantisse a existência da pré-imagem e outra que provesse a dinâmica caótica. A utilização de duas ou mais regras torna o AC não-homogêneo ou híbrido. A ideia principal do modelo unidimensional é a utilização da regra de contorno na borda, responsável pela validação final dos bits, e no restante do reticulado deve-se utilizar uma regra principal, responsável por gerar aleatoriedade do texto encriptado. A regra de contorno é baseada nas regras elementares com sensitividade à esquerda e à direita e que fazem apenas um deslocamento dos bits do reticulado, conforme apresentado na Figura 4.10. A regra principal deve ter a mesma sensividade da regra de contorno, o mesmo raio e deve exibir uma dinâmica caótica.

Um exemplo para o cálculo de uma pré-imagem segundo o modelo básico apresentado em [Macedo, 2007] é mostrado na Figura 4.11. Na Figura 4.11 (a) temos o reticulado inicial $\{0100101\}$, e o reticulado com as células $P_1, P_2, \cdots, P_7$, denotam os bits da pré-imagem a serem encontrados. O AC não-homogêneo é definido de forma que a regra de contorno é aplicada nas $m-1$ primeiras células do reticulado, e a regra principal nas células restantes. Neste exemplo, a regra principal (regra 30) e a regra de contorno (regra 15) são sensíveis à esquerda, portanto, o cálculo da pré-imagem deve ser seguido da direita para esquerda, como nos modelos precursores. As duas primeiras células do reticulado, que são atualizadas pela regra de contorno, estão destacadas em negrito (Figura 4.11 (a)). Essas células são denominadas de borda do reticulado. Para iniciar o cálculo são determinados os bits P_1 e P_7 pela regra de contorno (regra 15), pois os valores desses bits dependem unicamente dos estados das células de borda (Figura 4.11 (b)). Como a regra principal é sensível à esquerda, todas as demais vizinhanças que dependem dessa regra são deterministas e os demais bits são obtidos sequencialmente. Encontrados os bits iniciais, o cálculo segue da direita para esquerda (de P_6 à P_2) utilizando a regra principal. A Figura 4.11 (c) apresenta o cálculo do primeiro bit (P_6) que depende da regra principal. A Figura 4.11 (g), mostra todas as células do reticulado referente à pré-imagem preenchida. O processo pode ser repetido por T passos de tempo obtendo-se T pré-imagens consecutivas.

A evolução para frente é realizada a partir da última pré-imagem obtida aplicando-se a regra principal que atualiza todos o bits do reticulado, exceto os $m-1$ bits da borda que são atualizados pela regra de contorno. Essa evolução deve ser realizada pelos mesmos T passos de tempo utilizados no cálculo da pré-imagem obtendo o reticulado inicial novamente.

A chave secreta é uma palavra binária que define a metade dos bits da regra principal, também chamada de núcleo da regra. Os demais bits decorrem da propriedade de sensitividade da regra. A regra de contorno é definida pela sensitividade e pelo primeiro bit da regra principal. Se a chave é usada para gerar uma regra principal sensível à direita, por exemplo, a regra de contorno deve ser escolhida entre as duas regras de contorno sensíveis à direita possíveis em cada espaço de regras. Por exemplo, se forem usadas regras de raio 1, a regra de contorno sensível à direita deve ser escolhida dentre as regras 85 e 170. Em [Macedo, 2007] são apresentados argumentos que a cifragem será de qualidade superior se for escolhida a regra de contorno que tenha o bit referente à vizinhança 000 diferente do bit de saída da regra principal relativo à mesma vizinhança.

A partir desse modelo básico de cálculo de pré-imagem em um AC não-homogêneo, algumas versões diferentes foram propostas em [Macedo, 2007]. Para melhorar o tempo de cifragem um deslocamento da borda foi utilizado, pois esse deslocamento permite que a cifragem de pré-imagens consecutivas ocorra de forma paralela. Nessa modificação, a cada passo de tempo um deslocamento da borda é realizado, considerando o tamanho

do raio (r). Uma outra alteração no método básico que foi realizada a fim de melhorar segurança do método foi a utilização da rotação do núcleo da regra principal e da regra de contorno. Na próxima seção será apresentado a versão final que resultou no modelo HCA, para o qual foi solicitada a patente [Oliveira and Macedo, 2007].

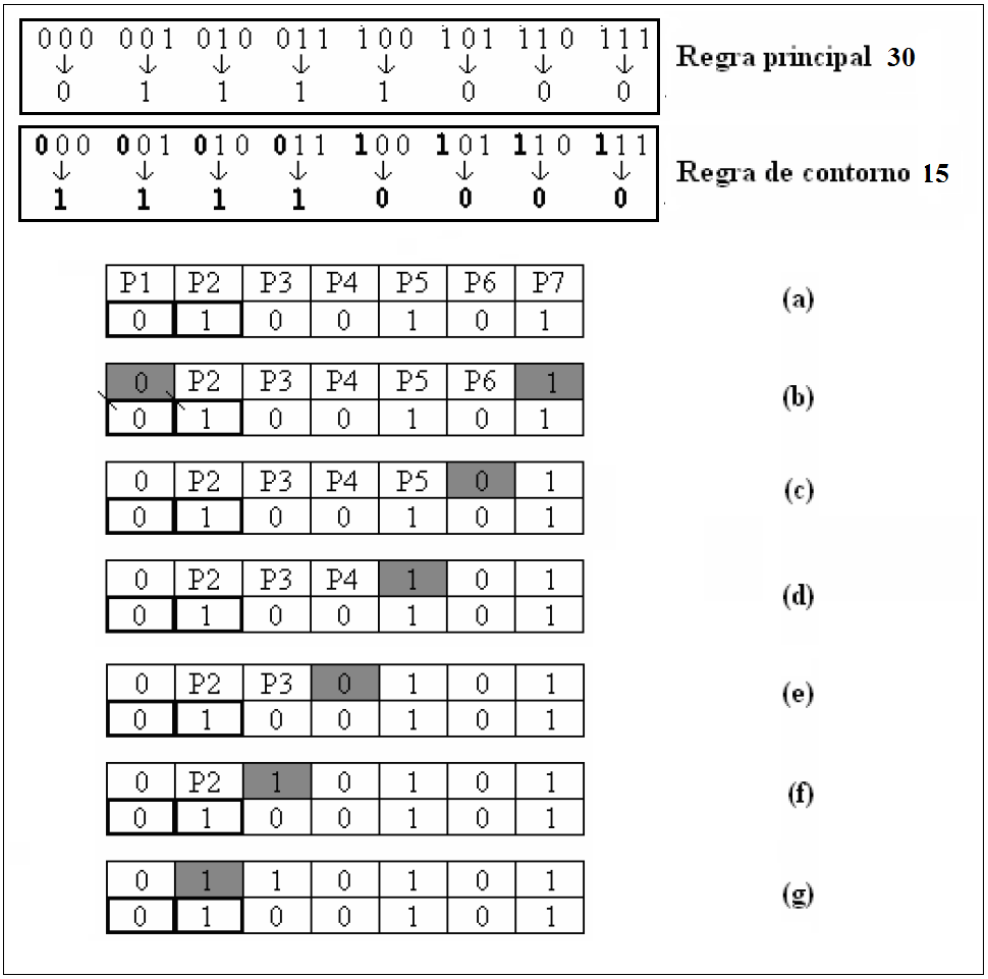


Figura 4.11: Cálculo de pré-imagem passo a passo do modelo de criptografia HCA. (a) Reticulado inicial. (b) Cálculo das células de borda a partir da regra de contorno. (c - g) Cálculo das demais células da pré-imagem a partir da regra principal [Macedo, 2007].

4.5.1 Versão final do modelo HCA

Uma das maiores motivações para se empregar ACs em criptografia está no paralelismo intrínseco dos mesmos, que tem um grande potencial para ser aproveitado em arquiteturas de hardwares paralelas. A versão básica mostrada anteriormente não permite um alto nível de paralelismo no processo de cifragem (cálculo da pré-imagem), apenas no processo de decifragem (evolução para frente) o paralelismo pode ser bem explorado. Para prover o paralelismo tanto na cifragem quanto na decifragem, algumas alterações no modelo básico se fizeram necessárias.

No modelo descrito na seção anterior, o cálculo da pré-imagem é realizado de forma

sequencial, sem possibilidade de implementação paralela. Baseado no modelo descrito na seção anterior, um novo modelo de cálculo de pré-imagem foi elaborado em [Macedo, 2007]. Esse novo modelo apresentado é capaz de prover um paralelismo na evolução para trás. Para isso, é necessária uma pequena modificação na forma como os bits da região de contorno são definidos no reticulado. Assim como na primeira versão, apenas uma regra principal e sua respectiva regra de contorno serão utilizadas. A ideia neste novo modelo é alterar, a cada passo de tempo, as posições das células que definem a região de contorno. A cada iteração, a borda, ou região de contorno, se desloca na mesma direção da sensitividade da regra principal, ao ser evoluído para trás. Sendo que na evolução para frente, a borda se desloca na direção oposta à da sensitividade.

Além disso, foi observado que na evolução do AC não-homogêneo podem ocorrer ciclos periódicos muito curtos, dependendo do tamanho do reticulado. Essa característica faz com que a cifragem de alguns textos claros seja prejudicada. Para resolver esse problema, a utilização de mais de uma regra principal ao longo da cifragem se faz necessária.

Na versão final do modelo proposto em [Macedo, 2007], o objetivo é prover o paralelismo no cálculo de pré-imagens consecutivas e fazer com que o AC seja evoluído a cada passo do cálculo de pré-imagem por uma regra diferente. O conjunto de regras utilizado deve obedecer à mesma direção da sensitividade, ou seja, todas as regras devem ser sensíveis à mesma direção. Este conjunto de regras pode ser definido por um único núcleo, que gera regras sensíveis à esquerda ou à direita. A ideia é usar o núcleo inicial, definido pela chave, para gerar a regra principal da primeira pré-imagem calculada. A cada passo do cálculo, a configuração da regra principal será gerada pelo núcleo da etapa anterior rotacionado à esquerda em um bit. Dessa forma, a chave do sistema criptográfico define um núcleo da regra que gera uma regra principal a cada passo de execução do cálculo de pré-imagem.

A Figura 4.12 apresenta o cálculo de três pré-imagens consecutivas a partir do reticulado inicial {0100101}, utilizando-se diferentes pares de regras (principal e contorno) a cada pré-imagem. A Figura 4.12 (a) apresenta as posições da borda que são deslocadas de uma pré-imagem para a outra. Cada uma das três pré-imagens calculadas possuem uma regra de transição diferente. Uma vez que o núcleo aqui empregado é {0111}, a primeira pré-imagem é executada pelo núcleo inicial {0111}, a segunda pelo núcleo rotacionado {1110} e a terceira pelo núcleo rotacionado novamente {1101}. As regras geradas por esses núcleos são representados na figura por A, B e C, respectivamente. Já a regra de contorno é dependente da regra principal gerada pelo núcleo. Por exemplo, se o núcleo for definido pelos bits {0111}, que gera a regra principal {01111000}, a regra de contorno será {11110000}, pois o primeiro bit da regra principal é {0}, sendo que o primeiro bit da regra de contorno corresponde ao seu complemento. As regras de contorno {00001111} e {11110000} são representados por D e E, respectivamente. A Figura 4.12 (b) apresenta o início do cálculo da primeira pré-imagem, que é igual ao que é feito no modelo da primeira

versão (Figura 4.11 (b)): define-se o valor das duas células que dependem apenas das células de borda do reticulado inicial, devido à regra de contorno. O cálculo nessa primeira pré-imagem prossegue de forma similar à Figura 4.11, como é possível observar na Figura 4.12 (c). Observe que no mesmo momento em que a quarta célula da primeira pré-imagem é calculada, os bits iniciais da segunda pré-imagem também já podem ser definidos, pois, eles dependem apenas dos bits que já foram definidos na primeira pré-imagem, nos passos anteriores (Figura 4.12 (d)).

Na Figura 4.12 (e) vemos que a quarta célula da primeira pré-imagem e a terceira célula da segunda pré- imagem também podem ser calculadas em paralelo, utilizando-se suas respectivas regras principais definidas ao lado do reticulado, nesse caso, A e B. Na Figura 4.12 (f) vemos que a terceira pré-imagem define os bits inicias com a regra de contorno, no mesmo momento em que são calculados um bit da segunda e um bit da primeira pré-imagem. A partir deste ponto (Figura 4.12 (g)), as próximas células de cada pré-imagem podem ser calculadas em paralelo, dependendo unicamente da respectiva regra principal. Assim, pode-se perceber que com o deslocamento da borda, é possível calcular quantas pré-imagens se desejar, com um atraso de duas unidades de tempo do início de uma pré-imagem para outra. Na Figura 4.12 são apresentadas passo a passo dez unidades de tempo para que o cálculo seja completamente concluído.

4.5.2 Definição formal do modelo de criptografia HCA

A criptografia do modelo HCA pode ser definida por uma 6-tupla($\mathcal{P}$, K, T, $\mathcal{C}$, ε, D) que satisfaz as seguintes condições:

1. $\mathcal{P}$ é um conjunto de reticulados iniciais de tamanho N, também denominados textos planos;
2. T representa a quantidade de passos;
3. $\mathcal{K}$ é o conjunto das chaves possíveis, sendo composta por uma palavra de 256 bits que é o núcleo da primeira regra principal. A partir do núcleo, um regra principal de raio 4 sensível à esquerda ou à direita é gerada. A regra de contorno também de raio 4 possui a mesma sensitividade da regra principal e é escolhida de um conjunto de apenas quatro regras possíveis;
4. $\mathcal{C}$ é um conjunto possível de textos cifrados tamanho L, tal que $L = N$;
5. Para cada $k \in \mathcal{K}$, tem uma regra de criptografia $e_k \in \varepsilon$ dada pelo cálculo de pré-imagem do modelo de AC híbrido e uma correspondente regra de descriptografia $d_k \in D$ dada pela evolução para frente do modelo de AC híbrido. Cada $e_k : \mathcal{P} \to \mathcal{C}$ e $d_k : \mathcal{C} \to \mathcal{P}$ são funções tais que $d_k(e_k(\mathcal{P})) = \mathcal{P}$.

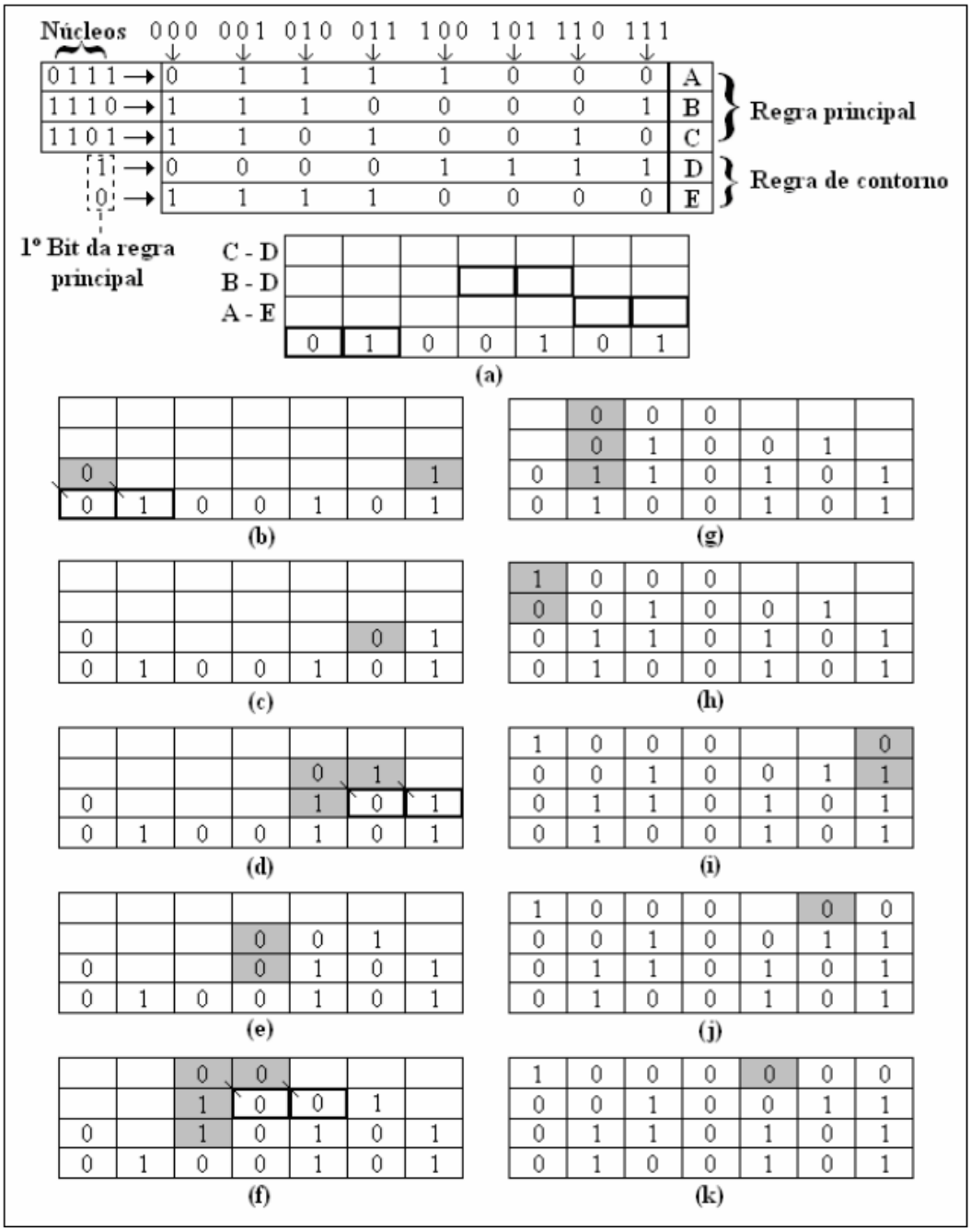


Figura 4.12: Cálculo paralelo de três pré-imagens do modelo de criptografia HCA. (a) Reticulado inicial. (b - k) Cálculo passo a passo das células do reticulado da pré-imagem [Macedo, 2007].

4.5.3 Considerações sobre o modelo de criptografia HCA

O modelo de criptografia HCA se mostrou bastante eficiente para a cifragem de mensagens lineares (textos unidimensionais). Esse modelo resolveu os aspectos deficientes nos modelos anteriores, pois é possível obter uma pré-imagem do mesmo tamanho do texto plano o que reduz o tempo de comunicação entre as entidades envolvidas durante a transmissão de dados. Outro aspecto melhorado é a garantia da existência de uma pré-imagem para qualquer texto plano qualquer que seja o reticulado inicial. Uma análise

desse modelo com o uso da Teoria dos Grafos será apresentada no Capítulo 5.

Em [Magalhaes, 2010] tentou-se aplicar o modelo unidimensional proposto por [Macedo, 2007] na criptografia de imagens. O modelo unidimensional HCA não se mostrou eficiente devido ao fato de que este algoritmo não contempla a grande redundância de dados, uma particularidade estrutural das imagens resultando em zonas de texturas. Por isso, um algoritmo que considera a característica bidimensional das imagens foi proposto em [Magalhaes, 2010] e será mostrado na próxima seção.

4.6 Modelo de Criptografia baseado em Autômatos Celulares bidimensionais (THCA)

O modelo THCA [Magalhaes, 2010] é fortemente baseado no modelo HCA discutido na seção anterior e manteve as características principais propostas no modelo unidimensional [Macedo, 2007]: uso de uma regra de contorno e uma regra principal, e cifragem baseada no cálculo de pré-imagens. A principal alteração proposta foi a utilização de um AC bidimensional (reticulado 2-dimensional e regra com vizinhança de Von Neumman). A regra com vizinhança de Von Neumman é mais adequada para a utilização da propriedade de sensitividade no cálculo da pré-imagem. Uma regra com essa vizinhança pode ser sensível na direção N (norte), L (leste), O (oeste) ou S (sul). A Figura 4.13 apresenta o exemplo de duas regras que podem ser aplicadas no método, sendo que ambas são sensíveis ao bit norte. A primeira regra pode ser aplicada como uma regra principal no modelo THCA e possui a seguinte sequência de bits de saída: 01101000111101001001011100010 11. A segunda regra é dada pela sequência 11111111111111110000000000000000 e pode ser usada como regra de contorno. Nessa segunda regra, o bit de saída é determinado apenas pelo bit norte da vizinhança, sendo que o bit de saída (novo valor da célula central) é dado pelo complemento do valor do bit norte.

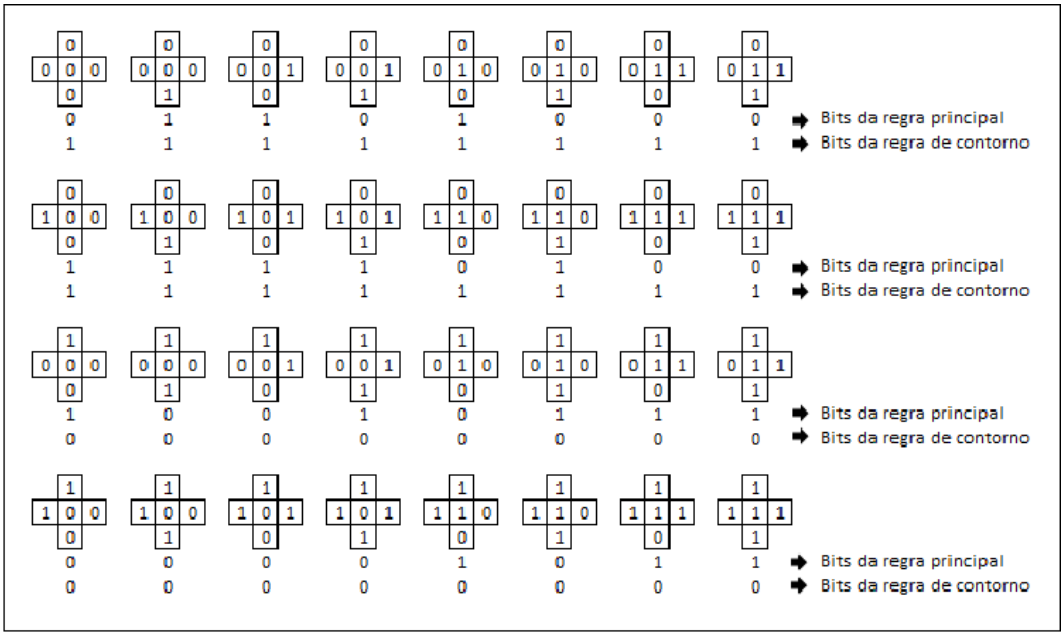


Figura 4.13: Regras de transição bidimensionais utilizadas no cálculo de uma pré-imagem.

O cálculo de uma pré-imagem empregado no modelo THCA pode ser entendido a partir do exemplo mostrado na Figura 4.14 utilizando as regras principal e de contorno indicadas na Figura 4.13 com sensitividade ao bit norte.

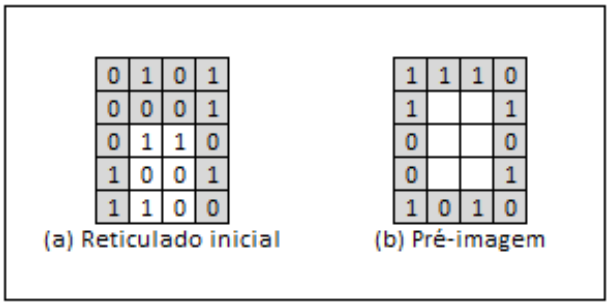


Figura 4.14: (a) Reticulado inicial utilizado no processo de cifragem. (b) Início do cálculo pré-imagem a partir das células da borda.

No reticulado inicial (correspondente ao texto plano) da Figura 4.14 (a) a borda para o reticulado da pré-imagem é identificada em negrito. A borda é formada por duas linhas consecutivas e duas colunas consecutivas do reticulado. As células da pré-imagem que dependem dos bits da borda do reticulado inicial, destacadas em negrito na Figura 4.14, são calculados seguindo a mesma direção da sensitividade da regra principal. Para realizar o cálculo dos bits que dependem da borda devemos verificar a regra de contorno. No exemplo, a regra de contorno indica que o estado da célula central da vizinhança deve receber no próximo instante de tempo o complemento do estado da célula ao norte da vizinhança. Isso significa que todos os bits destacados na Figura 4.14 (a) que formam a borda do reticulado inicial deverão deslocar uma posição no sentido norte e cada um

desses bits deve ser complementado. Os bits destacados na Figura 4.14 (b) representam os bits iniciais do cálculo da pré-imagem que dependem inicialmente dos bits da borda do reticulado inicial para serem calculados. A partir dessas células, os bits restantes da pré-imagem podem ser calculados utilizando-se a regra principal da Figura 4.13. Os bits restantes da pré-imagem devem ser calculados de acordo com a sensitividade ao bit norte da vizinhança, ou seja, o cálculo deve ser feito de baixo para cima, conforme apresentado na Figura 4.15. Os bits da pré-imagem devem ser calculados a partir da primeira linha acima das células definidas pela regra de contorno. Para o primeiro bit da pré-imagem, indicado por "?" na Figura 4.15 (a), da pré-imagem devemos verificar qual vizinhança casa com o padrão:

	?	
0	1	0
	1	

e leva o bit central do reticulado inicial ao estado 0. A vizinhança da regra principal (Figura 4.13) que satisfaz essa saída é a vizinhança

	0	
0	1	0
	1	

Portanto, a célula destacada por "?" na Figura 4.15 (a) deve ser preenchida pelo bit 0. Da mesma maneira, o próximo bit indicado por "?" na Figura 4.15 (b), é calculado e temos que observar qual a vizinhança casa com o padrão:

	?	
1	0	1
	1	

e leva o bit central do reticulado inicial para o bit 1. A vizinhança que satisfaz essa condição é a vizinhança apresentada abaixo:

	0	
1	0	1
	1	

e o bit "?" na Figura 4.15 (b) também deve ser preenchido por 0.

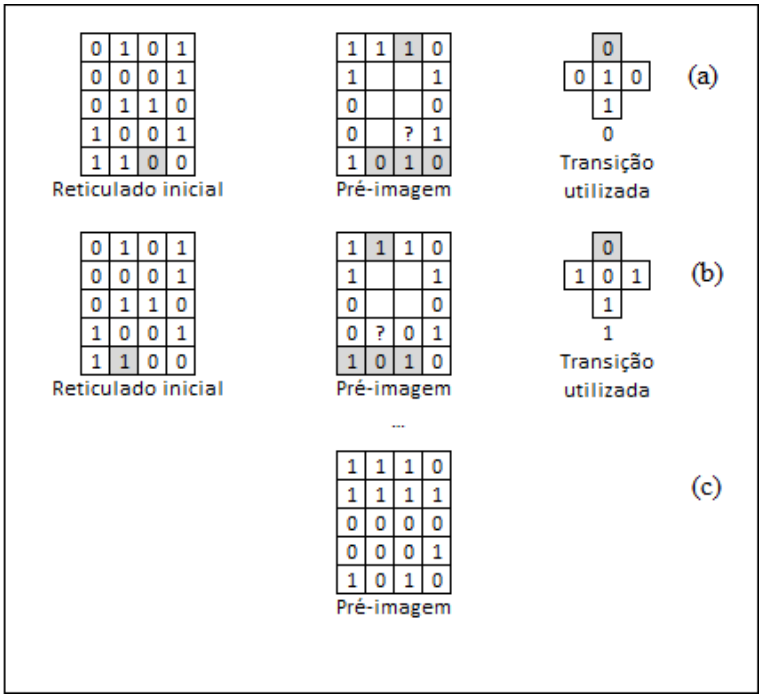


Figura 4.15: Cálculo da pré-imagem no modelo bidimensional.

Uma vez calculada a primeira linha externa, os bits restantes da segunda e terceira linhas podem ser calculados de forma similar até que toda a pré-imagem seja obtida, como apresentado na Figura 4.15 (c).

A cifragem de uma matriz binária correspondente ao reticulado inicial é realizada pelo cálculo de T pré-imagens consecutivas até que a matriz binária final define o texto cifrado. Quando utilizado na cifragem de imagens preto e branco, a imagem a ser cifrada corresponde ao reticulado inicial e a matriz correspondente à última pré-imagem calculada define a imagem cifrada.

No desenvolvimento do THCA também foram avaliadas e incorporadas as mesmas alterações no modelo unidirecional HCA para melhoria do paralelismo e da qualidade da cifragem. Assim, a borda também é deslocada de uma pré-imagem para a outra em $2r$ posições tanto no eixo norte-sul quanto no eixo leste-oeste. Além disso, uma regra principal diferente é aplicada a cada pré-imagem. Essas regras são geradas a partir de uma rotação dos bits do núcleo que são definidas pela chave criptográfica.

4.6.1 Definição formal do modelo de criptografia THCA

A criptografia do modelo THCA pode ser definida por uma 6-tupla($\mathcal{P}$, $\mathcal{K}$, T, $\mathcal{C}$, ε, D) que satisfaz as seguintes condições:

1. $\mathcal{P}$ é um conjunto de reticulados iniciais de tamanho $n \times m$, também denominados imagens originais;
2. T representa a quantidade de passos;

3. $\mathcal{K}$ é o conjunto das chaves possíveis, sendo composta por uma palavra de 256 bits que é o núcleo da regra principal. A partir do núcleo, uma regra principal sensível a um dos extremos N, S, L e O e uma regra de contorno que possui a mesma sensitividade da regra principal é escolhida de um conjunto de apenas oito regras possíveis.
4. $\mathcal{C}$ é um conjunto de textos cifrados de tamanho $n \times m$;
5. Para cada $k \in \mathcal{K}$, tem uma regra de criptografia $e_k \in \varepsilon$, dada pelo cálculo de pré-imagem do modelo de AC 2D híbrido, e uma correspondente regra de descriptografia $d_k \in D$, dada pela evolução para frente do modelo de AC 2D híbrido. Cada $e_k : \mathcal{P} \to \mathcal{C}$ e $d_k : \mathcal{C} \to \mathcal{P}$ são funções tais que $d_k(e_k(\mathcal{P})) = \mathcal{P}$.

4.6.2 Considerações sobre o modelo de criptografia THCA

O modelo de criptografia proposto em [Magalhaes, 2010] pode ser aplicado na criptografia de imagens preto e branco, nas quais os pixels brancos são representados por células no estado 0, enquanto os pixels preto são representados por células no estado 1. A Figura 4.16 (a) representa uma matriz binária de 16×16 bits e a Figura 4.16 (b) representa a respectiva imagem (16×16 pixels) preto e branco formada por essa matriz.

O modelo THCA possui boa capacidade propagação de perturbação no reticulado inicial e este modelo pode ser largamente utilizado em imagens preto e branco de dimensões $m \times n$, sendo que o número de pré-imagem é definido em função da maior dimensão da imagem. O algoritmo de criptografia THCA apresentou comportamento fortemente sensível à geometria da imagem. Além disso, as investigações apontaram que a sensitividade da regra deve ser escolhida no mesmo eixo da maior dimensão da imagem. Em [Magalhaes, 2010] o modelo THCA também foi aplicado na cifragem de imagens em escala de cinza. Como as imagens na escala cinza permitem 256 intensidades possíveis, cada pixel deve ser transformado em 8 bits. Dessa forma, a imagem está apta a ser processada pelo algoritmo. Entretanto, essa transformação gera um aumento do tamanho em pelo menos uma das dimensões da imagem, o que leva a um aumento considerável no número de pré-imagens necessárias.

Finalmente, quando o modelo THCA foi aplicado na cifragem de imagens coloridas, o aumento do tamanho da matriz binária tornou-se um impedimento maior para a utilização do método. Por exemplo, na utilização de imagens em padrão RGB cada pixel deve ser convertido em 24 bits. Assim, se a imagem da Figura 4.16 fosse convertida para o padrão RGB, na cifragem pelo THCA a matriz resultante teria tamanho 384×16 ou 192×32 ou 96×64 ou 48×128 (dependendo da forma de distribuição dos bits escolhida), o que levaria a um aumento significativo no número de pré-imagens. Para atenuar esse aumento na dimensão com consequente aumento no número de passos necessários, foi avaliado em [Magalhaes, 2010] a possibilidade de cifrar as matrizes do três canais R, G e B em separado. Entretanto, essa estratégia não se mostrou válida pois foi observada a formação

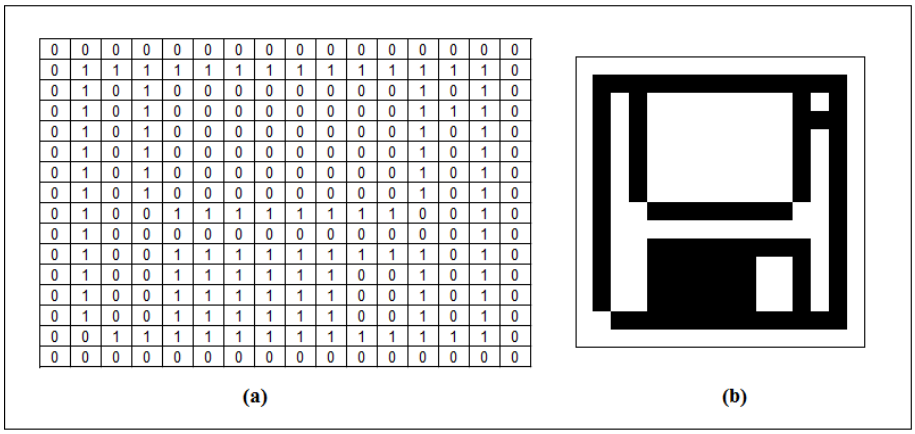


Figura 4.16: (a) Matriz binária de 16×16 pixels. (b) Imagem do disquete correspondente à matriz.

de padrões de zonas de textura na imagem cifrada. Uma das conclusões principais em [Magalhaes, 2010] é que o THCA se mostrou adequado à cifragem de imagens binárias, teve um acréscimo de tempo no caso das imagens em escala de cinza, mas teve um grande decaimento de performance no caso de imagens coloridas, devido ao alto número de pré-imagens necessárias. Assim, foi sugerido o uso de ACs tridimensionais em uma pesquisa que desse continuidade ao THCA.

4.7 Outros modelos de criptografia para imagens baseados em ACs

Alguns modelos de criptografia para imagens utilizando ACs unidimensionais também foram propostos anteriormente. Na referência [Yu et al., 2010], utilizou-se um AC unidimensional para a realização da cifragem, esse modelo é baseado em rotações e utiliza a operação XOR no processo de cifragem/decifragem. O método proposto em [Maleki et al., 2008] aborda a utilização de AC com memória, no entanto, nesse método existe perda de dados ao longo do processo da cifragem, enquanto que o método proposto nessa dissertação visa manter a integridade dos dados. Em [Jun, 2009], propõe-se um novo modelo de criptografia para imagens baseado no fato de que algumas regras de transição de ACs elementares resultam em alguns atratores de estados que podem ser utilizados como uma função de encriptação que transforma os valores dos pixels. O resultado desse algoritmo proposto em [Jun, 2009] resulta em imagens na escala de cinza. O algoritmo proposto nessa dissertação tem como resultado imagens coloridas e não imagens na escala de cinza.

Em [Machhout et al., 2009], o modelo proposto contempla substituição do valor dos pixels através de operações XOR com valores gerados a partir de um AC bidimensional com vizinhança de Von Neummanm, dessa forma o AC empregado no modelo é utilizado apenas para geração de números aleatórios, similar ao modelo unidimensional de Wolfram

[1984]. Na referência [Chen and Lai, 2007], propõe-se um novo modelo criptográfico em que a ideia principal se baseia em transformações nos valores dos pixels da imagens realizadas a partir de modificações e substituições de um AC bidimensional, nesse modelo o AC também é utilizado apenas como geração de números aleatórios. Outro método encontrado na literatura que utiliza ACs bidimensionais com memória e mapas caóticos para a cifragem simétrica de imagens, foi apresentado por [Habibipour et al., 2010]. Os modelos de criptografia para imagens que utilizam ACs bidimensionais apresentados não são altamente paralelizáveis como o modelo THCA proposto em [Magalhaes, 2010] e o modelo tridimensional proposto na presente dissertação de mestrado. Uma das características similares dos modelos apresentados em relação do modelo proposto é o fato da encriptação ser simétrica.

Mais recentemente, um novo modelo baseado no atrator de Lorenz's foi proposto em [Goncalves et al., 2010]. A ideia do método é baseada em modos de operação que proveem interação entre a chave, a mensagem e o sistema caótico, e a versão mais rápida do algoritmo tem performance comparável a do AES. O algoritmo proposto nessa dissertação por ser altamente paralelizável possui uma velocidade de cifragem bastante rápida e o método aqui apresentado também é um modelo de cifragem caótico.

4.8 Modelo de criptografia baseado em autômatos celulares tridimensionais (3DHCA)

Para melhorar a qualidade da imagem cifrada, de tal forma que a perturbação de maneira mais rápida, um novo modelo de criptografia baseado no cálculo de pré-imagens de autômatos celulares tridimensionais será investigado na presente dissertação de mestrado. Esse novo modelo pode ser aplicado em imagens na escala cinza, em imagens na escala RGB e também pode ser utilizado na cifragem de textos. Esse método é fortemente baseado nos modelos precursores HCA [Macedo, 2007] e THCA [Magalhaes, 2010] e será apresentado no Capítulo 6. Antes disso, será apresentada no Capítulo 5 uma análise da segurança do método HCA, baseada na Teoria dos Grafos.

Capítulo 5

Análise do modelo unidimensional HCA segundo a teoria dos grafos

A primeira investigação desenvolvida nessa dissertação foi o estudo do modelo de criptografia HCA, proposto em [Macedo, 2007], do ponto de vista teórico, através da transformação desse modelo em um grafo de um autômato finito com saída. O principal objetivo desse estudo é investigar o potencial de segurança do HCA buscando relacioná-lo a um modelo teórico.

5.1 Modelagem da evolução temporal para frente por máquina de Moore

A evolução temporal para frente de um autômato celular padrão (homogêneo, síncrono e contorno periódico) pode ser transformada em um autômato finito com saída, independentemente da regra aplicada, sendo possível sua modelagem como Máquina de Moore ou Máquina de Mealy [Sutner, 1991]. A Máquina de Moore possui um conjunto finito de estados Q, um estado inicial s_0, um alfabeto de entrada Σ_1, um alfabeto de saída Σ_2, um conjunto de transições δ: Q x $\Sigma_1 \rightarrow Q$ e uma função λ: $Q \rightarrow \Sigma_2$ que define a saída associada a cada estado. No caso da Máquina de Moore que modela a evolução para frente de um AC binário tanto o alfabeto de entrada quanto o de saída são binários $\Sigma_1 = \{0, 1\}$ e $\Sigma_2 = \{0, 1, \varepsilon\}$, sendo ε o símbolo da palavra vazia. Ou seja, tanto as transições quanto as saídas do estados são representados por bits. Nesse modelo, os m primeiros estados percorridos após o estado s_0 possuem saída vazia e correspondem à leitura dos $m - 1$ bits da primeira vizinhança do reticulado. A partir do m-ésimo estado percorrido a saída associada ao estado do autômato relaciona-se a um bit saída da regra de transição (estado q_i), tal que $0 \leq i \leq 2^m - 1$. A Figura 5.1 representa a máquina de Moore responsável pela execução para frente de um AC de raio 1. O estado denominado q_0 é equivalente à leitura da vizinhança 000 nas três últimas células do reticulado, o estado q_1 relaciona-se

à vizinhança 001 e assim por diante até o estado q_7, que relaciona-se à vizinhança 111. As saídas b_0, b_1, b_2, b_3, b_4, b_5, b_6 e b_7 associadas aos estados q_0, q_1, q_2, q_3, q_4, q_5, q_6 e q_7 referem-se aos bits de saída da regra de transição referentes a cada vizinhança. As transições entre os oito estados na extremidade do grafo (estados de q_0 a q_1) são definidas como a leitura da próxima célula no reticulado, considerando-se um processamento do reticulado da esquerda para a direita. Por exemplo, suponha que os últimos três bits lidos no reticulado sejam 011. Nesse caso, o estado corrente do autômato seria o q_3 e a saída associada a esse estado (b_3) seria o último bit gerado na fita de saída. Essa saída estabelece o novo valor da célula central da vizinhança 011 no próximo passo de tempo. Se, continuando a leitura do reticulado, o próximo bit à direita for 0, então a vizinhança muda de 011 para 110. Nesse caso, o autômato muda do estado q_3 para o estado q_6 e a saída b_6 estabelece o novo valor da célula da vizinhança 110.

Como exemplo, a Figura 5.2 apresenta a Máquina de Moore equivalente à aplicação da regra de transição da Figura 2.1 {01111000} na atualização dos estados de um reticulado binário qualquer.

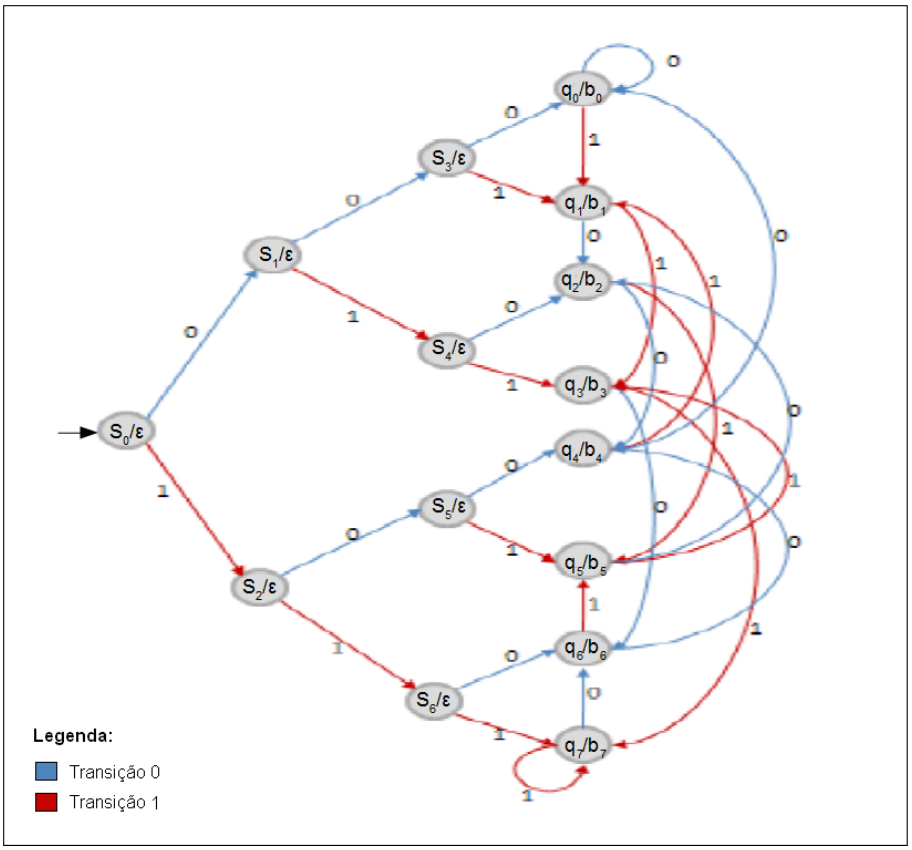


Figura 5.1: Máquina Moore genérica para AC de raio 1.

A título de exemplo, suponha que o reticulado inicial seja o mesmo da Figura 2.1 {101110} e a sequência binária de estados do reticulado seja posicionada na fita de entrada da Máquina de Moore da Figura 5.2. Para que a primeira saída seja o novo estado da primeira célula do reticulado e a última saída o novo estado da última célula, as

extremidades do reticulado são replicadas. A vizinha mais à esquerda da primeira célula (o estado da última célula) e a vizinha mais à direita da última célula (o estado da primeira célula) são copiados para simular a condição periódica, definindo a seguinte entrada para o processamento da Máquina de Moore: {01011101}. A sequência de transição de estados ao processar essa entrada é dado por: $s_0 \rightarrow s_1 \rightarrow s_4 \rightarrow q_2 \rightarrow q_5 \rightarrow q_3 \rightarrow q_7 \rightarrow q_6 \rightarrow q_5$. Nesse caso, a sequência de bits gravada na saída, será {101000}, que corresponde ao reticulado em $t = 1$ da Figura 2.1. Caso se deseje aplicar a regra de transição mais uma vez, basta repetir o precedimento: copiar os estados da primeira e última célula e voltar ao estado inicial s_0.

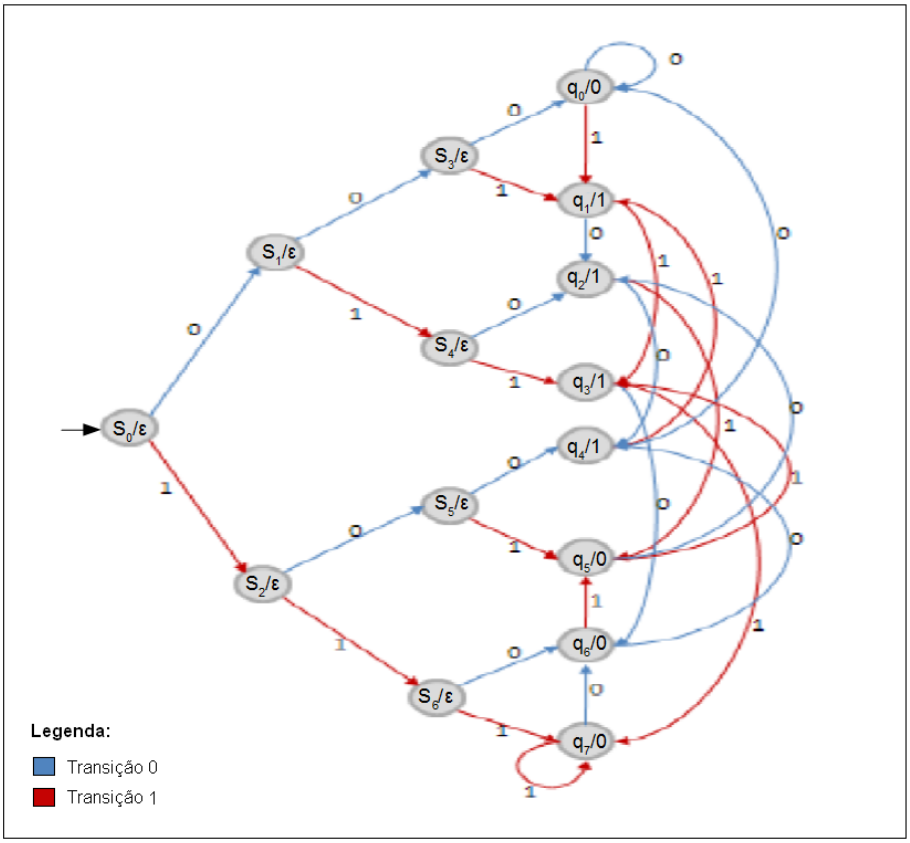


Figura 5.2: Máquina Moore relativa à execução *forward* da Regra 30.

No caso do Modelo de AC híbrido utilizado no modelo criptográfico HCA [Macedo, 2007], duas regras diferentes são aplicadas na atualização do reticulado. Para um AC de raio r e reticulado de N células, sendo $N >> r$, a regra de contorno é aplicada nas $2r$ células da borda do reticulado, e a regra principal é aplicada no restante do reticulado, num total de $N-2r$ células. Na modelagem pela Máquina de Moore, a evolução para frente desse modelo de AC híbrido pode ser facilmente adaptada do modelo de AC padrão, visto na Figura 5.1. Suponha por simplicidade, que a borda está localizada nas $2r$ primeiras células do reticulado da Figura 4.11 (0110101) que será utilizada como exemplo. Assim, para evoluir esse reticulado para frente, as $m-1$ primeiras células devem ser atualizadas de acordo com a regra de contorno: {11110000} (regra 15). A regra de contorno indica

apenas se os $m-1$ bits que compõem a borda devem ou não ser complementados, em relação ao bit da célula sensível. No caso da utilização da regra de contorno {11110000} os $m-1$ bits do reticulado deverão ser definidos pelo complemento do bit mais à esquerda da vizinhança. A partir do m-ésimo bit lido, todos os demais devem ser atualizados de acordo com a regra principal: {01111000} (regra 30). Assim, o grafo apresentado na Figura 5.3 modela a evolução para frente de um AC híbrido, mais especificamente da evolução para frente do modelo HCA. A principal diferença entre o grafo da evolução para frente de um AC genérico e de um AC híbrido é o fato desse último grafo conter $(2r+1) \times (m-1)$ mais estados que o primeiro. Esses estados adicionais representam justamente as saídas da regra de contorno.

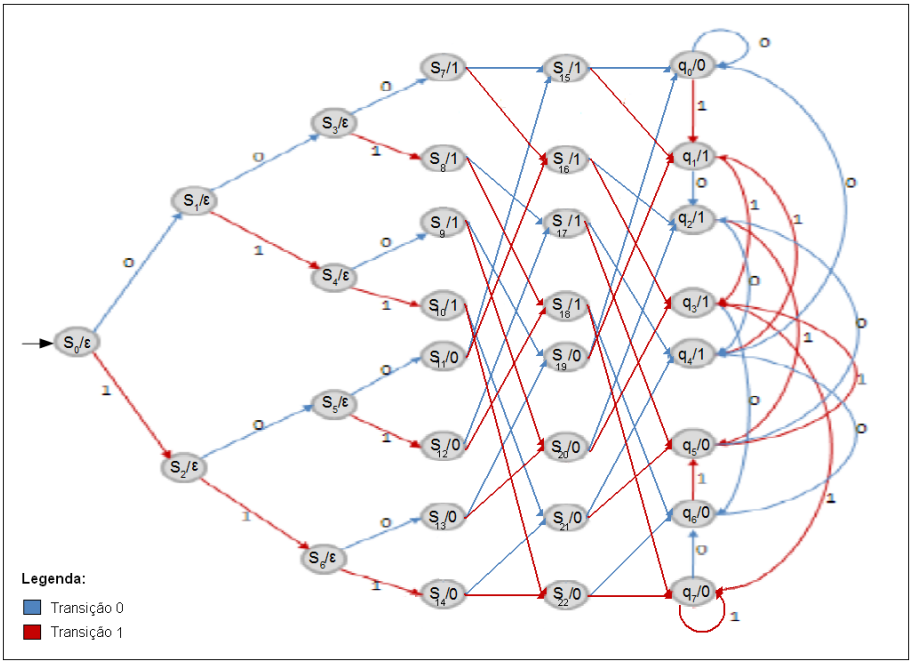


Figura 5.3: Máquina Moore relativa à execução *forward* de um AC híbrido.

Para exemplificar o início do cálculo de uma pré-imagem, suponha que o bit da primeira célula seja replicado para a última posição, adjacente à última célula, e o bit da última célula seja clonado para a posição adjacente à primeira célula. Assim, teremos o seguinte reticulado: {101101010}. Inicialmente, a Máquina de Moore começa no estado s_0, em seguida, a máquina lê o bit 1, e vai para o estado s_2. No próximo passo, a máquina lê o bit 0 e vai para o estado s_5. Em seguida, a máquina lê o bit 1 e vai para o estado s_{12} que retorna como saída o bit 0, que corresponde à saída $101 \rightarrow 0$ dada pela regra de contorno. No próximo instante, a máquina lê novamente o bit 1 e a máquina vai para o estado s_{18} que tem como saída o bit 1 ($011 \rightarrow 1$ dada pela regra de contorno). Terminados os bits da borda que foram preenchidos pela regra de contorno, a máquina avança para um dos estados q_i que representam os estados que retornam a saídas da regra principal. Assim, no próximo instante de tempo a máquina lê o bit 0 e vai para o estado q_6 e retorna como saída 0 ($110 \rightarrow 0$ dada pela regra principal). Até agora a máquina percorreu nessa ordem os

estados s_0, s_2, s_5, s_{12}, s_{18}, q_6 e retornou como saída a seguinte string: {010}. A partir do estado q_6 a máquina processa de maneira similar à Máquina de Moore genérica que modela a evolução para frente de um AC homogêneo. Assim, a sequência de estados percorrida a partir da entrada {101101010} é: $s_0 \rightarrow s_2 \rightarrow s_5 \rightarrow s_{12} \rightarrow s_{18} \rightarrow q_6 \rightarrow q_5 \rightarrow q_2 \rightarrow q_5 \rightarrow q_2$ e a saída final retornada pela máquina é a sequência {0100101}. Se for necessário calcular duas ou mais evoluções para frente, o processo se repete. Ou seja, o primeiro e o último bit devem ser copiados e a máquina reinicia a partir do estado s_0 novamente.

5.2 Modelagem do cálculo de pré-imagem como máquina de Moore

O cálculo de pré-imagem em ACs com regra de transição com sensitividade às extremidades, como o utilizado na cifragem dos modelos de Gutowitz, HCA e outros discutidos no Capítulo 4, também é possível ser modelado por uma máquina de Moore. Por ser composto por vizinhanças deterministas, o cálculo de pré-imagens em ACs com regras sensíveis a um dos extremos, é um processo sequencial e determinista. Dessa forma, também é possível modelá-lo por um autômato finito com saída. A evolução para frente vista na seção anterior é modelada sempre pelo mesmo grafo, sendo que sua topologia não se altera de uma regra de transição para outra, tal que a única diferenciação é relacionada à saída que deve ser associada a cada um dos estados. No cálculo da pré-imagem, por outro lado, a topologia do grafo varia de acordo com a regra de transição do AC.

A Figura 5.4 apresenta uma regra genérica de transição com sensitividade à direita de raio $r = 1$:

000	001	010	011	100	101	110	111
b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7

Figura 5.4: Regra de transição sensível à direita.

Pela propriedade da regra temos que $b_0 = \overline{b_1}$, $b_2 = \overline{b_3}$, $b_4 = \overline{b_5}$ e $b_6 = \overline{b_7}$. Suponha que se deseje calcular a partir de um reticulado inicial, uma pré-imagem que tenha r células a mais de cada lado e que tenha as $2r$ células iniciais à esquerda especificadas com seus estados já conhecidos. Por exemplo, a Figura 5.5, que apresenta o início de uma pré-imagem a do reticulado 10110110, onde se escolheu arbitrariamente os valores 0 e 1 para inicializar as células P_1 e P_2.

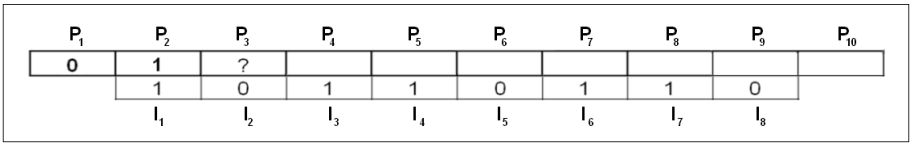


Figura 5.5: Cálculo de pré-imagem.

O que se deseja é estabelecer uma regra inversa que define para cada vizinhança parcial de 2 células e o bit de saída no reticulado inicial, qual será o valor do próximo bit à direita na vizinhança. Assim, a partir da regra de transição direta, que determina a evolução temporal do AC para frente, deseja-se obter uma espécie de regra inversa que calcula a pré-imagem de forma sequencial. Essa regra deve ser tal que atenda:

00?	00?	01?	01?	10?	10?	11?	11?
1	0	1	1	0	1	1	0

Figura 5.6: Regra inversa.

Se os 2 bits à esquerda da vizinhança parcial são P_1 e P_2 e o bit de saída é I_1, podemos escrever uma regra que forneça para cada tripla (P_1, P_2, I_1) o valor de P_3, ou seja,

$$\begin{array}{ccc} P_1 & P_2 & I_1 \\ & \downarrow & \\ & P_3 & \end{array}$$

Se a sequência de bits de saída da regra sensível à direita é $b_0b_1b_2b_3b_4b_5b_6b_7$ e a mesma tem sensitividade à direita, temos a seguinte regra indireta apresentada na Figura 5.7:

000	001	010	011	100	101	110	111
b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7

Figura 5.7: Regra indireta.

ou seja, é a mesma regra definida na Figura 5.4 (que representa a evolução para frente mas com uma formação de vizinhança diferente). Nesse caso, os dois primeiros bits da "vizinhança" da regra vêm dos bits da pré-imagem P_1 e P_2 e o terceiro bit vem do reticulado inicial I_1.

Assim, se a regra direta for a regra 89:

000	001	010	011	100	101	110	111
1	0	0	1	1	0	1	0

Figura 5.8: Regra 89.

temos que a mesma regra pode ser utilizada como regra inversa, com a diferença que a vizinhança é elaborada com os 2 bits iniciais da vizinhança seguidos com o valor do estado da célula central no reticulado inicial e o bit de saída se refere ao estado da próxima célula à direita no reticulado.

Portanto, dada a regra direta, a mesma regra pode ser utilizada como inversa. Entretanto, na elaboração do grafo que representa o diagrama da Máquina de Moore, as transições de estado variam de acordo com os bits da regra. Isso ocorre porque quando o cálculo sequencial é efetuado, a passagem de uma vizinhança para outra sofre alteração de duas células, sendo que uma delas é o último estado calculado na vizinhança anterior e a outra é o próximo bit lido no reticulado inicial.

Por exemplo, no cálculo de uma célula P_3 na Figura 5.5, a vizinhança formada é $P_1P_2I_1$. Na sequência, para o cálculo da célula P_4 na Figura 5.5, a vizinhança é formada por $P_2P_3I_2$. Ou seja, ela não depende apenas do próximo bit lido no reticulado inicial (I_2), mas também do último bit calculado (P_3), que é dado pela regra de transição.

Assim, dada uma regra direta com sensitividade à direita, como na Figura 5.4.
sua regra inversa é equivalente à regra direita e um grafo genérico da Máquina de Moore é apresentado na Figura 5.9.

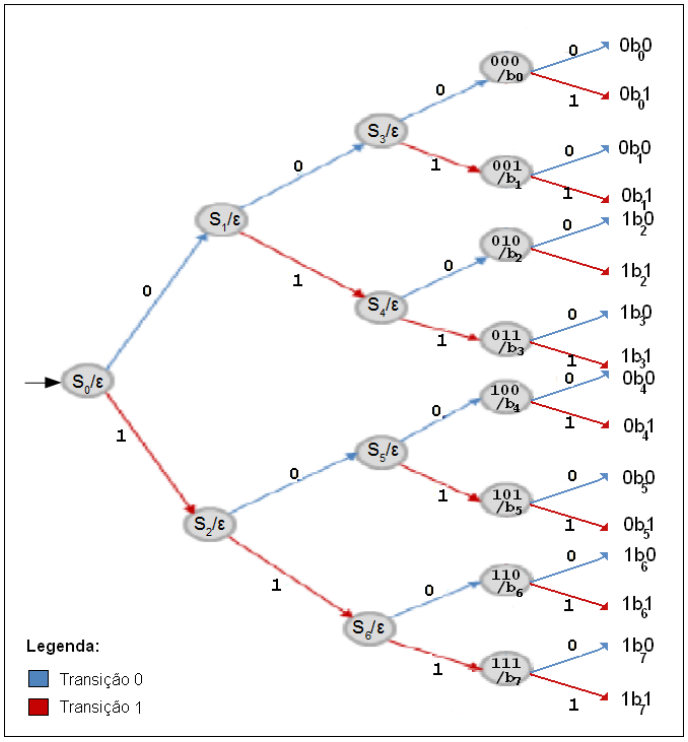


Figura 5.9: Máquina de Moore genérica para o cálculo de pré-imagens.

Observando-se o grafo da Figura 5.9 fica claro que as transições entre os estados na extremidade do grafo (000 à 111) dependem dos bits de saída da regra inversa, que são os mesmos da regra direta. Portanto, embora o conjunto de estados e a estrutura do grafo para as três primeiras leituras não se modifique, as transições na extremidade do grafo, que

modelam o processamento de todas as células da pré-imagem, a partir da segunda célula calculada, são modificadas para cada regra dada. As Figuras 5.10 (a) e (b) exemplificam os grafos obtidos para o cálculo de pré-imagem a partir de duas regras: a regra 30, sensível à esquerda e a regra 89, sensível à direita.

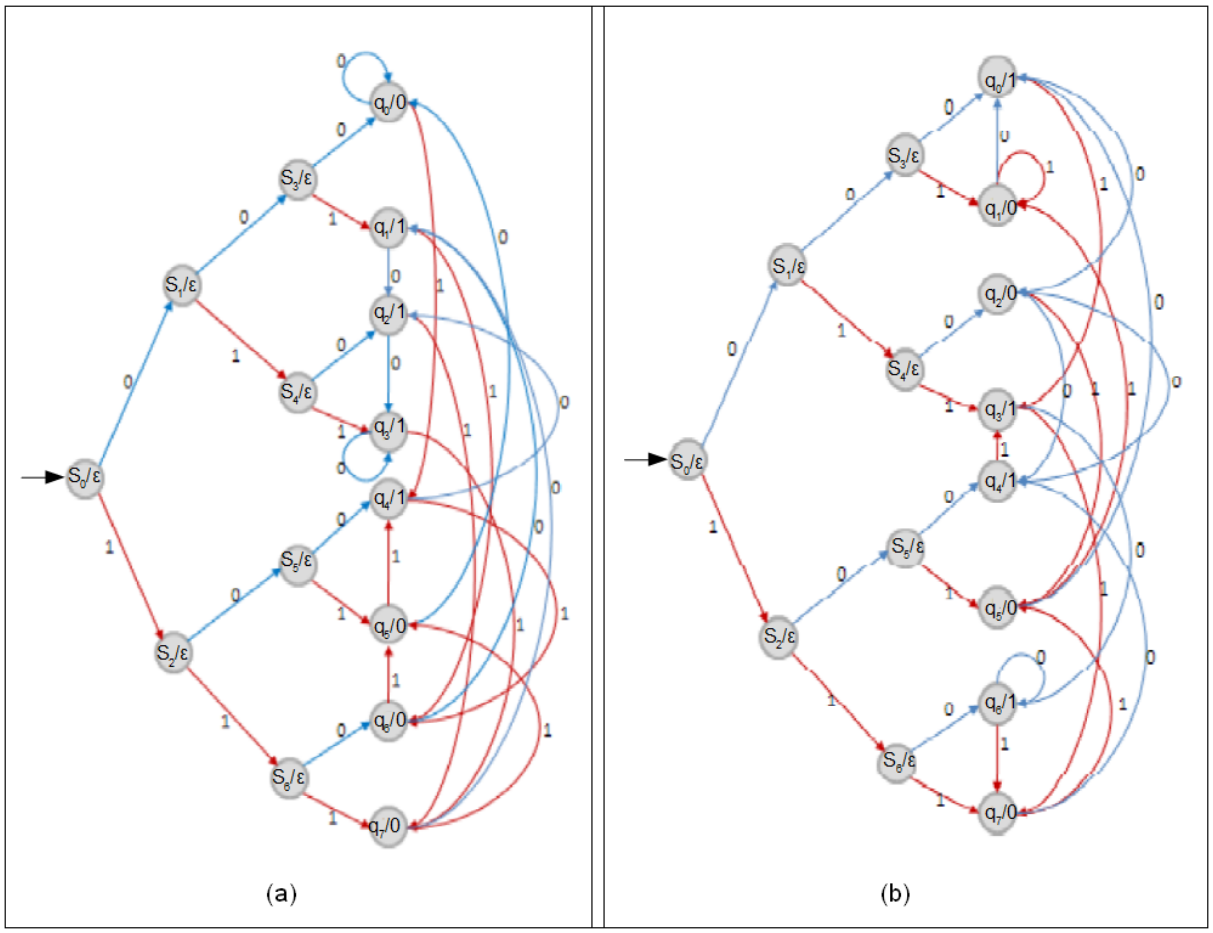


Figura 5.10: (a) Máquina de Moore relativa à execução *backward* da Regra 30. (b) Máquina de Moore relativa à execução *backward* da Regra 89.

O Apêndice D apresenta um algoritmo elaborado nessa dissertação para a construção de uma Máquina de Moore que modela o cálculo de pré-imagem com uma regra sensível a um dos extremos (esquerda ou direita).

A adaptação da máquina de Moore para o modelo de AC híbrido no HCA, com uso da regra de contorno também é simples e similar ao discutido para o AC padrão. A Figura 5.11 apresenta a máquina de Moore que modela a evolução para trás do AC híbrido utilizando a regra 30 como regra principal e a regra 15 como regra de contorno. Nesse caso, temos que os estados s_i com $i > 0$ retornam a saída da regra de contorno. O mesmo não é observado na máquina de Moore para a evolução *backward* para o AC padrão, onde os estados s_i com $i > 0$ retornam ε que é a mesma saída retornada pelo estado s_0. As saídas dos estados q_j com $0 \leq j \leq 2^m$ tanto para o AC padrão quanto para o AC híbrido, representam a saída da regra principal. É possível perceber que a transição dos estados

s_i para os estados q_i não é definida em função dos três últimos bits lidos, como foi feito no AC padrão da Figura 5.10. Essa transição é definida em função dos dois últimos bits gravados combinados com o último bit lido. Por exemplo, se o estado atual é s_3 significa que os dois últimos bits lidos foram "00" e os dois últimos gravados foram "11". Nesse caso, ao se ler o bit 0, por exemplo, a transição é para o estado q_3 que representa a vizinhança "110" e não para o estado q_0, que representa a vizinhança "000".

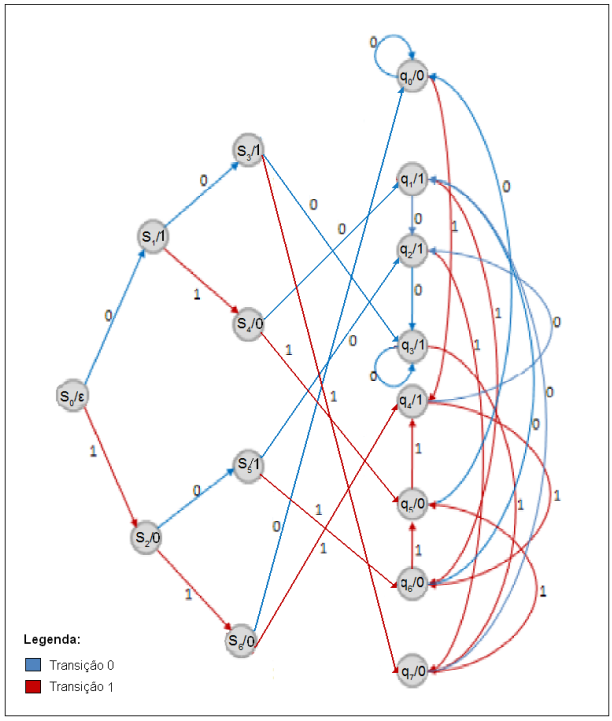


Figura 5.11: Máquina de Moore da evolução *backward* para o AC híbrido do modelo HCA.

Para exemplificar um passo do cálculo de pré-imagem utilizando a máquina de Moore para AC híbrido a Figura 4.11 será novamente utilizada. A regra principal utilizada é a regra 30 {01111000} e a regra de contorno é a regra 15 {11110000}. Inicialmente a máquina se encontra no estado inicial s_0 e tem como fita de entrada a cadeia {0100101}. Em seguida, ela processa o primeiro símbolo da fita de entrada (reticulado inicial). O primeiro símbolo que será lido é o 1 que está posicionado abaixo de P_2 na Figura 4.11. Assim, a máquina vai para o estado s_2 e retorna 0 como saída (regra de contorno). Essa saída deve ser copiada para a posição P_1 na fita de saída. O próximo símbolo da fita de entrada que deve ser lido, é o símbolo 0 que está localizado abaixo de P_1 na Figura 4.11. A partir do processamento desse símbolo, a máquina vai para o estado s_5 e retorna 1 como saída (regra de contorno). Essa saída deve ser gravada para a posição P_7 da fita de saída. Como o reticulado inicial que representa a fita de entrada tem contorno periódico, o próximo símbolo da fita de entrada que deve ser lido é o 1 que está posicionado abaixo da célula P_7. A partir do processamento desse símbolo, a máquina vai para o estado q_6 e retorna a saída 0 (regra principal) que deve ser copiada para a célula P_6. O processamento segue da mesma maneira para os demais símbolos da fita de

entrada, sendo que o último símbolo que será processado é o símbolo 1 que está abaixo da célula P_3 e copiará a saída para a célula P_2. A máquina percorrerá, nessa ordem, os estados $s_0 \rightarrow s_2 \rightarrow s_5 \rightarrow q_6 \rightarrow q_1 \rightarrow q_6 \rightarrow q_1 \rightarrow q_2$ e retornará como saída a sequência: $\{0110101\}$ que representa a pré-imagem relativa ao reticulado inicial. Caso duas ou mais pré-imagens forem necessárias o processo se repete e a máquina volta para o estado inicial s_0 novamente.

Embora os grafos obtidos pelo modelo HCA, como o da Figura 5.11 sejam diferentes dos grafos obtidos com o AC homogêneo, como os da Figura 5.10 a conclusão de nosso estudo é que devemos nos concentrar na porção cíclica dos grafos, representada pelos nós q_i e suas transições. Nesse caso, não existe distinção entre o grafo da Figura 5.10 (a) e a Figura 5.11, pois ambos se baseiam na regra 30. A segurança do método deve ser analisada em relação a essa parte cíclica, que representa o processamento da regra principal no método HCA.

5.3 Análise da segurança do HCA

O RSA [Rivest et al., 1978] é um algoritmo de criptografia de chaves assimétricas fundamentado na teoria dos números. A segurança do RSA advém da dificuldade de se fatorar números grandes, sendo este um problema NP-completo.

De forma similar, um dos objetivos dessa dissertação é compreender o algoritmo de criptografia HCA e analisar sua segurança buscando associá-lo a um problema NP-completo, uma vez que essa associação é uma forma de ganhar a evidência para a segurança de um modelo criptográfico. Ou seja, deve-se reduzir o problema a um problema NP-completo, fazendo a seguinte analogia: se é difícil computacionalmente resolver um problema NP-Completo, também será difícil quebrar a segurança do algoritmo de criptografia, caso a força do algoritmo analisado esteja ligada de fato a esse problema NP-completo.

Como foi possível modelar a principal operação da etapa de cifragem o cálculo de pré-imagem como um grafo, acreditamos que problemas estudados na Teoria dos Grafos, possam nos ajudar a encontrar essa associação a um problema NP, auxiliando de alguma forma na análise da segurança desse modelo criptográfico.

A partir da análise dos grafos associados a regras sensíveis, como as empregadas no método HCA, foi possível observar uma propriedade comum a todos os grafos: os grafos são hamiltonianos, possuindo sempre dois circuitos hamiltonianos. Nessa análise, a parte inicial do grafo (estados s_i, onde $0 \leq i < 2^m - 1$) foram desconsiderados, uma vez que essa parte do grafo é utilizada apenas para "*decidir*" em qual dos estados q_i, o processamento da palavra será iniciado. Dessa forma, os grafos analisados foram reduzidos à porção cíclica do grafo, formada apenas pelos estados que representam as vizinhanças e que gravam o bit de saída da regra de transição do AC associado a cada estado (estados q_i na

Figura 5.1).

Como exemplo, no grafo da Figura 5.10 (a), associado à regra 30, existe o circuito hamiltoniano $q_0 \rightarrow q_4 \rightarrow q_6 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_7 \rightarrow q_5 \rightarrow q_0$ e outro $q_0 \rightarrow q_4 \rightarrow q_2 \rightarrow q_3 \rightarrow q_7 \rightarrow q_1 \rightarrow q_6 \rightarrow q_5 \rightarrow q_0$, enquanto que no grafo da Figura 5.10 (b) associado à regra 89, o circuito hamiltoniano é dado por $q_0 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3 \rightarrow q_6 \rightarrow q_7 \rightarrow q_5 \rightarrow q_1 \rightarrow q_0$ e o segundo por $q_0 \rightarrow q_3 \rightarrow q_6 \rightarrow q_7 \rightarrow q_4 \rightarrow q_2 \rightarrow q_5 \rightarrow q_1 \rightarrow q_0$.

Além da constatação dessa propriedade, podemos observar que percorrer o circuito hamiltoniano no caso do método HCA, equivale à decifrar a chave criptográfica, uma vez que ao passar por todos os estados q_i, todos os bits de saída da regra seriam gravados na fita de entrada, não necessariamente na ordem da regra.

Assim, analisando-se um único passo de cálculo de pré-imagem, caso soubéssemos qual é o circuito hamiltoniano dos estados (ou a sequência de entradas que forçam as transições), seria possível descobrir os bits da chave.

Entretanto, sabe-se que, dado um grafo, descobrir se o mesmo é hamiltoniano ou descobrir qual é o circuito hamiltoniano em um grafo são problemas NP-completos. Por outro lado, se fosse possível quebrar a chave com facilidade seria possível descobrir o circuito hamiltoniano, que sabe-se ser um problema NP-completo para um grafo arbitrário.

No entanto, os grafos que modelam o cálculo de pré-imagens do HCA como uma Máquina de Moore compartilham uma estrutura muito particular. Por exemplo, cada vértice possui exatamente duas arestas de entrada e duas arestas de saída. Além disso, o número de vértices é fixo e depende do tamanho da regra de transição, que depende por sua vez do raio adotado na vizinhança. Em um estudo subsequente que dê continuidade a essa análise, é necessário que essas e outras propriedades dos grafos que representam as Máquinas de Moore associadas ao método HCA sejam consideradas para se analisar se a busca do circuito hamiltoniano encontrado nesse tipo de grafo é de fato um problema NP-completo. As propriedades estruturais específicas desses grafos poderiam tornar a busca por esse circuito um problema de menor complexidade. Além disso, ainda que se prove que o problema continua NP-completo, mesmo considerando-se as especificidades desses grafos, também é necessário verificar se a NP-completude pode de fato assegurar a segurança do método. Sabe-se que a NP-completude é usada apenas para fornecer evidências de que problemas são intratáveis, no entanto, não há garantia de que o problema intratável ao qual o algoritmo criptográfico esteja ligado (através da redução), forneça uma prova da segurança do modelo de criptografia analisado.

Assim, na análise iniciada nessa dissertação constatamos que: (i) o processo básico do método de cifragem (o cálculo de uma pré-imagem) pode ser modelado por um grafo; (ii) esses grafos possuem a propriedade de serem hamiltonianos e de possuírem dois circuitos hamiltonianos; (iii) o conhecimento desses circuitos está associado à chave secreta; (iv) sabe-se que o problema de encontrar circuitos hamiltonianos em grafos genéricos é um problema NP-completo. Entretanto, não podemos garantir que essas constatações por si

só já comprovam a segurança do método HCA. Mas, elas iniciam uma análise que pode ser aprofundada em um trabalho de pesquisa posterior.

5.4 Análise da segurança dos métodos THCA e 3DHCA

Embora não seja apresentado nessa dissertação, vislumbramos também ser possível construir máquinas de Moore para modelar o cálculo de pré-imagens de ACs bidimensionais e tridimensionais que também utilizem regras com sensitividade a uma célula no extremo da vizinhança. Esse tipo de regra e de cálculo de pré-imagem foi utilizado na elaboração do modelo THCA [Magalhaes, 2010] e no modelo 3DHCA elaborado na presente dissertação e apresentado no próximo capítulo. Dessa fora, acreditamos que com as devidas adaptações necessárias aos espaços bidimensional e tridimensional, os resultados alcançados analisando o modelo precursor HCA poderão ser ampliados para os modelos THCA e 3DHCA. Deixamos essa análise como proposta para trabalhos futuros que venham a dar continuidade à presente investigação.

Capítulo 6

Modelo de criptografia tridimensional 3DHCA

O modelo proposto na presente dissertação de mestrado é um modelo de criptografia em blocos. Modelos de criptografia com essa característica utilizam um particionamento do texto total a ser cifrado, sendo que cada um desses blocos são cifrados de forma isolada e posteriormente eles são recombinados de maneira a obter o texto cifrado. Entretanto, devido à utilização da técnica de recombinação, parte das propriedades de confusão e difusão do modelo podem ser perdidas. Particularmente, quando aplicado à criptografia de imagens, a divisão de uma imagem em blocos que são cifrados independentemente e que são recombinados de forma a permitir um paralelismo desses processos, pode levar a uma imagem cifrada com zonas de textura. Por exemplo, a Figura 6.1 apresenta um exemplo de cifragem em bloco no qual uma imagem foi particionada em pequenos blocos para a realização da criptografia. Após a cifragem de cada bloco separadamente, os blocos cifrados foram recombinados para formar a imagem cifrada. Nessa imagem é possível perceber que mesmo com a cifragem independente dos blocos, a imagem cifrada mantém características da imagem original, as chamadas zonas de texturas.

Por isso, uma das características desejáveis em um método de criptografia voltado à cifragem de imagens é que ele possa cifrar uma imagem como um bloco único, sem utilização de técnicas de recombinação de blocos. Imagens, por outro lado, são blocos de informação de tamanho razoavelmente grande. Por exemplo, uma imagem preto e branco de 512×512 pixels possui 2^{18} bits, enquanto que uma imagem em escala de cinza de 512×512 pixels possui 2^{21} bits e outra colorida (padrão HSI ou RGB) possui 3×2^{21} bits. Os métodos de criptografia convencionais operam com blocos lineares de tamanho bem mais modestos: DES, 2DES e 3DES (64 bits) e AES (128 bits). Assim, a recombinação de blocos acaba sendo uma prática usual quando empregamos métodos tradicionais que usam blocos de texto unidimensionais. Além disso, ainda que um método criptográfico seja adaptado para se operar com blocos de maior tamanho, deve-se levar em conta que a cifragem de um único bloco será computacionalmente intensiva, e é importante que o

modelo permita uma implementação com alto nível de paralelismo.

Dessa forma, os autômatos celulares tornam-se candidatos naturais a modelos para criptografia de imagens. A possibilidade de trabalhar com o arranjo de bits em qualquer dimensão, além da unidimensional, faz com que eles possam operar nas dimensões mais adequadas à manipulação de imagens. Em [Magalhaes, 2010], o arranjo bidimensional das imagens, através do emprego de ACs com reticulados em duas dimensões se mostrou bastante adequado à cifragem de imagens preto e branco. Entretanto, a performance do método caiu ao ser aplicado em imagens em escala de cinza e se tornou bastante limitado na cifragem de imagens coloridas. Dessa forma, essa dissertação dá continuidade aos modelos HCA [Macedo, 2007] e THCA [Magalhaes, 2010], investigando uma adaptação dos mesmos ao espaço tridimensional. O novo método foi chamado de 3DHCA (*Three-dimensional Hybrid Cellular Automata*): modelo criptográfico baseado em ACs tridimensionais híbridos.

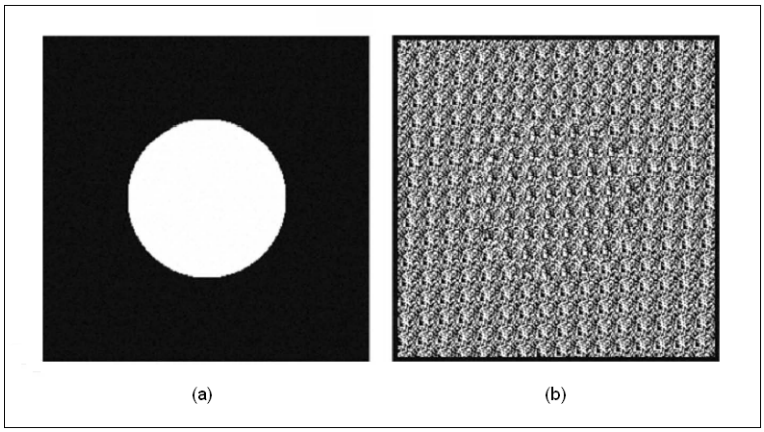


Figura 6.1: (a) Imagem original. (b) Imagem cifrada por blocos resultando em zonas de texturas [Wu et al., 2010].

O principal objetivo da proposição de um novo modelo criptográfico baseado em ACs tridimensionais é posibilitar a cifragem de uma imagem colorida em um único bloco. Entretanto, devido á alta dimensionalidade das imagens coloridas é importante que o método tenha uma rápida propagação da entropia e um alto nível de paralelismo. Com o emprego de ACs tridimensionais caóticos temos a possibilidade de arranjar os bits da imagem original de forma a estreitar a distância máxima entre os pontos extremos de forma a facilitar a propagação das entropia. Além disso, o arranjo tridimensional permite a obtenção de um nível de paralelismo ainda maior que os obtidos com ACs unidimensionais e bidimensionais.

Nesse capítulo, o modelo tridimensional será apresentado de forma detalhada, desde a versão básica inicial até a versão final que se mostrou mais eficiente.

6.1 Imagens digitais e padrão RGB

Como o modelo de criptografia foi desenvolvido com o intuito principal de ser empregado em imagens coloridas, uma breve definição de imagem digital com padrão RGB será apresentada. Uma imagem digital é a representação de uma imagem bidimensional usando números binários codificados de modo a permitir seu armazenamento, transferência, impressão ou reprodução, e seu processamento por meios eletrônicos. As imagens digitais utilizadas neste trabalho possuem o padrão de cores RGB. Podemos entender RGB como a abreviatura do sistema de cores aditivas formado por Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*). O propósito principal do sistema RGB é a reprodução de cores em dispositivos eletrônicos como monitores de TV e computador, *scanners* e câmeras digitais, assim como nas imagens digitais tradicionais.

Uma cor no modelo de cores RGB pode ser descrita pela indicação da quantidade de vermelho, verde e azul contida em um pixel, conforme apresentado na Figura 6.2. Assim, um pixel da imagem é uma posição do plano e três valores são associados a ele, cada um refrente a um dos três canais RGB. No caso da Figura 6.2, o primeiro pixel da imagem está destacado na posição (0,0) e vemos que esse pixel tem valores R(230), G(26) e B(224) e a cor resultante é a composição desses três canais, próxima à cor Magenta puro. Cada um dos canais pode variar entre o mínimo (completamente escuro) e máximo (completamente intenso). Quando todos os canais de cor estão no mínimo, a cor resultante é preta. Se todos os canais de cor estão no máximo, a cor resultante é branca. Uma das representações mais usuais para as cores é a utilização da escala de 0 à 255, bastante encontrada na computação pela conveniência de se guardar cada valor de cor em 1 byte (8 bits). Dessa forma, cada pixel da imagem é associado à três bytes (ou 24 bits), sendo um byte referente a cada canal. As cores mais conhecidas no padrão RGB, são representadas conforme indicado na Tabela 6.1.

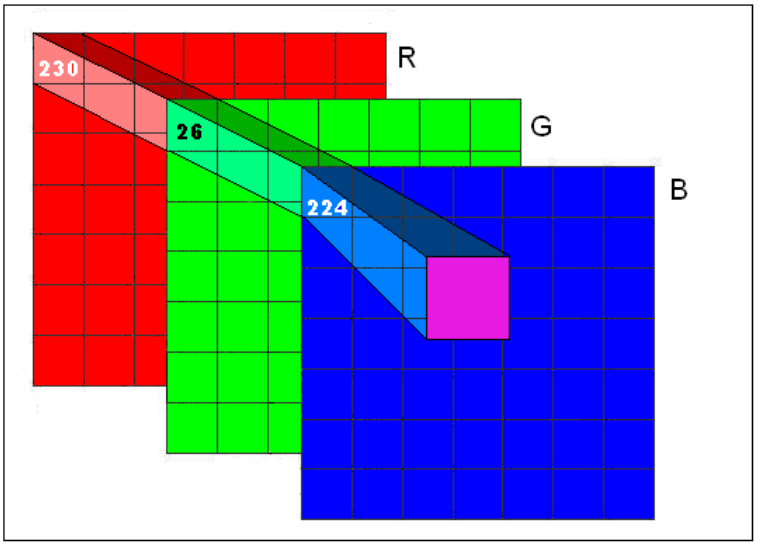


Figura 6.2: Exemplo de imagem RGB

Tabela 6.1: Principais cores e respectivas representações padrão RGB.

Cor	Representação	Cor	Representação
Branco	RGB(255,255,255)	Amarelo	RGB(255,255,0)
Azul	RGB(0,0,255)	Magenta	RGB(255,0,255)
Vermelho	RGB(255,0,0)	Ciano	RGB(0,255,255)
Verde	RGB(0,255,0)	Preto	RGB(0,0,0)

6.2 Base do modelo tridimensional

A primeira etapa desse trabalho consistiu em estudar os modelos de criptografia antecessores [Gutowitz, 1995], [Lima, 2005], [Macedo, 2007] e [Magalhaes, 2010]. A partir desse estudo, passamos a buscar um novo modelo de criptografia que melhorasse aspectos que não foram abordados em [Magalhaes, 2010]. A principal melhoria esperada em nosso modelo é a propagação da perturbação ao longo do espaço tridimensional, sem perder o foco que o método deve processar em alto nível de paralelismo, objetivando a implementação do modelo em um *hardware* massivamente paralelo, por exemplo. No novo modelo, as principais características propostas em [Macedo, 2007] são preservadas a saber: a utilização de um AC híbrido com duas regras de transição (regra principal e regra de contorno), bem como o processo de cifragem ser realizado a partir do cálculo de pré-imagem.

Assim como nos modelos criptográficos unidimensional e bidimensional em [Macedo, 2007] e [Magalhaes, 2010], o reticulado tridimensional, formado após a transformação do espaço 2D para o espaço 3D, possui contorno periódico, ou seja, o último plano é vizinho do primeiro e vice-versa. Essa periodicidade aplica-se tanto aos planos verticais quanto aos horizontais. A vizinhança é definida pelo modelo de Von Neumann tridimensional apresentada na Seção 2.3. Para facilitar a representação e implementação das regras de transição, a regra principal e a regra de contorno foram construídas a partir de uma transformação do espaço tridimensional para o unidimensional, respeitando-se a ordem imposta na Figura 6.3 (b), tal que X representa a saída da regra principal e Y representa a saída da regra de contorno. Entretanto, apesar dessa representação unidimensional, a aplicação da mesma no reticulado tridimensional segue as direções representadas na Figura 6.3 (a). Nos exemplos apresentados nessa dissertação, a regra principal utilizada no processo de cifragem é sensível ao bit norte (N) indicado na Figura 6.3 (a) e essa sensitividade deve ser estendida à regra de contorno que deve conter a mesma sensitividade da regra principal.

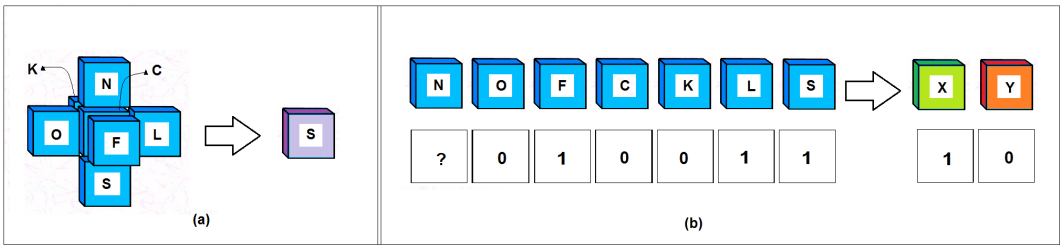


Figura 6.3: (a) Domínio e Contradomínio da vizinhança tridimensional de raio 1. (b) Transformação da vizinhança tridimensional para uma vizinhança unidimensional.

Um dos aspectos que diferencia este modelo em relação aos modelos antecessores é a cardinalidade do espaço de chaves para um mesmo tamanho de regra. A quantidade de formas para realizar a cifragem (direção do cálculo) no novo modelo é maior, pois neste caso existem seis tipos de sensitividade enquanto que em [Macedo, 2007] existem apenas dois tipos (esquerda e direita) e [Magalhaes, 2010] existem quatro tipos (norte, sul, leste e oeste). Por exemplo, no caso de regras unidimensionais de 128 bits (vizinhança de raio 3), 64 bits da regra podem ser utilizados para gerar 2 chaves diferentes (sensível à esquerda ou sensível à direita). Por outro lado, no caso de regras tridimensionais de 128 bits (vizinhança de raio 1), os 64 bits podem gerar 6 chaves diferentes.

Até que se chegasse à versão final do modelo tridimensional, diversas alterações foram realizadas a fim de refinar o modelo. A seguir serão apresentadas as principais versões implementadas. A não ser quando especificado, por simplicidade, os exemplos que serão apresentados nas próximas seções utilizam regras de raio 1 sensíveis ao bit do norte.

6.3 Modelo básico com três blocos

A primeira versão do método que foi elaborada consistiu em adaptar o modelo bidimensional proposto em [Magalhaes, 2010] (com borda e núcleo fixos) para um modelo tridimensional básico. Inicialmente, foram utilizados três blocos de dados para serem cifrados, cada um representando um canal de cores (R, G e B), e duas regras foram utilizadas para a realização da cifragem dos três blocos: a regra principal e a regra de contorno. Assim, para uma imagem $m \times n$ (largura n e altura m) no padrão RGB são cifrados três blocos binários tridimensionais de tamanho $m \times n \times 8$.

O processo de cifragem inicia-se pela fragmentação da imagem RGB (*red, green, blue*) de tamanho $m \times n$ (pixels) em três matrizes bidimensionais inteiras $m \times n$. Cada uma dessas matrizes representa um canal de cores e cada célula da matriz contém um número inteiro entre 0 e 255. Cada componente de cor do pixel de uma imagem RGB é formado por três *bytes* que estão associados a uma das cores na seguinte ordem: vermelho, verde e azul. Em seguida, a matriz bidimensional inteira é transformada em uma matriz tridimensional binária de dimensões $8 \times m \times n$.

Como exemplo, a Figura 6.4 mostra uma imagem de tamanho 5×5 pixels. Na Figura 6.5 essa imagem foi transformada em três matrizes bidimensionais, onde cada uma de suas células são preenchidas por números inteiros. Neste caso, temos uma matriz que representa o canal vermelho (R), uma matriz representa o canal verde (G) e outra que representa o canal azul (B). A transformação da matriz bidimensional inteira para uma matriz tridimensional binária é realizada transformando cada célula das matrizes bidimensionais R, G e B em um número binário. Um exemplo de transformação para o espaço tridimensional do primeiro pixel da imagem é indicado na Figura 6.6. O pixel destacado na Figura 6.6 possui os valores R = 255, G = 0 e B = 0. Essa transformação é realizada de tal maneira que o bit menos significativo ocupe o plano superior e o bit mais significativo o plano inferior.

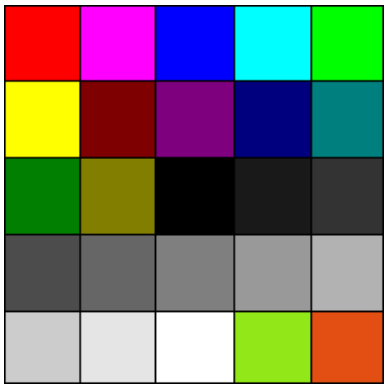


Figura 6.4: Exemplo de uma imagem RGB de tamanho 5×5 pixels.

A Figura 6.7 mostra caracterização das bordas dos modelos: unidimensional (a), bidimensional (b) e tridimensional (c). No modelo unidimensional, a quantidade de células utilizadas para a borda corresponde a $2r$. Já no modelo bidimensional, a quantidade de células que correspondem à borda (N_b) é apresentada na Equação 6.1, sendo que n representa a largura da imagem, m representa a altura da imagem e r é o raio da regra que está sendo utilizada como chave criptográfica.

$$N_b = 2rm + 2rn - 4r^2 \tag{6.1}$$

R					G					B				
255	255	0	0	0	0	0	0	255	255	0	255	255	255	0
255	127	127	0	0	255	0	0	0	127	0	0	127	127	127
0	130	0	25	51	127	127	0	25	51	0	0	0	25	51
76	102	127	153	178	76	102	127	153	178	76	102	127	153	178
204	229	255	146	227	204	229	255	231	79	204	229	255	24	18

Figura 6.5: Exemplo da imagem fragmentada em três matrizes R, G e B de ordem 5.

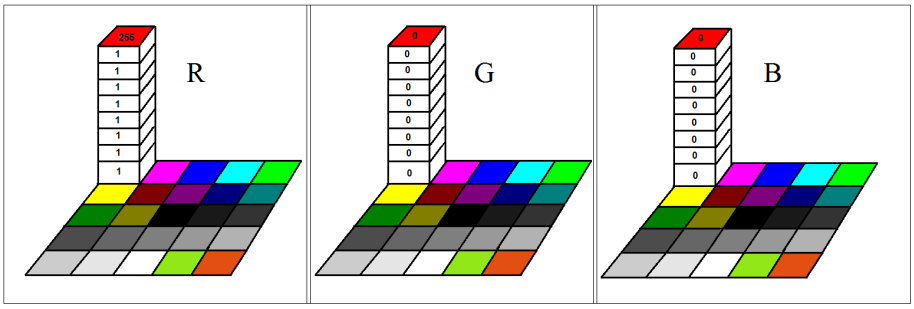


Figura 6.6: Exemplo de imagem bidimensional transformada em um reticulado tridimensional binário para o canal R.

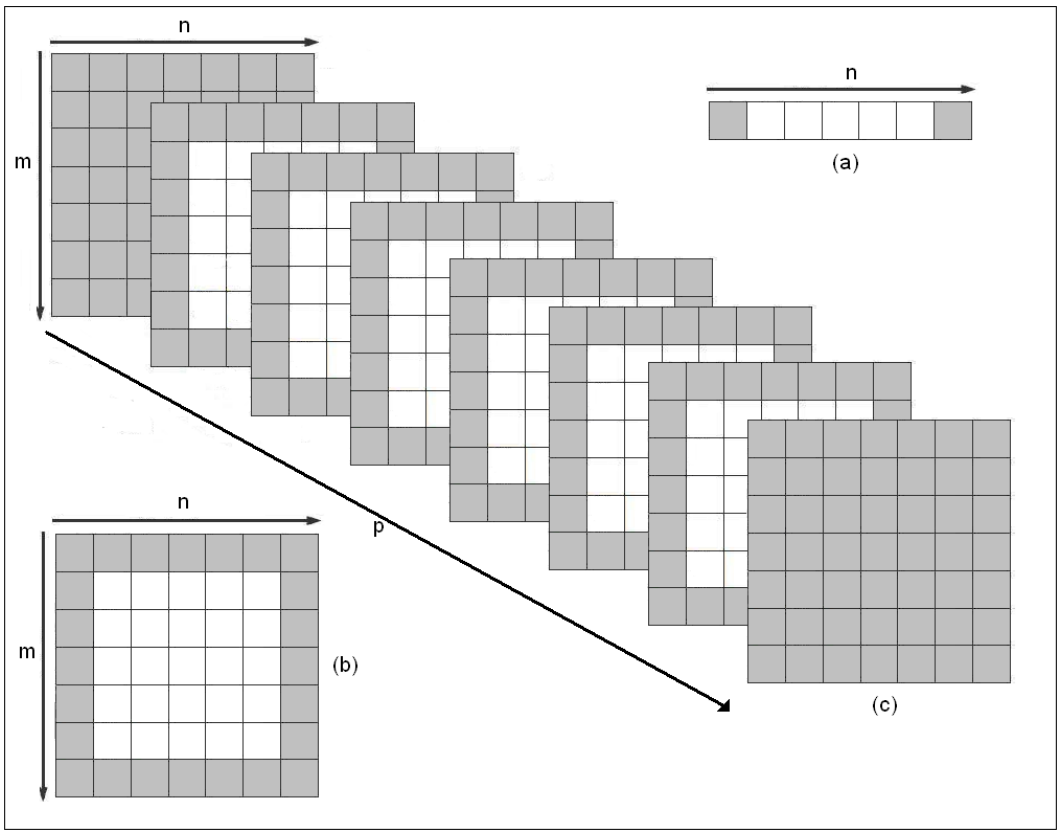


Figura 6.7: (a) Quantidade de células da borda do modelo unidimensional. (b) Quantidade de células da borda do modelo bidimensional. (c) Quantidade de células da borda do modelo tridimensional.

A quantidade de células que compõem a borda de cada um dos blocos do modelo tridimensional básico pode ser calculada conforme a Equação 6.2, que foi derivada da Fórmula 6.1, proposta em [Magalhaes, 2010]. O cálculo é feito levando em consideração a largura (n), altura (m) e profundidade (p) do cubo e o raio (r) da regra utilizada:

$$N_b = 2rmn + (p - 2r) \times (2rm + 2rn - 4r^2) \tag{6.2}$$

Seja n a largura da imagem, m a altura, p a profundidade do bloco tridimensional e r o raio da regra utilizada e considerando $p = 8$, visto que, os três blocos têm profundidade

8, então a quantidade total de células que compõem as bordas em cada um dos três blocos binários tridimensionais é indicada na Equação 6.3:

$$N_b = 3 \times (2rmn + (8 - 2r) \times (2rm + 2rn - 4r^2)) \tag{6.3}$$

6.3.1 Cálculo de pré-imagens no modelo básico com três blocos

Como exemplo, apresentaremos um passo de cálculo da pré-imagem do canal R da Figura 6.4. Serão utilizados duas regras tridimensionais geradas a partir dos núcleos apresentados na Tabela 6.2. Esses núcleos compõem o conjunto de chaves criptográficas utilizadas durante a cifragem. A partir de cada núcleo, uma regra tridimensional com sensitividade ao bit norte foi gerada. Cada uma das vizinhanças e suas respectivas saídas são apresentadas na Tabela 6.3, segundo a linearização da vizinhança tridimensional representada na Figura 6.3 (b), na qual a coluna X apresenta o bit de saída para a regra principal gerada a partir do núcleo correspondente na Tabela 6.2, enquanto a coluna Y apresenta o bit de saída da regra de contorno. Como as regras são sensíveis ao bit norte (primeiro bit da vizinhança na representação unidimensional) é possível perceber que a sequência de bits na primeira metade da Tabela 6.3 (vizinhança 0000000 a 0111111 ou índice 0 a 63) corresponde exatamente à sequência de bits no núcleo, tanto para a regra principal quanto para a regra de contorno. Por outro lado, a segunda metade da tabela (índice 64 a 127) apresenta uma sequência binária que é o complemento dos núcleos. Dessa forma, temos duas regras geradas sensíveis ao bit norte, criadas a partir dos núcleos da Tabela 6.2.

Tabela 6.2: Núcleos das regras utilizadas.

Tipo	**Núcleo**
Regra Principal	01011111001001111110011011101000 01111101101110010010110110100000
Regra de Contorno	11111111111111111111111111111111 11111111111111111111111111111111

A Figura 6.8 apresenta um exemplo de cálculo de pré-imagem utilizando-se a matriz tridimensional binária obtida a partir da transformação do canal R da imagem da Figura 6.4 e as regras apresentadas na Tabela 6.3. Como a imagem tem 5×5 pixels, a matriz tridimensional referente ao canal R é formada por $5 \times 5 \times 8$ bits. Para facilitar a visualização do processo, a Figura 6.8 apresenta a matriz tridimensional R dividida em 8 planos de tamanho 5×5. A coluna "Índice" da Figura 6.8 mostra os índices de cada um dos oito planos da altura do reticulado tridimensional. O plano com menor índice (plano 0) possui os bits menos significativos, enquanto que o plano com maior índice (plano 7) possui os

bits mais significativos. A coluna "Matriz R" da Figura 6.8 apresenta os bits de cada um dos 8 planos do reticulado inicial correspondente ao canal R da Figura 6.4, para o qual se deseja calcular uma pré-imagem.

O cálculo da pré-imagem inicia-se com a determinação da borda do reticulado tridimensional. Essa borda é composta pelo piso, paredes e teto do reticulado tridimensional. O piso do reticulado tridimensional corresponde a todas as células (bits) do plano 7 e o teto corresponde a todas células do plano 0, destacados em negrito na coluna "Borda" da Figura 6.8. As paredes que compõem a borda do reticulado representam as primeiras e últimas linhas e colunas dos planos internos do reticulado 3D (índice 1 a 6) que também estão destacados em negrito na coluna "Borda" da Figura 6.8. O cálculo da pré-imagem é iniciado pelo cálculo dos valores dos bits das bordas da pré-imagem. A regra de contorno e detalhada pelo valor de saída Y corresponde a cada vizinhança na Tabela 6.3 é tal que o bit da célula central da vizinhança será o complemento do bit do norte numa evolução para frente do reticulado do AC. Essa característica faz com que no cálculo da pré-imagem de um reticulado, haja um deslocamento espacial norte seguida de uma complementação dos valores de cada um dos bits, neste caso, direção norte, pois foi empregada uma regra com sensitividade ao norte. Esse processo é mostrado na coluna "Borda" da Figura 6.8 que apresenta apenas o cálculo dos bits na borda da pré-imagem.

Para detalhar o processo apresentado na Figura 6.8, vamos analisar como os bits da borda referente ao piso da pré-imagem (plano índice 7) são obtidos. Esses bits são calculados a partir dos bits do plano mais ao sul (que são deslocados para o norte). Como esse plano é o piso, e o contorno do reticulado é periódico, o plano de índice 0 é considerado mais ao sul do plano índice 7. Os bits do plano de índice 0 no reticulado inicial são (coluna "Matriz R" na Figura 6.8):

1	1	0	0	0
1	1	1	0	0
0	0	0	1	1
0	0	1	1	0
0	1	1	0	1

Tabela 6.3: (X) Bit de saída da regra principal (Y) Bit de saída da regra de contorno.

Índice	Vizinhança	X	Y	Índice	Vizinhança	X	Y
0	0000000	0	1	64	1000000	1	0
1	0000001	1	1	65	1000001	0	0
2	0000010	0	1	66	1000010	1	0
3	0000011	1	1	67	1000011	0	0
4	0000100	1	1	68	1000100	0	0
5	0000101	1	1	69	1000101	0	0
6	0000110	1	1	70	1000110	0	0
7	0000111	1	1	71	1000111	0	0
8	0001000	0	1	72	1001000	1	0
9	0001001	0	1	73	1001001	1	0
10	0001010	1	1	74	1001010	0	0
11	0001011	0	1	75	1001011	1	0
12	0001100	0	1	76	1001100	1	0
13	0001101	1	1	77	1001101	0	0
14	0001110	1	1	78	1001110	0	0
15	0001111	1	1	79	1001111	0	0
16	0010000	1	1	80	1010000	0	0
17	0010001	1	1	81	1010001	0	0
18	0010010	1	1	82	1010010	0	0
19	0010011	0	1	83	1010011	1	0
20	0010100	0	1	84	1010100	1	0
21	0010101	1	1	85	1010101	0	0
22	0010110	1	1	86	1010110	0	0
23	0010111	0	1	87	1010111	1	0
24	0011000	1	1	88	1011000	0	0
25	0011001	1	1	89	1011001	0	0
26	0011010	1	1	90	1011010	0	0
27	0011011	0	1	91	1011011	1	0
28	0011100	1	1	92	1011100	0	0
29	0011101	0	1	93	1011101	1	0
30	0011110	0	1	94	1011110	1	0
31	0011111	0	1	95	1011111	1	0
32	0100000	0	1	96	1100000	1	0
33	0100001	1	1	97	1100001	0	0
34	0100010	1	1	98	1100010	0	0
35	0100011	1	1	99	1100011	0	0
36	0100100	1	1	100	1100100	0	0
37	0100101	1	1	101	1100101	0	0
38	0100110	0	1	102	1100110	1	0
39	0100111	1	1	103	1100111	0	0
40	0101000	1	1	104	1101000	0	0
41	0101001	0	1	105	1101001	1	0
42	0101010	1	1	106	1101010	0	0
43	0101011	1	1	107	1101011	0	0
44	0101100	1	1	108	1101100	0	0
45	0101101	0	1	109	1101101	1	0
46	0101110	0	1	110	1101110	1	0
47	0101111	1	1	111	1101111	0	0
48	0110000	0	1	112	1110000	1	0
49	0110001	0	1	113	1110001	1	0
50	0110010	1	1	114	1110010	0	0
51	0110011	0	1	115	1110011	1	0
52	0110100	1	1	116	1110100	0	0
53	0110101	1	1	117	1110101	0	0
54	0110110	0	1	118	1110110	1	0
55	0110111	1	1	119	1110111	0	0
56	0111000	1	1	120	1111000	0	0
57	0111001	0	1	121	1111001	1	0
58	0111010	1	1	122	1111010	0	0
59	0111011	0	1	123	1111011	1	0
60	0111100	0	1	124	1111100	1	0
61	0111101	0	1	125	1111101	1	0
62	0111110	0	1	126	1111110	1	0
63	0111111	0	1	127	1111111	1	0

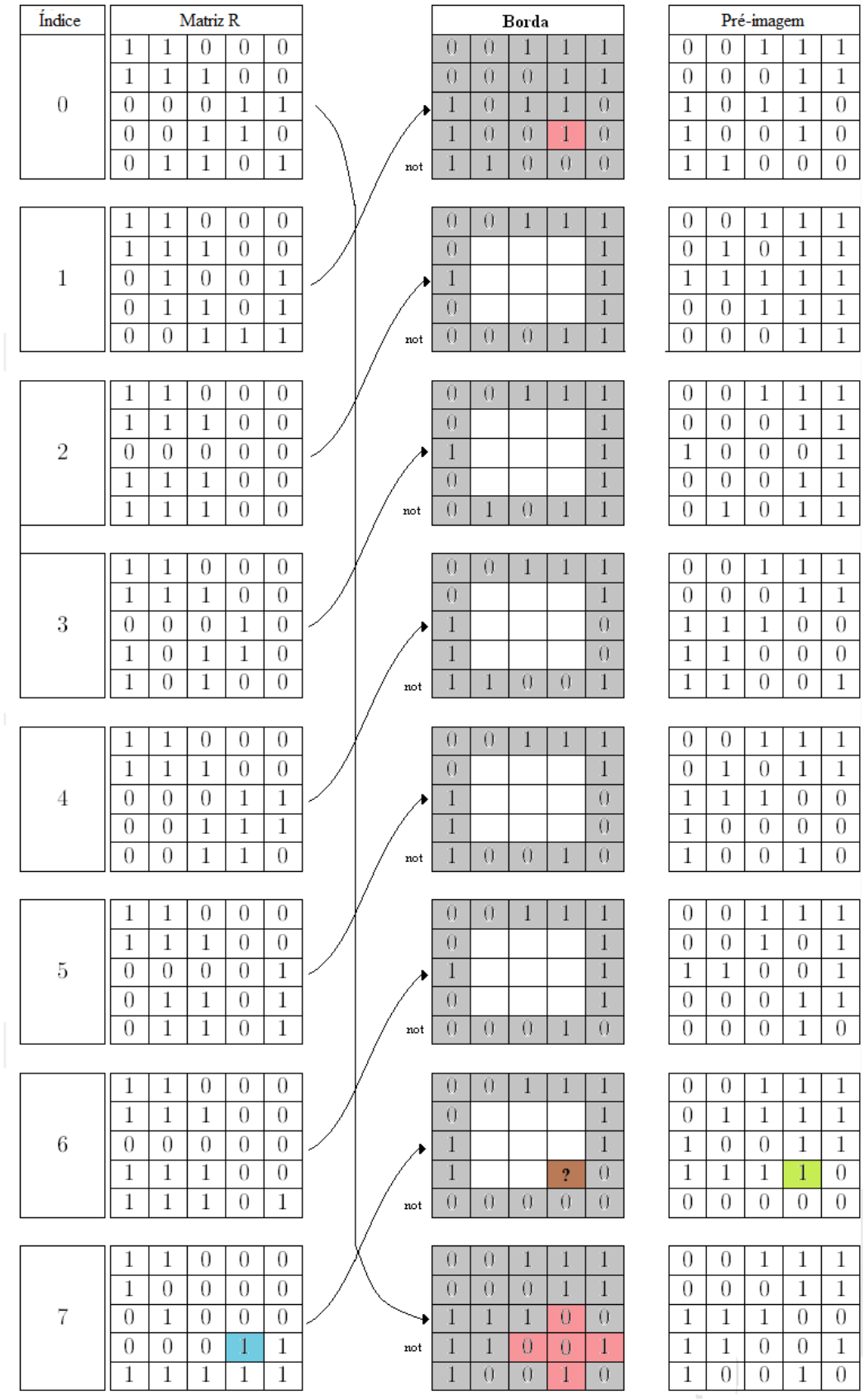


Figura 6.8: Exemplo de cálculo da pré-imagem para raio 1 e sensitividade ao norte

Portanto, os bits do plano índice 7 da pré-imagem serão dados pelo complemento desse plano (coluna "Borda" na Figura 6.8):

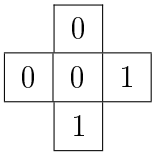
0	0	1	1	1
0	0	0	1	1
1	1	1	0	0
1	1	0	0	1
1	0	0	1	0

De forma similar, a Figura 6.8 apresenta os valores de todos os bits da borda da pré-imagem, sempre obtidos pelo complemento dos bits do plano no reticulado original (coluna "Matriz R") mais ao sul em relação ao plano da pré-imagem sendo calculado. Assim, observe na Figura 6.8 que os bits das paredes do plano (índice 6), da coluna "Matriz R", são trazidos complementados para o plano de índice 5, da coluna "Borda". Esse procedimento é repetido para todos os demais planos do reticulado, exceto em dois: o plano de índice zero e o último plano (índice 7). O plano com índice 0, da coluna "Borda", recebe todos os bits complementados do plano com índice 1, da coluna "Matriz R", e este representa o topo da borda do reticulado. Enquanto que o último plano (índice 7), conforme visto detalhadamente, recebe todos os bits invertidos do plano (índice 0), da coluna "Matriz R", formando o piso da borda do reticulado tridimensional.

Calculados os bits da borda, a próxima etapa consiste em calcular os bits centrais do reticulado a partir da regra principal. Chamaremos essas células de bits internos para diferenciá-las da borda. Os bits internos também são calculados obedecendo a direção do cálculo imposta pela sensitividade das regras. No plano índice 6 da coluna "Borda" da Figura 6.8 foi destacada com o símbolo "?", para representar o cálculo do próximo bit interno, após o cálculo das células que dependem apenas da regra de contorno. Para calcular o valor do bit "?" é preciso identificar qual é a vizinhança na qual ele representa o bit ao norte. Essa vizinhança é apresentada na Figura 6.9, que segue a mesma convenção apresentada na Figura 6.3. A seguir, detalhamos como os valores desses bits são obtidos. Essa vizinhança é elaborada a partir dos bits da borda calculados na Figura 6.8. Os bits do plano abaixo do bit "?" (que está no plano índice 6) são obtidos no plano índice 7. Pela convenção apresentada na Figura 6.3, os bits que formam o plano abaixo do bit norte são:

	K	
O	C	L
	F	

No caso da vizinhança do bit "?", os valores desses bits, que estão destacados em vermelho no plano índice 7 da coluna "Borda", são:



O valor da célula S da vizinhança é dada pelo bit na mesma posição de "?" em dois planos abaixo do plano 6, ou seja, devido ao contorno periódico, no plano 0. O bit S está destacado em vermelho no plano de índice 0 e tem valor 1. Assim temos a vizinhança tridimensional representada na Figura 6.9 (a). Se linearizarmos essa vizinhança obteremos a vizinhança ?010011 apresentada na Figura 6.9 (b).

Uma vez definida a vizinhança parcial do bit "?" na pré-imagem que está sendo calculada é necessário verificar o valor do bit de saída no reticulado original, referente ao valor da célula central. Assim o valor do bit de saída (X) é dado pelo valor da célula da mesma posição da célula C no plano 7 do reticulado original (coluna "Matriz R"), destacado em azul, que tem o valor 1. Assim, para descobrir o valor da célula marcada com "?" é necessário verificar na regra principal qual vizinhança atende ?010011 $\rightarrow$ 1. A única vizinhança que atende a restrição ?010011 $\rightarrow$ 1, de acordo com a Tabela 6.3, é a vizinhança 1000111, portanto o estado da célula marcada com '?' é 1.

De forma similar, os demais bits da região central do reticulado tridimensional são calculados até a obtenção da pré-imagem completa. Na Figura 6.8, o resultado obtido na última coluna "Pré-imagem" corresponde aos planos da pré-imagem da matriz tridimensional R representada na coluna "Matriz R". As matrizes tridimensionais G e B podem ser cifradas de forma similar utilizando as mesmas regras (principal e contorno).

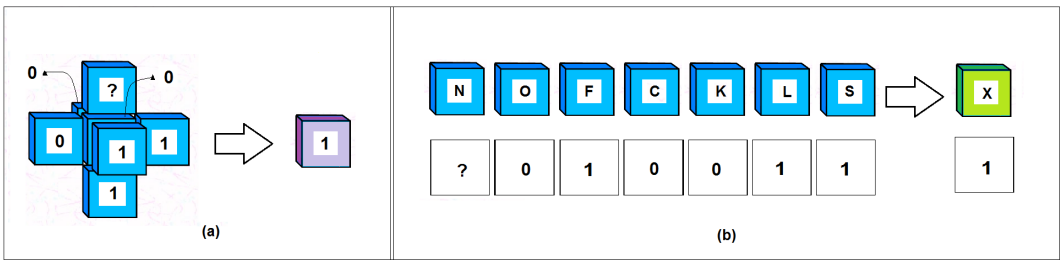


Figura 6.9: Representação da vizinhança do bit identificado por "?".

Para visualizar a imagem cifrada, basta realizar uma transformação dimensional, transformando cada número binário no seu respectivo decimal. Dessa maneira, são criadas três matrizes bidimensionais que combinadas entre si geram uma imagem do mesmo tamanho da imagem original. A imagem cifrada referente ao processo de cálculo de pré-imagem após 1 passo de tempo descrito anteriormente pode ser visualizada na Figura 6.10.

No exemplo anterior, a imagem foi cifrada fazendo a evolução para trás por um único passo de tempo, ou seja, apenas uma pré-imagem foi calculada. Entretanto, esse processo pode ser repetido por T passos de tempo, obtendo-se T pré-imagens consecutivas.

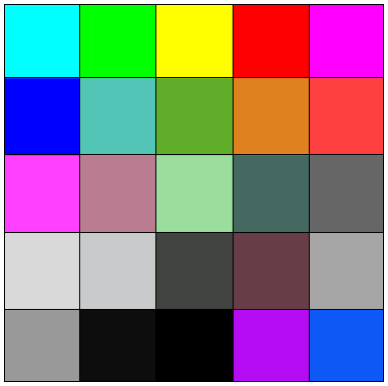


Figura 6.10: Exemplo de uma imagem cifrada por regra de raio 1 e sensitividade ao norte.

6.3.2 Processo de decifragem no modelo básico com três blocos

O processo de decifragem é realizado pela evolução *forward* do reticulado tridimensional pelos mesmos T passos utilizados no processo de criptografia. No caso da cifragem do exemplo anterior, temos $T = 1$. A decifragem é realizada de forma síncrona (paralela) para todas as células de cada um dos reticulados tridimensionais (R, G e B) e de tal forma que a execução de cada um desses reticulados também seja realizada de forma simultânea. A Figura 6.11 exemplifica o processo de decifragem. A coluna "Imagem cifrada" da Figura 6.11 apresenta o reticulado encriptado que se deseja decifrar, correspondente à matriz do canal R da imagem cifrada na Figura 6.10. Na evolução para frente, a borda é calculada no sentido inverso ao processo utilizado no cálculo da pré-imagem, ou seja, o cálculo é realizado deslocando-se os bits complementados (devido à regra de contorno) no sentido norte-sul. Assim, a borda é dada pelo primeiro e segundo planos e pelas paredes dos demais planos, conforme apresentado na coluna "Borda" da Figura 6.11. Embora a evolução para frente possa ser realizada de forma síncrona, uma abordagem sequencial será utilizada para exemplificar o processo de decifragem. Assim, suponha que vamos calcular primeiro o bit representado por "?" na terceira coluna da Figura 6.11. O bit "?" representa o bit de saída da vizinhança em vermelho na coluna "Imagem cifrada" da Figura 6.11. Para facilitar a visualização, a Figura 6.12 apresenta essa vizinhança na forma tridimensional. A partir da regra principal da Tabela 6.3, utilizando-se a linearização $1000000 \rightarrow ?$ vemos que o símbolo "?" deve ser preenchido por 1 (em destaque no plano de índice 2 da coluna "Matriz R" da Figura 6.11). De maneira análoga, os demais bits do reticulado que representa a matriz R original são calculados. Como todas as vizinhanças dependem unicamente da coluna "Imagem cifrada", todos os bits da coluna "Matriz R" podem ser obtidos simultaneamente.

Embora a borda seja apresentada separadamente na Figura 6.11, é importante destacar que, diferentemente da cifragem, todos os bits podem ser calculados de forma síncrona. Isso, porque todas as células do reticulado dependem unicamente dos estados da imagem cifrada.

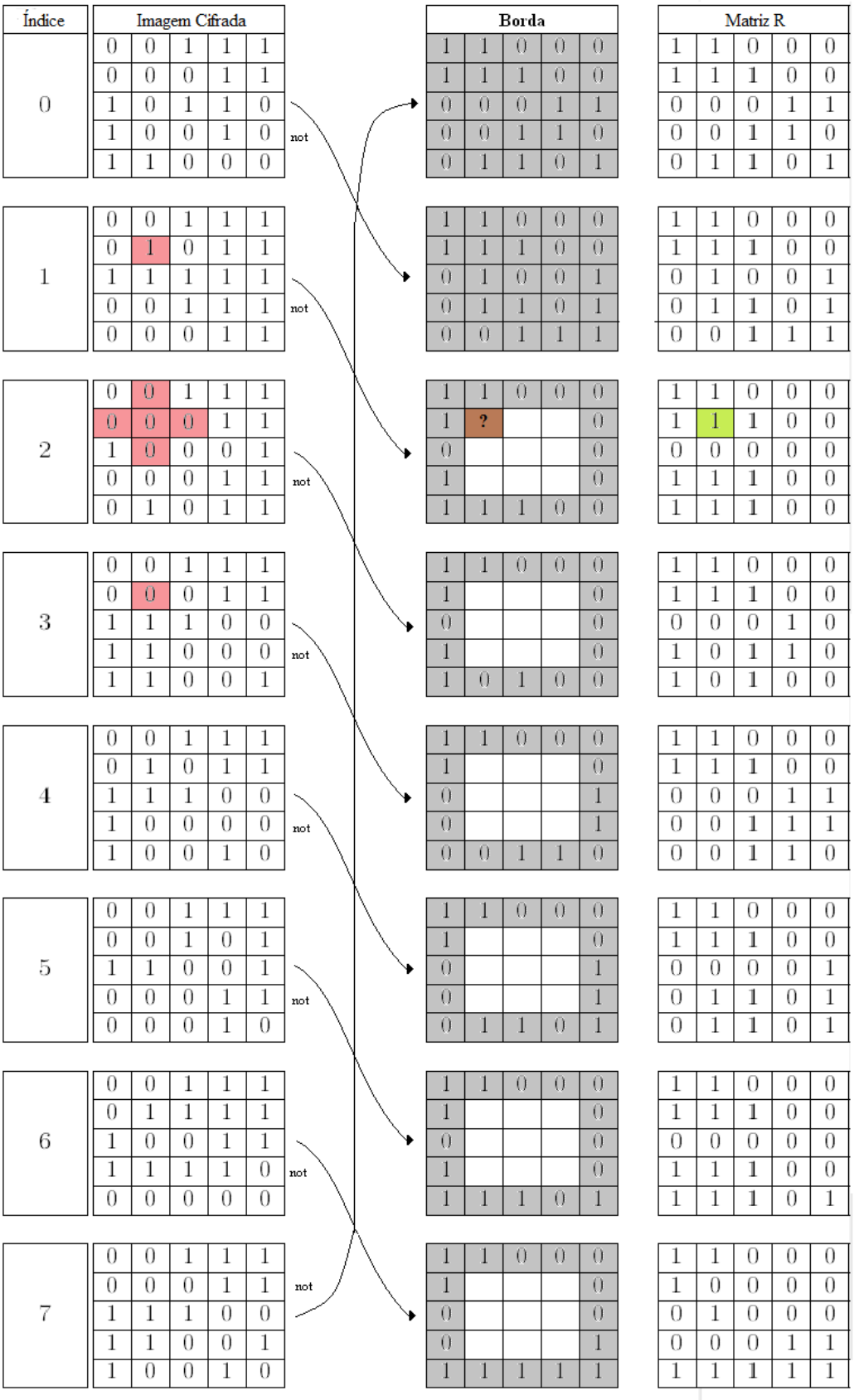


Figura 6.11: Exemplo de decifragem (execução *forward*) para raio 1 e sensitividade ao norte

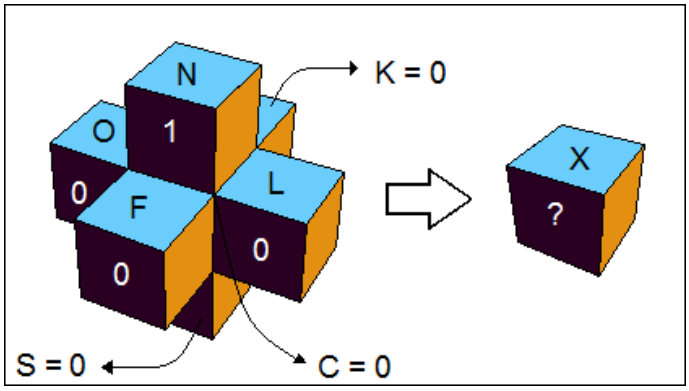


Figura 6.12: Representação da vizinhança.

6.3.3 Análise do paralelismo no modelo básico com três blocos

A maior vantagem de se utilizar autômatos celulares na criptografia é a possibilidade de se paralelizar o processo. Utilizando-se o modelo básico com três blocos, a etapa de decifragem é realizada pela simples evolução para frente do AC. A quantidade de tempo necessária para realizar essa tarefa é dada pelo número de passos (T) utilizados na cifragem, visto que todas as células do reticulado são atualizadas de forma síncrona. Ou seja, se implementado em um *hardware* paralelo, a decifragem seria executada em exatamente T ciclos de relógio de processamento.

O processo de cálculo de pré-imagem, que representa a etapa de cifragem, não é possível de ser realizado de forma totalmente simultânea, mas possui um certo nível de paralelismo. O cálculo de uma única pré-imagem ($T = 1$), conforme foi observado na Figura 6.8, inicia quando todos os bits de contorno são calculados. As células de borda do reticulado podem ser calculadas ao mesmo tempo. Em seguida, todas as células internas (células que não pertencem à borda) de um mesmo plano podem ser atualizadas de forma síncrona. A título de exemplo, todas as células internas pertencentes ao mesmo plano (índice 6) que a célula preenchida por "?", podem ser atualizadas de forma paralela. As células internas do plano com índice 5 devem esperar um ciclo de *clock* para serem calculadas (em relação ao plano índice 6). Esse processo é repetido até que todas as células internas do último plano sejam totalmente preenchidas. Seja, r o raio da regra que está sendo utilizada e p a profundidade do bloco tridimensional, temos que NC_{C3D} indica qual é número de *clocks* necessário em um processamento paralelo para cifrar uma único bloco no modelo básico tridimensional, dado pela Equação 6.4:

$$NC_{C3D} = T \times (1 + p - 2r) \tag{6.4}$$

Como já foi discutido anteriormente, temos 3 blocos (R, G e B) nesse modelo, cada um com profundidade $p = 8$, então a quantidade de passos necessários para realizar a cifragem é dada pela Equação 6.5 se consideramos que os três blocos são cifrados em paralelo. A quantidade de passos se torna dependente apenas do raio da regra e da quantidade de

passos que foram aplicados para realizar a cifragem sendo independente do tamanho da imagem $m \times n$. Isso ocorre porque cada plano da imagem tem tamanho $m \times n$ e pode ser calculado de forma síncrona.

$$NC_{C3D} = T \times (9 - 2r) \tag{6.5}$$

Conforme analisado anteriormente, a quantidade de tempo para a decifragem é mostrada na Equação 6.6.

$$NC_{D3D} = T \tag{6.6}$$

Como exemplo, uma única pré-imagem para Figura 6.4 poderia ser calculada em $NC_{C3D} = 1 \times (1 + 8 - 2) = 7$ ciclos de relógio e a decifragem poderia ser calculada em $NC_{D3D} = 1$ ciclo de relógio de processamento.

No modelo proposto em [Magalhaes, 2010], a fórmula para a quantidade de ciclos de *clock* na cifragem do THCA é mostrado na Equação 6.7 e a quantidade de ciclos na decifragem do modelo bidimensional vale T, esse valor é o mesmo gasto na decifragem do modelo tridimensional.

$$NC_{C2D} = T \times (1 + m - 2r) \tag{6.7}$$

A fim de contrastar o modelo bidimensional com o tridimensional, considere um exemplo, onde uma regra de raio 2 é utilizada para cifrar uma imagem de 512×512 pixels, por 50 passos de tempo utilizando-se uma arquitetura totalmente paralela. O total de células calculadas nos dois modelo é de 314572800. No modelo bidimensional essa imagem seria transformada para três blocos bidimensionais de dimensões $(512 \times 2) \times (512 \times 4)$ bits e teríamos $NC_{C2D} = 50 \times (1 + (512 \times 2) - 4) = 51050$ ciclos de relógio na cifragem para cifrar os três blocos de forma síncrona. No modelo tridimensional apresentado nessa dissertação, teríamos $NC_{C3D} = 50 \times (1 + 8 - 4) = 250$ ciclos de *clock*. Portanto, o número de ciclos de *clock* em uma implementação paralela ideal cai bastante do modelo bidimensional básico para o modelo tridimensional aqui discutido.

6.3.4 Considerações sobre o modelo básico com três blocos

Esse primeiro modelo tridimensional investigado possui um bom nível de paralelismo, devido ao fato de cada plano da matriz poder ser cifrado de forma paralela. Além disso, o paralelismo do modelo pode ser ainda incrementado pelo fato dos três canais poderem ser cifrados em paralelo. Entretanto, o modelo básico com três blocos, apresentado nessa seção, apresentou algumas deficiências nas análises iniciais. A Figura 6.13 apresenta dois exemplos de imagens cifradas utilizando o primeiro modelo básico com três blocos. Para os exemplos apresentados a seguir, foram foram utilizados 50 passos de tempo para

imagens com dimensões de 100×100 pixels. No primeiro exemplo, uma imagem com todos os pixels brancos (a) foi cifrada resultando na imagem apresentada em (b). No segundo exemplo, uma imagem preto e branco com um meio círculo (c) foi cifrada e o resultado é apresentado em (d). A primeira deficiência identificada refere-se à utilização de uma borda fixa. Quando utilizamos modelos sem deslocamento de borda, estes deixam vestígios na cifragem da imagem plana, como pode ser observado na Figura 6.13 (b) e (d). Observe que as bordas das figuras originais são mantidas nas Figuras 6.13 (b) e (d). Na Figura 6.13 (b), a borda preservada após a cifragem é totalmente branca, idêntico à informação da borda da imagem original na Figura 6.13 (a). Na Figura 6.13 (d) a borda preservada é em parte preta e branca, conforme a informação da borda da imagem original na Figura 6.13 (c). Essas informações preservadas não são adequadas para um sistema criptográfico. Esse problema foi solucionado utilizando-se um deslocamento da borda, similar ao utilizado no método unidimensional HCA, que será apresentado nas próximas seções.

Outra deficiência no modelo foi observada ao cifrar uma imagem simples, que possui um padrão de cores pouco variável, como por exemplo, imagens que possuem todos os canais completamente escuros (preto) ou completamente intensos (branco). Ao cifrar uma imagem com todos os pixels brancos, que pode ser observada na Figura 6.13 (a), notou-se que as cores dos pixels da imagem cifrada resultante na Figura 6.13 (b), variaram na escala de cinza. O mesmo pode ser notado com a cifragem da imagem preto e branco na Figura 6.13 (c), que também resultou em uma cifragem com pixels na escala cinza, como é mostrado na Figura 6.13 (d).

Esse padrão de cores na escala cinza que é observado na imagem resultante cifrada, ocorre devido ao fato de que todos os canais estão configurados inicialmente da mesma maneira, ao se aplicar uma mesma regra simultaneamente nos três canais eles permanecem com as configurações idênticas, resultando assim, em uma imagem cifrada na escala de cores cinza. Esse coloração resultante é uma característica particular do padrão RGB.

A solução adotada para resolver esse problema foi a utilização da junção obtida a partir da intercalação de cada um dos blocos que representam os canais de cores R, G e B em um único bloco. Essa solução será apresentada na próxima seção.

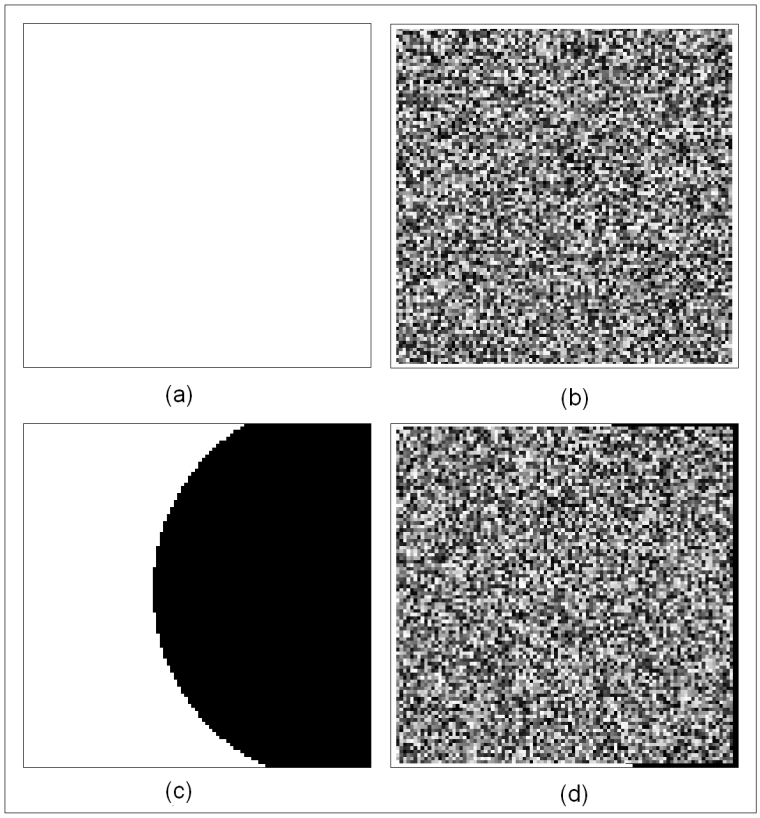


Figura 6.13: Exemplo de imagens cifradas utilizando o modelo 3D simples. (a) Imagem com todos pixels brancos. (b) Imagem "a" cifrada utilizando regra principal e de contorno indicadas na Tabela 6.2. (c) Imagem meio circulo preto e branco. (d) Imagem "c" cifrada utilizando regra principal e de contorno indicadas na Tabela 6.2.

6.4 Modelo com bloco único

Uma modificação no modelo básico exibido anteriormente foi realizada a fim de permitir a propagação ao longo dos três canais. A realização da cifragem de forma isolada em cada bloco, como visto na seção anterior, acarreta uma baixa qualidade na cifragem de algumas imagens como o problema da cifragem de imagens com poucas variações de cores. Ao cifrar imagens com os três canais organizados inicialmente da mesma maneira (imagens originais na escala cinza), no modelo anterior, a imagem cifrada resultante também apresentava as cores na escala de cinza, o que é considerado uma desvantagem do modelo, pois exibe uma característica importante da imagem original. A modificação realizada foi a junção dos três blocos binários tridimensionais (R, G e B), resultando em um único bloco para a realização do processo de cifragem/decifragem. A junção dos blocos foi feita de forma intercalada, através da sobreposição dos planos de maneira alternada de cada um dos blocos. O plano de índice 0 do bloco R é colocado no plano de índice 0 do bloco RGB, o plano de índice 0 do bloco G é colocado no plano de índice 1 do bloco RGB, e o processo é repetido até que todos os 24 planos que compõem o bloco único RGB, sejam

completamente preenchidos. O Algoritmo 3 que realiza esta tarefa é mostrado a seguir.

Algoritmo 3 Criação de bloco único com intercalação para a cifragem/decifragem.

```
for i = 0 → 7 do
    for j = 0 → m − 1 do
        for k = 0 → n − 1 do
            blocoRGB[3 ∗ i + 0][j][k] ← blocoR[i][j][k]
            blocoRGB[3 ∗ i + 1][j][k] ← blocoG[i][j][k]
            blocoRGB[3 ∗ i + 2][j][k] ← blocoB[i][j][k]
        end for
    end for
end for
```

O cálculo da borda também segue a mesma ideia adotada no modelo anterior, porém, a quantidade de células de borda (N_b) é calculada de acordo com a Equação 6.8, derivada da Equação 6.2, utilizando-se $p = 24$:

$$N_b = 2rmn + (24 - 2r) \times (2rm + 2rn - 4r^2) \tag{6.8}$$

Comparando-se as Equações 6.3 e 6.8 é possível observar uma mudança na quantidade de células da borda. No modelo discutido na Seção 6.3, uma imagem $m \times n$ é transformada em 3 blocos $m \times n \times 8$, enquanto no modelo de bloco único ela é transformada em um bloco de tamanho $m \times n \times 24$. Embora o número total de células e de planos bidimensionais seja o mesmo, o número de células na borda é diferente nos dois modelos. No primeiro modelo, 24 planos são distribuídos em 3 blocos, sendo que três planos são pisos de borda e três deles são tetos. Nos 18 planos restantes apenas as células das paredes dos planos são considerados células de borda. No segundo modelo, dos 24 planos, apenas dois são considerados piso e teto e nos demais (22) são planos onde apenas as paredes são consideradas bordas. Com isso, o número de células da borda é menor no segundo modelo e portanto o número de células internas, calculados de forma sequencial, é maior. Supondo que a imagem inicial seja a mesma utilizada na Figura 6.8, agora teríamos 24 planos do índice 0 (piso) ao índice 23 (teto) e o cálculo das células internas começando no plano de índice 22 e terminando no plano de índice 1. Se o modelo de bloco único fosse aplicado no mesmo exemplo apresentado na Seção 6.3 teríamos no total de 600 células, 198 células internas para atualização sequencial e 402 células de borda atualizadas de maneira síncrona. Por outro lado, no modelo com 3 blocos a proporção seria de 438 células para a borda e 168 células internas.

6.4.1 Cálculo de pré-imagem no modelo básico com bloco único

O processo de atualização das células no cálculo de pré-imagem nesse segundo modelo é realizado da mesma maneira que no modelo com três blocos. Entretanto, nesse modelo, teremos apenas um único bloco a ser criptografado. A principal diferença no cálculo de pré-imagens do modelo bloco único em relação ao modelo com três blocos é a quantidade de células internas (células que não pertencem à borda) que serão atualizadas, conforme visto anteriormente. Uma breve análise do paralelismo do modelo será considerada na próxima seção.

6.4.2 Análise do paralelismo no modelo básico com bloco único

O modelo básico com intercalação apresenta uma pequena diferença de cálculo em relação ao modelo anterior da quantidade de ciclos de *clock* no processo de cifragem. Considerando que p que representa a profundidade do bloco, nesse caso, $p = 24$. Assim, embora o cálculo que representa cifragem nesse modelo seja dado pela Equação 6.4 vista para o modelo de três blocos, a quantidade final de ciclos considerando-se $p = 24$ é dada pela Equação 6.9. Comparando-se com a Equação 6.5 (que considera $p = 8$), vemos que o modelo de bloco único utiliza mais ciclos para uma imagem do mesmo tamanho.

$$NC_{C3D} = T \times (25 - 2r) \tag{6.9}$$

A decifragem, assim como no modelo predecessor, é realizada em T ciclos de relógio, se realizado em uma arquitetura com alto grau de paralelismo e que permita a simultânea atualização de todas as células.

Para comparar o modelo bidimensional [Magalhaes, 2010] e os dois modelos tridimensionais apresentados até aqui, o mesmo exemplo discutido na Seção 6.3 será apresentado. Suponha que uma imagem de 512×512 pixels, seja cifrada por uma regra de raio 2 por $T = 50$ passos de tempo, utilizando-se uma arquitetura totalmente paralela. No modelo bidimensional, essa imagem seria transformada em um único bloco bidimensional com os canais de cores intercalados com as seguintes dimensões $(512 \times 3) \times (512 \times 8)$. O número de ciclos gastos pelo modelo é de $NC_{C2D} = 50 \times (1 + (512 \times 3) - 4) = 76650$ ciclos de relógio. No modelo tridimensional de bloco único, o bloco a ser cifrado teria as dimensões $24 \times 512 \times 512$ e o tempo gasto para realizar a cifragem é de $NC_{C3D(2)} = 50 \times (25 - 4) = 1050$ ciclos de relógio. No modelo tridimensional de três blocos, seriam 3 blocos cifrados de dimensões $8 \times 512 \times 512$ e o número de ciclos é dado pela Equação 6.4, é: $NC_{C3D(1)} = 50 \times (9 - 4) = 250$. O total de células calculadas para todos os modelos foi de 314572800.

A vantagem do primeiro modelo em relação ao modelo de bloco único se dá principalmente pelo fato de cada bloco de 8 bits poder ser cifrado em paralelo. Entretanto,

deve-se levar em conta que o primeiro modelo requer uma arquitetura mais robusta que o segundo, que permita que esses processos possam ser realizados efetivamente em paralelo. Se fosse utilizada uma placa do tipo FPGA [Zambreno et al., 2005], por exemplo, na implementação, o modelo de 3 blocos iria requerer uma placa com uma quantidade de unidades lógicas 200% superior ao demandado pelo modelo de bloco único.

6.4.3 Considerações sobre o modelo básico com bloco único

O modelo básico com bloco único permitiu a propagação de informação ao longo dos três canais R, G e B. Uma das consequências dessa dinâmica foi a solução do problema da cifragem de imagens muito simples. A Figura 6.14 apresenta a cifragem das mesmas imagens utilizadas na Figura 6.13. Vemos que as imagens cifradas não apresentam um padrão restrito de cores como no modelo anterior que levou à imagens em tonalidades de cinza (apenas 256 cores). Entretanto, como pode ser observado na Figura 6.14, a borda para esse tipo de imagens ainda é congelada, restando ainda essa deficiência identificada no modelo anterior. Esse tipo de característica não é adequada para métodos de criptografia, pois, um criptoanalista poderia atacar o método a partir do vestígio da imagem original que foi deixado na imagem cifrada.

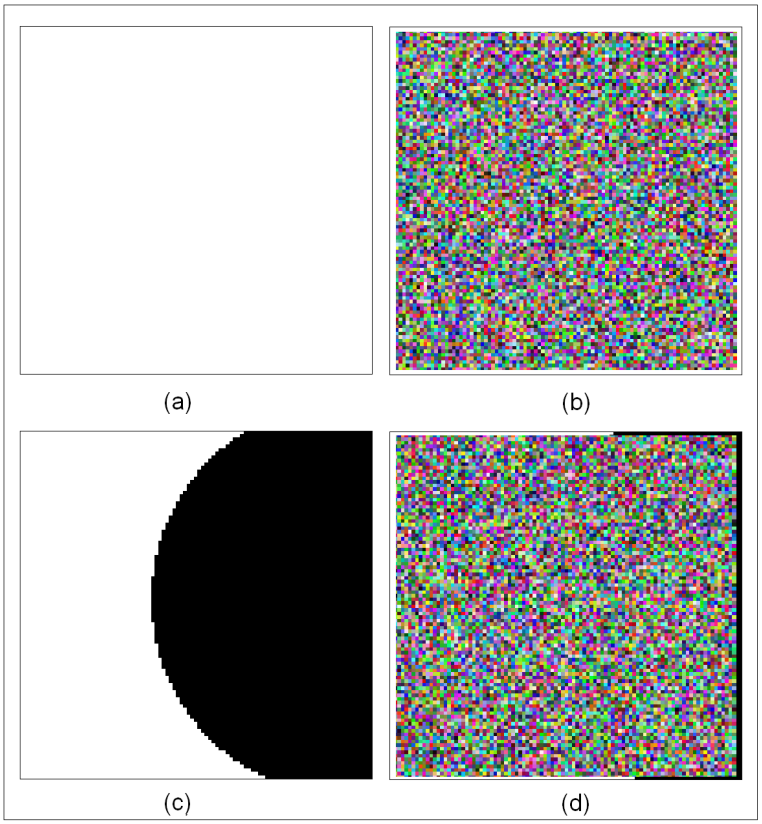


Figura 6.14: Exemplo de imagens cifradas utilizando o modelo básico com intercalação.(a) Imagem com todos pixels brancos. (b) Imagem "a" cifrada utilizando regra principal e de contorno indicadas na Tabela 6.2. (c) Imagem meio circulo preto e branco. (d) Imagem "c" cifrada utilizando regra principal e de contorno indicadas na Tabela 6.2.

Para resolver esse problema, um modelo tridimensional com deslocamento espacial da borda foi desenvolvido a fim de que a borda se mova ao longo do reticulado e isso faz com a regra de contorno não interfira negativamente no processo de cifragem. Além disso, a rotação da borda impõe um nível de paralelismo ainda maior no modelo, pois permite que os passos de pré-imagem sejam calculados de forma paralela, com um pequeno atraso entre as pré-imagens consecutivas. A nova versão para o método será apresentada na próxima seção.

6.5 Modelo com deslocamento da borda

Após concluída a primeira versão do modelo e por sua vez verificada a validade do algoritmo, a próxima etapa foi buscar um modelo em que o desempenho fosse melhorado, ou seja, que a velocidade da cifragem fosse superior em relação ao modelo básico com intercalação, utilizando-se uma arquitetura paralela.

A alteração para deslocar a borda linearmente, do reticulado tridimensional visa alterar o início do cálculo da borda que possui a mesma sensitividade da regra, de um passo de tempo para o outro, de forma a permitir que o cálculo das próximas pré-imagens possam ser iniciadas mesmo antes do termino do cálculo de uma pré-imagem anterior.

A ideia para esse deslocamento é a mesma apresentada em [Macedo, 2007] e [Magalhaes, 2010], para os modelos unidimensional (HCA) e bidimensional (THCA), respectivamente. Nos modelos básicos, quando uma pré-imagem estiver sendo calculada, sem deslocamento linear da borda principal, o cálculo de uma nova pré-imagem só pode ser iniciado quando o reticulado da pré-imagem corrente for totalmente preenchido. Esse atraso é necessário porque a obtenção dos valores das células na borda nos modelos básicos depende do cálculo do último plano do reticulado tridimensional. No exemplo de Figura 6.4 está apresentado apenas o cálculo de uma pré-imagem. Entretanto, se desejássemos realizar o cálculo da próxima pré-imagem, essa só poderia ser iniciada após o cálculo dos valores das células do plano índice 6. A próxima pré-imagem seria iniciada com o cálculo das bordas (planos de índices 0 e 7). Como os valores da borda no plano de índice 7 dependem dos valores das células do plano de índice 6 e estas por sua vez são as últimas a serem calculadas na primeira pré-imagem, o cálculo de uma pré-imagem subsequente no modelo básico só pode ser iniciado após a conclusão do cálculo da pré-imagem corrente.

Por outro lado, ao utilizar um deslocamento da borda a cada passo de tempo, o cálculo da nova borda ficará dependente somente em relação ao tamanho do deslocamento empregado. Um deslocamento de borda mínimo para que as pré-imagens sejam paralelizadas seria deslocar a borda em uma posição em relação à borda anterior na direção oposta à da sensitividade, o que provocaria um atraso de único ciclo de *clock* para que a nova pré-imagem fosse iniciada. Voltando ao exemplo da Figura 6.4, suponha que no cálculo da segunda pré-imagem, o piso e o teto da borda fossem considerados como os planos de

índices 0 e 1, respectivamente, ao invés dos planos 7 e 0 considerados na primeira pré-imagem. Como os valores das células dos planos 0 e 1 são os primeiros a serem calculados na primeira pré-imagem, o cálculo da borda da segunda pré-imagem poderia ser iniciado simultaneamente ao primeiro plano interno da pré-imagem corrente. Da mesma forma, se no cálculo da terceira pré-imagem o piso e o teto forem deslocados para os planos de índice 1 e 2, respectivamente, as células da terceira pré-imagem poderiam ser iniciadas no mesmo ciclo de *clock* em que o primeiro plano interno da segunda pré-imagem. Assim, caso o piso e o teto da borda sejam deslocados, a cada pré-imagem, em um plano (na direção contrária à sensitividade norte das regras), cada pré-imagem abaixo poderia ser iniciada em paralelo à pré-imagem antecessora, com 1 ciclo de *clock* de atraso. A Figura 6.15 exemplifica como ficaria a definição das bordas a cada pré-imagem.

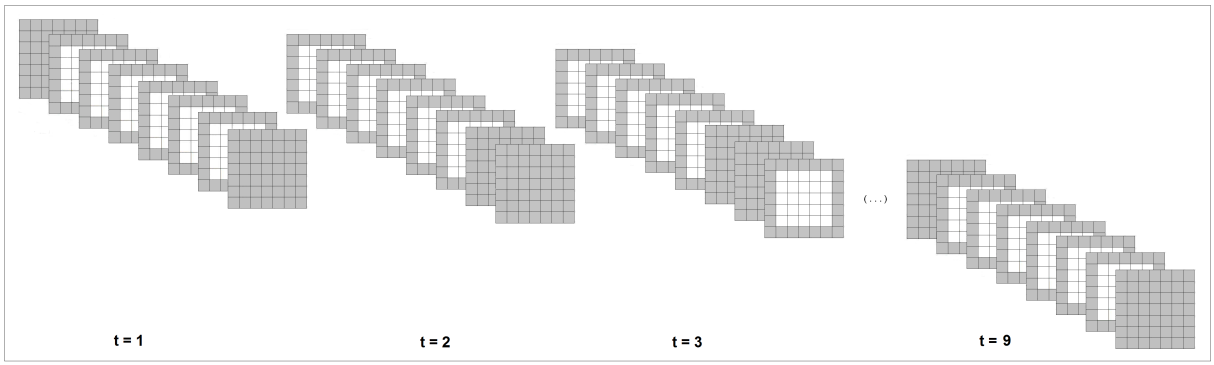


Figura 6.15: Deslocamento de uma posição da borda no mesmo eixo da sensitividade.

Na investigação do modelo bidimensional em [Magalhaes, 2010] foi observado que, embora apenas o deslocamento da borda na direção da sensitividade seja necessário para o cálculo paralelo de pré-imagens consecutivas, um deslocamento da borda no sentido ortogonal ao da sensitividade também contribui para a qualidade do modelo. Assim, em [Magalhaes, 2010] quando a regra possui sensitividade ao bit do norte, a borda é deslocada de cima para baixo de uma pré-imagem para outra e também é deslocada no eixo oeste-leste. Essa alteração não provoca nenhum atraso no cálculo da pré-imagem e se mostrou benéfica por permitir uma propagação mais rápida de qualquer alteração realizada na imagem a ser cifrada. Através de testes experimentais, verificamos que uma alteração ortogonal em relação à direção da sensitividade, também aumentou a qualidade da cifragem do modelo tridimensional. Ou seja, além do deslocamento dos planos que identificam o piso e o teto da borda a cada pré-imagem calculada, é interessante que as paredes da borda também sejam deslocadas. A Figura 6.16 apresenta como ficaria a definição das bordas, na cifragem da mesma imagem da Figura 6.4, utilizando um deslocamento da borda nas duas direções a cada pré-imagem nos três eixos: norte-sul, leste-oeste, frente-fundo.

Finalmente, foi observado nos modelos unidimensional [Macedo, 2007] e bidimensio-

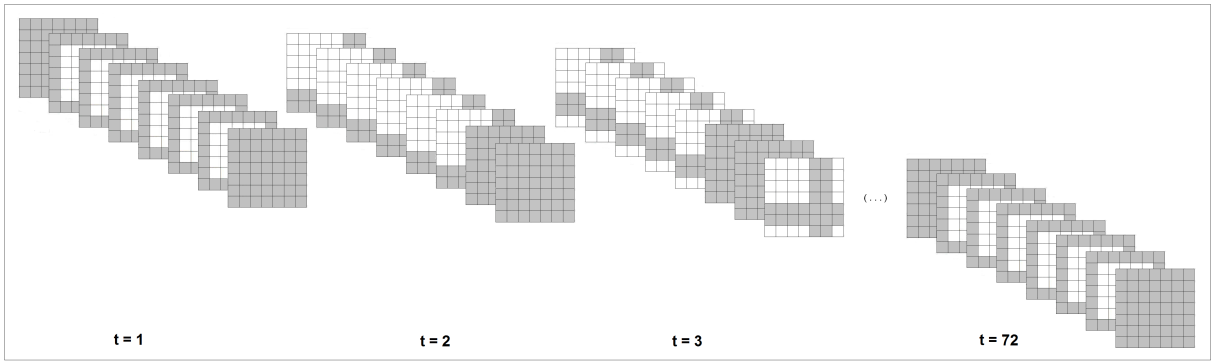


Figura 6.16: Deslocamento espacial das bordas nos três eixos com atraso de um ciclo de *clock*.

nal [Magalhaes, 2010] que no deslocamento da borda no eixo da sensitividade, os dois modelos seriam beneficiados se esse deslocamento fosse de duas posições. Embora esse deslocamento da borda provoque um atraso de 2 ciclos de *clock* entre duas pré-imagem consecutivas, a melhoria obtida na propagação do comportamento caótico compensa esse atraso maior, uma vez que leva a um menor número de pré-imagens necessárias. A Figura 6.17 mostra um exemplo com atraso de duas posições similar ao empregado nos modelos HCA e THCA que foi adaptado para o modelo tridimensional.

Através de alguns testes experimentais que serão apresentados no Capítulo 7, identificamos que o deslocamento ideal dos planos de piso e teto e das colunas no modelo tridimensional são 4 posições. A Figura 6.18 exemplifica a definição das bordas em pré-imagem consecutivas ao cifrar a imagem original da Figura 6.4.

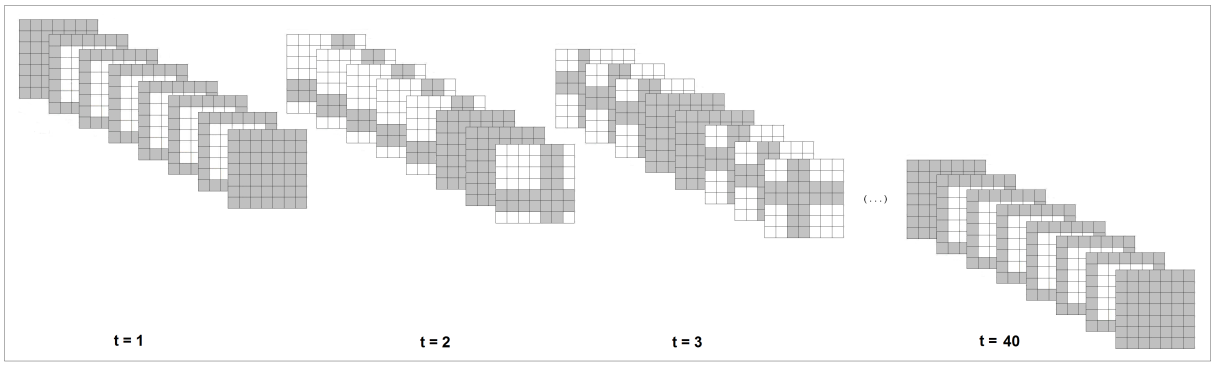


Figura 6.17: Deslocamento espacial das bordas nos três eixos com atraso de dois ciclos de *clock*.

A análise de deslocamento realizada anteriormente refere-se à utilização do modelo com regras de transição de raio 1. Entretanto, para qualquer raio, esses deslocamentos devem ser realizados a cada pré-imagem e o tamanho desses deslocamentos é proporcional a $4r$, tanto na direção da sensitividade (teto e piso) quanto na direção ortogonal (paredes).

Dessa forma, suponha um AC tridimensional qualquer de raio r e um reticulado $p \times m \times n$, sendo p a profundidade do reticulado, m a quantidade de linhas e n a quantidade

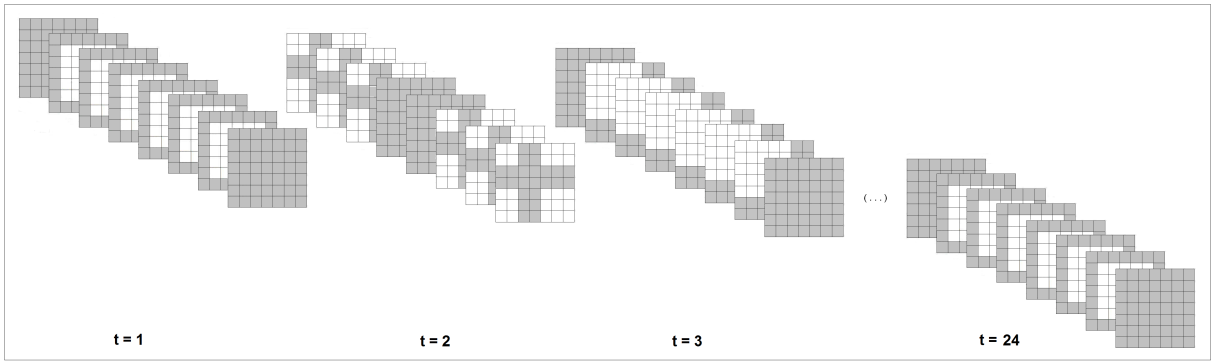


Figura 6.18: Deslocamento espacial das bordas com atraso de quatro ciclos de *clock*.

de colunas. Considerando T o número de pré-imagens e $4r$ o deslocamento entre duas pré-imagens consecutivas, temos que o número de ciclos da de relógio gastos NC_{C3D} é dado pela Equação 6.11.

$$NC_{C3D} = 4T + p - 2r + 1 \tag{6.10}$$

Considerando-se $p = 24$, temos:

$$NC_{C3D} = 4T + 25 - 2r \tag{6.11}$$

Contrastando-se essa equação com a Equação 6.9 elaborada para o modelo de bloco único com borda fixa, vemos o efeito do deslocamento da borda para $r = 1$ e $T = 20$, por exemplo, temos que o número de ciclos de *clock* na cifragem cai de $23T$ para $4T + 23$, ou seja, de 460 para 103. Por outro lado, a decifragem precisaria de apenas 20 ciclos, nos dois modelos.

6.6 Modelo com rotação do núcleo da regra

A última alteração avaliada no modelo tridimensional foi a rotação do núcleo. Essa rotação também é uma alteração que já havia sido empregada nos modelos unidimensional e bidimensional, propostos respectivamente em [Magalhaes, 2010] e [Macedo, 2007]. Ela foi realizada a fim de aumentar a segurança do método, resultando em uma melhoria da qualidade da cifragem.

Em uma primeira análise em [Macedo, 2007] foi identificada que para uma maior segurança do método, seria interessante que diferentes regras principais fossem aplicadas a cada passo de evolução (para frente ou para trás). Para tal, a cada novo cálculo de pré-imagem, cada bit do núcleo da regra principal é rotacionado em uma posição resultando em uma nova regra principal e uma nova regra de contorno. A nova regra de contorno também pode ser alterada a cada passo, selecionando-se dentre as duas possíveis de serem aplicadas com a mesma sensitividade. Para a regra principal, suponha o núcleo {0111}

para regra de raio 1 utilizada no modelo unidimensional. O primeiro passo de cálculo de pré-imagem seria executado considerando a regra principal {01111000} e a regra de contorno {00001111}. Na próxima iteração, a próxima pré-imagem que estiver sendo calculada, utilizará a regra de núcleo {10110100} e será utilizada a regra de contorno {11110000}. Dessa forma, um núcleo inicial será repetido após 5 passos de pré-imagem. No caso do modelo tridimensional, utilizando o núcleo de 64 bits mostrado na 6.2, esse núcleo somente será aplicado novamente ao reticulado após 65 passos de pré-imagem. Dependendo da quantidade de passos utilizados no método, cada regra será aplicada uma única vez durante o processo de cifragem.

6.6.1 Considerações sobre o modelo com intercalação, deslocamento das bordas e rotação do núcleo da regra

O modelo tridimensional com rotação do núcleo apresenta melhores resultados em relação aos modelos anteriores. Essa melhoria já era esperada, pois ao utilizar uma rotação nas bordas, necessariamente, faz com que a regra de contorno não provoque um "congelamento" na região de fronteira do reticulado, provocando a propagação da entropia necessária à cifragem mais rapidamente. Essa melhoria pode ser observada nas duas imagens cifradas observadas na Figura 6.19 (b) e Figura 6.19 (d). Observe que em relação às versões propostas anteriormente o modelo se apresentou bastante eficaz, ou seja, nenhum vestígio de borda foi encontrado na imagem cifrada. Experimentos realizados serão apresentados nos próximos capítulos a fim de provar a eficiência e segurança do modelo tridimensional.

6.7 Modelo final 3DHCA

O modelo final que foi empregado nos testes apresentados nesse trabalho utiliza o modelo tridimensional com bloco único, deslocamento tridimensional das bordas e rotação do núcleo para a geração de uma regra principal diferente a cada passo. As demais especificações do modelo derivam de experimentos realizados e serão apresentadas no próximo capítulo. Os resultados dos experimentos bem como a especificação dos testes serão devidamente apesentados no Capítulo 7, para reforçar a validade do método.

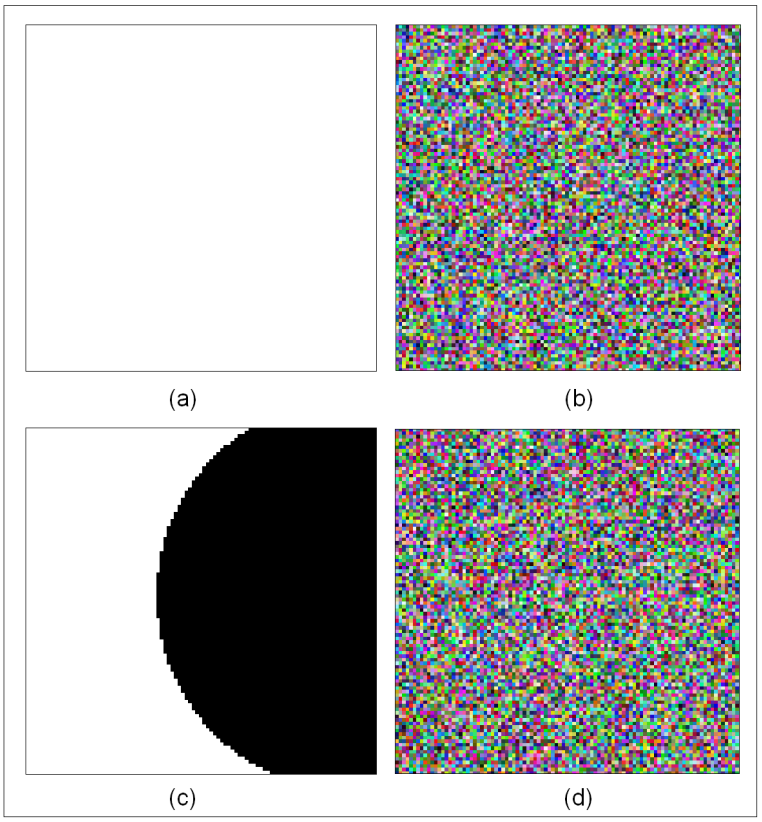


Figura 6.19: Exemplo de imagens cifradas utilizando o modelo de bloco único com deslocamento.(a) Imagem com todos pixels brancos. (b) Imagem "a" cifrada utilizando regra principal e de contorno indicadas na Tabela 6.2. (c) Imagem meio circulo preto e branco. (d) Imagem "c" cifrada utilizando regra principal e de contorno indicadas na Tabela 6.2.

Capítulo 7

Resultados experimentais

Na elaboração do modelo final (com bloco único, deslocamento espacial da borda e rotação do núcleo), experimentos foram realizados com modelos mais simples a fim de validar o modelo tridimensional para a realização da criptografia. Embora o novo método criptográfico proposto nessa dissertação seja altamente paralelizável, o modelo foi implementado utilizando uma abordagem sequencial na linguagem de programação C padrão. A implementação do mesmo em uma arquitetura descentralizada como as placas FPGA [Zambreno et al., 2005] fica como sugestão para investigações futuras.

O primeiro conjunto de experimentos realizado verificou a qualidade de cifragem de cada um dos modelos implementados. Nesses experimentos iniciais, uma cifragem é considerada de boa qualidade quando a imagem cifrada não guarda qualquer padrão similar à imagem original. Esses experimentos foram apresentados no Capítulo 6 e comentados individualmente. A partir das conclusões principais desses experimentos, o modelo final foi elaborado.

As próximas seções consideram testes realizados a partir do modelo com bloco único, deslocamento tridimensional da borda e rotação do núcleo da regra, sendo esta a versão final implementada. Esses experimentos partiram de adaptações para o espaço tridimensional dos experimentos propostos em [Macedo, 2007] e [Magalhaes, 2010] com os modelos HCA e THCA. Antes de apresentar os resultados desses experimentos, apresentaremos alguns conceitos importantes utilizados na avaliação do método 3DHCA, tais como a propagação de uma perturbação e a entropia.

7.1 Considerações sobre os experimentos

7.1.1 Teste de propagação da perturbação

A análise da propagação da perturbação se baseia na imagem da diferença entre uma imagem original cifrada e uma segunda imagem cifrada após a pertubação de um bit na parte central da imagem original. Em seguida calcula-se o XOR entre a imagem original

cifrada e a imagem perturbada cifrada. Esse teste é adequado para verificar se o algoritmo de criptografia proposto é seguro diante do ataque entre um texto plano escolhido, em especial em relação à criptoanálise diferencial. Para ilustrar o processo de análise, observe o exemplo da Figura 7.1, que apresenta uma imagem original A (círculo azul em fundo branco) na Figura 7.1 (a) e uma imagem perturbada A' na Figura 7.1 (b). A imagem A' possui apenas 1 bit alterado em relação à imagem A, na posição indicada na figura. A Figura 7.1 (c) apresenta a imagem resultante entre $A \oplus A'$. Observe na imagem do XOR na Figura 7.1 (c), que todos os pixels estão em preto, exceto o pixel central, que é o pixel que se difere nas duas imagens.

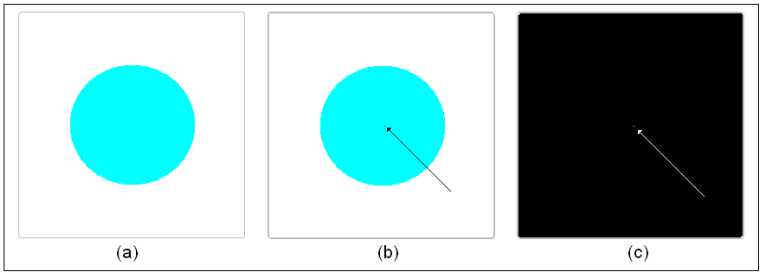


Figura 7.1: (a) Imagem original A. (b) Imagem perturbada A'. (c) Imagem do XOR entre $A \oplus A'$.

Em um método de criptografia é esperado que, embora as Figuras 7.1 (a) e (b) sejam extremamente similares, as cifragens das duas imagens devem ser completamente diferentes, como exemplificado na Figura 7.2. As imagens nas Figuras 7.2 (a) e 7.2 (b) foram obtidos aplicando-se o método 3DHCA por 25 passos (ou seja, foram calculadas 25 pré-imagens) a partir das imagens nas Figuras 7.1 (a) e 7.1 (b) respectivamente. Nesse caso, ao realizar o XOR ($\oplus$) das imagens cifradas espera-se uma imagem do XOR resultante que possui todos os pixels com as cores altamente embaralhadas, como o XOR apresentado na Figura 7.2 (c). Isso mostra que o método criptográfico implementado possui alto índice de aleatoriedade. Para verificar se a imagem do XOR resultante possui a aleatoriedade necessária foram computados o percentual de 0s dessa imagem - que deveria ser próxima a 50% - e a entropia do reticulado tridimensional, que será apresentado na próxima seção. Ou ainda, podemos entender que a imagem perturbada não guarda nenhuma informação da imagem original cifrada. Segundo [Zeghid et al., 2007], isso é denominado propriedade de confusão e difusão, uma propriedade muito importante para qualquer sistema criptográfico. A existência dessa propriedade foi avaliada no método tridimensional aqui proposto.

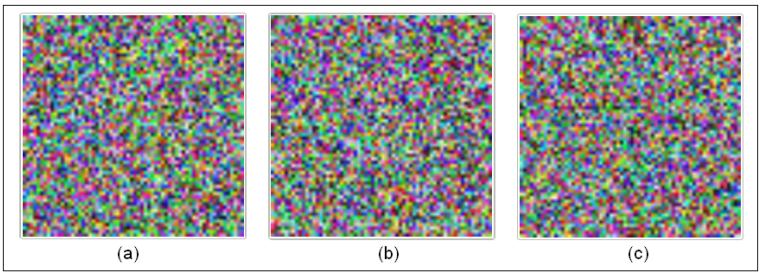


Figura 7.2: (a) Imagem plana A cifrada por $T = 25$. (b) Imagem perturbada A' cifrada por $T = 25$. (c) Imagem do XOR entre $A \oplus A'$ cifradas.

7.1.2 Entropia

A entropia é uma ferramenta básica para a Teoria da Informação, introduzida em [Shannon, 1948]. A entropia pode ser entendida como um medida estatística para tratar a informação, e ela é computada como uma função de distribuição de probabilidade. Ou seja, é uma medida capaz de identificar a aleatoriedade de eventos. A entropia de uma sequência de k eventos é definida pela Equação 7.1, onde p_i é a probabilidade de ocorrência do evento i.

$$S = -\sum_{i=1}^{k} p_i \times \log_2 p_i \tag{7.1}$$

Na investigação do modelo unidimensional [Macedo, 2007], a entropia foi utilizada para medir o nível de aleatoriedade de uma sequência binária linear. Para tal, devemos calcular a ocorrência de N janelas de tamanho j, com $j < N$. O tamanho dessas janelas deve satisfazer à equação de normalização (Equação 7.2), de tal forma que a entropia tenha variação entre 0 e 1 para qualquer tamanho de palavra binária, então o tamanho da janela é dado por: $j = \log_2 N$.

$$s = \frac{-\sum_{i=1}^{k} p_i \times \log_2 p_i}{j} \tag{7.2}$$

O Apêndice E detalha e apresenta exemplos de cálculo de entropia unidimensional utilizada em [Macedo, 2007]. Uma adaptação desse cálculo foi proposta em [Magalhaes, 2010] para arranjos binários bidimensionais para que este fosse compatível com a dimensão utilizada pelo algoritmo de criptografia THCA, conforme detalhado no Apêndice E.

Para nossos experimentos, uma adaptação para o modelo tridimensional foi necessária. Tomando como base o modelo para arranjos unidimensionais e bidimensionais, o modelo para o cálculo da entropia foi reformulado para arranjos binários tridimensionais. Para isso devemos considerar uma configuração 3D da janela. A título de exemplo, considere um reticulado $N = 16 \times 16 \times 16 = 4096$, o tamanho da janela nesse caso deve ser igual a $j = \log_2 4096 = 12$. Dessa forma, existem 3 configurações de janelas 3D possíveis,

uma delas é a configuração: $3 \times 2 \times 2$. Na Figura 7.3, pode-se observar como se dá a construção das janelas tridimensionais passo a passo, onde cada reticulado tridimensional foi apresentado como uma sequência de 16 planos de tamanho 16×16 para facilitar a visualização.

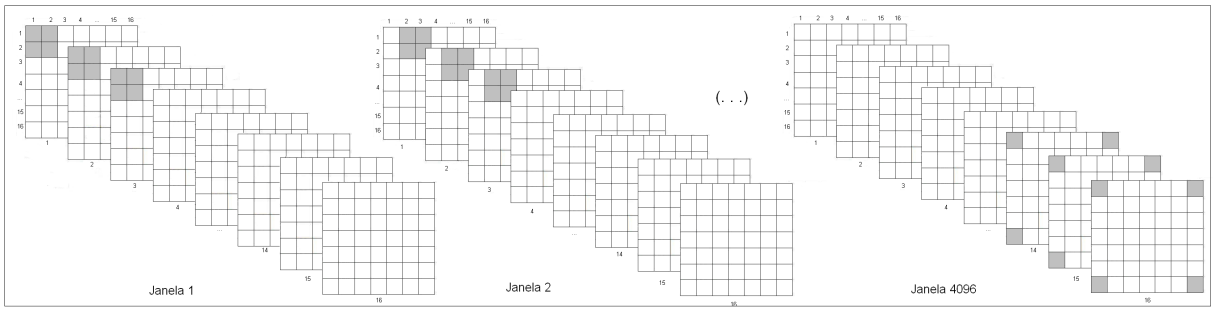


Figura 7.3: Construção das janelas de um reticulado de ordem $16 \times 16 \times 16$.

O cálculo de entropia que utiliza janelas tridimensionais foi utilizado nos experimentos de duas maneiras. Primeiramente, ele foi aplicado para avaliar o nível de aleatoriedade de uma imagem cifrada. Assim, partindo-se de imagens com baixa entropia por serem caracterizados por um padrão (por exemplo, a imagem com um círculo na Figura 7.1 (a), que possui entropia tridimensional para o canal R = 0,06, o canal G = 0,0 e o canal B = 0,0), a medida nos auxiliou a identificar se a cifragem era de boa qualidade. É de se esperar que uma imagem cifrada (por exemplo, a apresentada na Figura 7.2 (a), que possui entropia tridimensional para o canal R = 0,9449, o canal G = 0,9444 e o canal B = 0,9448) tenha uma alta entropia. Assim, o cálculo de entropia nos ajudou a identificar se o método estaca realizando a cifragem adequadamente e até mesmo uma estimativa inicial do número de pré-imagens adequada. Assim, embora os cálculos não tenham sido apresentados no Capítulo 6, eles foram de grande valia nas avaliações realizadas para refinamento do modelo básico até a construção do modelo final 3DHCA. Posteriormente, o cálculo de entropia tridimensional foi utilizado para avaliar o grau de aleatoriedade de um reticulado tridimensional obtido a partir de duas imagens similares cifradas pelo 3DHCA. Dessa forma, o cálculo da entropia nos auxilou a identificar se a imagem obtida pelo XOR das duas imagens cifradas possuía um alto nível de aleatoriedade como desejado, a exemplo do XOR apresentado na Figura 7.2 (c). A entropia utilizando janelas unidimensionais também foi calculada para avaliar o grau de aleatoriedade do núcleo das regras. Tanto para o cálculo da entropia do núcleo das regras como para o cálculo da entropia do reticulado tridimensional, o valor esperado da entropia é o mais próximo de 1.

7.2 Especificação do tamanho do deslocamento da borda

Após os testes iniciais que auxiliaram a especificação do modelo final, o primeiro experimento teve por objetivo verificar o tamanho do deslocamento das bordas que deve

ser aplicado ao modelo tridimensional proposto. Para entender melhor o modelo dividimos o experimento em duas etapas. A primeira etapa verifica o tamanho do deslocamento mais adequado ao modelo criptográfico da borda no mesmo eixo da sensitividade. Uma vez especificado o tamanho do deslocamento da borda no eixo da sensitividade, alterações nos demais eixos foram realizadas para verificar se o modelo sofre grandes variações na propagação da perturbação em relação a esse deslocamento.

7.2.1 Deslocamento da borda no eixo da sensitividade

Neste experimento realizado com o modelo final do 3DHCA, foi analisado o impacto no modelo, referente ao deslocamento da borda discutido na Seção 6.5. Sabe-se que quanto menor o deslocamento, menor o atraso entre o cálculo de duas pré-imagens consecutivas. Foram utilizados 100 cubos binários tridimensionais de dimensões $64 \times 64 \times 64$. Cada um desses cubos tridimensionais binários tiveram seus valores gerados aleatoriamente e cifrados por 30, 50 e 100 passos de tempo. Foram utilizadas 10 regras de raio 1 cujos núcleos (chaves criptográficas) tinham entropias com uma faixa de valores recomendados em [Macedo, 2007] e [Magalhaes, 2010], ou seja, acima de 0,75. Esse conjunto de núcleos (e suas respectivas entropias) está apresentado na Tabela G.1 do Apêndice G. Foram avaliados deslocamentos de 1, 2 e 4 posições no mesmo eixo da sensitividade da regra, correspondendo a um deslocamento de borda de r, $2r$ e $4r$, uma vez que o raio da regra é 1. A análise da propagação da perturbação foi utilizada para verificar os resultados obtidos. Ou seja, foi avaliado qual o deslocamento ideal a se fazer na borda a se obter uma propagação da perturbação adequada. Quanto mais próxima de 0,5 for a relação entre a quantidade de 0s e 1s no XOR realizado entre os reticulados cifrados, melhor foi a propagação da perturbação para o método. Outro valor analisado é o desvio padrão do percentual de zeros que deve ser abaixo de 10%. Segundo [Sen et al., 2002], métodos criptográficos submetidos a esse tipo de teste são considerados bons. Os valores da entropia tridimensional do XOR entre os reticulados cifrados foram analisados para verificar se o modelo de criptografia atende a propriedade de confusão e esses valores devem estar próximos a 1.

A primeira variação aplicada foi o deslocamento de 1 posição. Esse é o deslocamento ideal para o paralelismo, pois ele acarreta um aumento de apenas um ciclo de *clock* entre uma pré-imagem calculada e a próxima a ser cifrada. Para cada reticulado, foi criado um reticulado similar com a alteração de apenas um bit em relação ao reticulado original. Os dois reticulados similares foram cifrados e, ao final, um terceiro reticulado tridimensional foi criado a partir do XOR dos reticulados cifrados. Os resultados são apresentados na Tabela 7.1. A coluna A da Tabela 7.1 indica a quantidade de passos que foram utilizados em cada bloco de experimentos. A coluna B da mesma tabela indica o valor médio da porcentagem de 0s calculada no XOR das imagens cifradas. Esses valores

foram encontrados a partir da cifragem dos 100 pares de reticulados similares A e A' pelas 10 regras de raio 1. A porcentagem de zeros no XOR entre os reticulados cifrados a partir de A e A' foi computada para as 1000 cifragens. Assim, foi realizada uma média da porcentagem de zeros e em seguida foi calculado o desvio padrão, indicado na coluna C da tabela. Quanto mais próximos de 0.5 os valores da coluna B estiverem, pode-se dizer que a diferença entre os reticulados cifrados é significativa. A aleatoriedade dessa diferença é medida pela entropia. A coluna D da tabela, mostra a entropia média, ou seja, para cada um dos 1000 resultados obtidos após a cifragem de 100 cubos e 10 regras de raio 1, uma média da entropia do XOR foi realizada e seu desvio padrão é indicado na coluna E.

Tabela 7.1: Deslocamento de 1 célula da borda no eixo da sensitividade.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
30	0.962665	0.000433	0.094233	0.000825
50	0.962291	0.000273	0.095243	0.000358
100	0.962161	0.000280	0.095692	0.000164

Observe que os resultados encontrados na coluna B da tabela correspondem a um resultado muito distante de 0.5, apesar de ser esperado um percentual de zeros próximo a 50% no XOR realizado entre os dois reticulados cifrados. Esse valor também é confirmado pela entropia próxima de 0 informada na coluna D. Isso significa que o emprego desse deslocamento de borda não é adequado ao método, visto que, a propagação da pertubação foi muito distante da esperada, mesmo considerando uma quantidade de 100 passos de pré-imagens.

O segundo teste realizado utilizou o mesmo tamanho de deslocamento empregado em [Macedo, 2007] e [Magalhaes, 2010], ou seja, foi aplicado um deslocamento na borda de duas posições. A Tabela 7.2 apresenta colunas similares à Tabela 7.1 e aponta que os resultados com um deslocamento de 2 posições é superior ao apresentado na Tabela 7.1. Tanto o percentual de 0s do XOR apresentado na coluna B, quanto a entropia da coluna D, se aproximam dos valores esperados: 0,5 (ou 50%) para o percentual de 0s e valores próximos de 1 para a entropia. Entretanto, tomando como base diversos experimentos realizados, verificamos que para o tamanho da amostra utilizado e para reticulados de tamanho $64 \times 64 \times 64$ o valor médio adequado no percentual de zeros é em torno de 0,5 com uma precisão de 3 casas decimais e a entropia acima de 0.95. Assim, embora os resultados com 80 passos de pré-imagem estejam próximos ao esperado, o valor de 0,507 indica que existem reticulados aonde o XOR dos reticulados similares cifrados ainda guardam um padrão indesejado. Esse resultado foi confirmado visualmente analisando-se a imagem

gerada pelo XOR nos casos onde o percentual de zeros mais se afastou do 0,5. As Figuras 7.4 (a) e (b) apresentam exemplos de padrões observados. Assim, confirmamos que com o deslocamento da borda em 2 células, o número de pré-imagens ideal para reticulados de tamanho $64 \times 64 \times 64$ é 100. Além disso, aplicando-se 100 passos de pré-imagem a análise do XOR entre as imagens cifradas não apresentou padrões, similar à imagem da Figura 7.4 (c). Com esse deslocamento, existe um atraso de 2 ciclos de *clock* entre duas pré-imagens consecutivas.

Tabela 7.2: Deslocamento de 2 células da borda no eixo da sensitividade.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
30	0.783182	0.001427	0.476469	0.002322
50	0.631067	0.000846	0.761200	0.000573
80	0.507584	0.000811	0.942345	0.000345
100	0.500271	0.000972	0.954019	0.000108

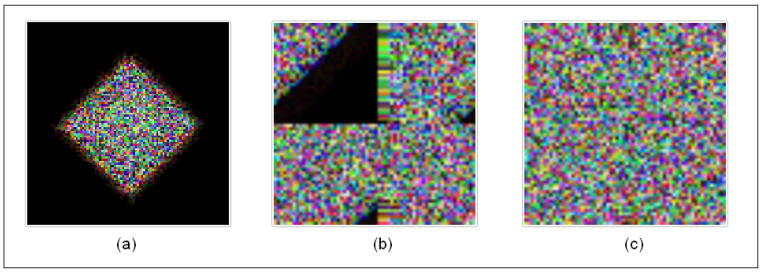


Figura 7.4: (a) Exemplo 1 de reticulado XOR que preserva padrão da imagem original utilizando modelo sem deslocamento de bordas. (b) Exemplo 2 de reticulado XOR que preserva padrão da imagem original utilizando modelo com deslocamento de bordas. (c) Exemplo de reticulado XOR que não apresenta padrão da imagem original utilizando modelo com deslocamento de bordas.

A última variação analisada foi um deslocamento de 4 posições, que provoca um atraso de quatro ciclos de *clock* em relação ao cálculo da próxima pré-imagem. Conforme pode ser observado na Tabela 7.3, a quantidade de passos necessários para a cifragem é bastante inferior à apresentada na Tabela 7.2, pois com 30 passos de pré-imagem já é alcançado um nível de aleatoriedade no XOR similar ao obtido com deslocamento de 2 posições e 100 pré-imagens. Esses resultados também foram confirmados visualmente analisando-se as imagens geradas pelo XOR que resultaram em um percentual mais distante de 0.5. Nenhum padrão na diferença das imagens foi identificado confirmando que 30 pré-imagens são suficientes para a propagação da perturbação. Assim, o número de pré-

imagens necessárias com 4 posições de deslocamento compensa o atraso na quantidade de ciclos de relógio de processamento.

Tabela 7.3: Deslocamento de 4 células da borda no eixo da sensitividade.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
30	0.501455	0.000965	0.953513	0.000131
50	0.499893	0.001010	0.953997	0.000100
100	0.499989	0.001103	0.954024	0.000092

A partir dos resultados desse experimento, no modelo de criptografia tridimensional final e nos experimentos subsequentes, foi aplicado um deslocamento da borda de 4 posições no mesmo eixo da sensitividade.

7.2.2 Deslocamento espacial das bordas

O segundo experimento que foi realizado adequou o modelo em relação ao tamanho do deslocamento empregado nas demais bordas. Cada uma das sub-colunas da coluna A indicam a quantidade de posições que foram deslocadas, sendo que p indica o deslocamento da borda no mesmo eixo da sensitividade e m e n representam o deslocamento nos demais eixos. A primeira linha de dados da tabela foi retirada dos experimentos anteriores, onde o deslocamento foi de 4 posições em todos os eixos do reticulado. Nas demais linhas um deslocamento de 2 posições para m e n e 1 posição para m e n foram aplicados nos demais eixos do reticulado. Esses experimentos verificaram que o tamanho do deslocamento aplicado nas demais bordas não interferem diretamente no resultado da propagação da perturbação conforme pode ser observado pelas colunas C e E da Tabela 7.4. Entretanto, se não for aplicado nenhum deslocamento nos demais eixos, a borda é congelada. Ou seja, tanto o reticulado inicial A quanto o A' possuem a mesma borda. Nesse caso, apenas os bits centrais serão diferentes nos dois reticulados e a borda funciona como uma barreira para a propagação, visto que é sempre mantida igual para os dois reticulados (lembrando que a perturbação foi no bit central do reticulado). Dessa forma, a propagação sempre carrega em torno de 2% de informação idêntica para os dois reticulados e esse valor faz com que a média seja em torno de 0,52. Esse valor médio não é adequado para o processo de cifragem, visto que o valor esperado é de 50% com 3 casas de precisão.

Assim, o deslocamento nas demais bordas adotado foi mantido idêntico ao deslocamento da borda no eixo da sensitividade, por uma questão de simplicidade do modelo, uma vez que o deslocamento nos dois eixos não acarreta atrasos no cálculo de duas pré-imagens

consecutivas. Portanto, no modelo final foi adotado um deslocamento de 4 posições nas três dimensões para regras de raio 1. De forma genérica, o deslocamento adotado é $4r$ posições nas três dimensões.

Tabela 7.4: Verificação da interferência do deslocamento das demais bordas do reticulado.

Bloco $64 \times 64 \times 64$ com diferentes rotações na borda.							
A			B	C	D	E	F
p	m	n	Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
4	4	4	30	0.501455	0.000965	0.953513	0.000131
4	2	2	30	0.501512	0.000997	0.953512	0.000116
4	1	1	30	0.501358	0.001012	0.953573	0.000100
4	0	0	30	0.531458	0.000862	0.931479	0.000169

7.3 Experimentos para definição da quantidade de passos de de pré-imagem na cifragem

Os experimentos relatados na Seção 7.2 utilizaram apenas reticulados com tamanho $64 \times 64 \times 64$. Embora, a análise anterior tenha sido suficiente para decidir o deslocamento das bordas nas 3 dimensões, sabemos que o número de passos necessários para uma cifragem de boa qualidade varia em função do tamanho do reticulado a ser cifrado. Nessa seção, descrevemos experimentos onde o número de passos de pré-imagem necessários a uma boa propagação da perturbação na cifragem foi avaliado, em relação ao tamanho do reticulado a ser cifrado.

7.3.1 Experimentos com reticulados cúbicos

Nesses experimentos, verificou-se a quantidade de passos necessários para propagação da pertubação de 1 bit do reticulado original, através da análise do reticulado do XOR obtido entre os dois reticulados cifrados, um a partir do reticulado original e outro perturbado. Essa etapa foi realizada a fim de constatar a proporcionalidade entre o tamanho do bloco cifrado e a quantidade de passos necessários para que a perturbação se propague ao longo de todo o reticulado.

Nesses experimentos foram utilizados 100 cubos binários tridimensionais gerados aleatoriamente, para a verificação e adequação do método em relação ao tamanho do bloco e a quantidade de pré-imagens que deve ser utilizada. Cada um desses cubos binários foram cifrados por 10 regras. Essas regras são as mesmas utilizadas nos experimentos da

Seção 7.2.1 e apresentadas no Apêndice G. Os resultados obtidos foram utilizados para se ter uma estimativa da quantidade de passos para a cifragem necessárias em relação a cada tamanho de blocos utilizados.

O primeiro tamanho de reticulado testado foi $16 \times 16 \times 16$. A avaliação de uma determinada quantidade de passos de pré-imagem fornece a difusão adequada de uma perturbação foi feita de acordo com a coluna B (porcentagem média de zeros) da Tabela 7.5 e pela entropia tridimensional, indicada na coluna D. Para o tamanho $16 \times 16 \times 16$, foi possível observar que uma quantidade de 10 pré-imagens já é suficiente para obter um reticulado cifrado, analisando-se o reticulado do XOR tanto em termos do percentual de zeros observados quanto da entropia observada.

Tabela 7.5: Quantidade de pré-imagens necessárias para propagação da perturbação em reticulados $16 \times 16 \times 16$.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
5	0.723484	0.020369	0.672308	0.033468
8	0.521584	0.011142	0.926247	0.004577
10	0.500801	0.007960	0.930980	0.001187
15	0.500181	0.007528	0.930933	0.001178

O segundo tamanho de cubo utilizado foi $32 \times 32 \times 32$. De acordo com coluna B da Tabela 7.6, temos que a quantidade suficiente de passos de pré-imagem nos experimentos é igual a 20. Essa quantidade de passos de pré-imagem foi definida de acordo com a porcentagem de 0s obtidos da operação do XOR entre os dois reticulados cifrados a partir do reticulado original e do reticulado perturbado. A coluna B da Tabela 7.6 indica que os resultados com 20 passos estão próximos a 50% e são suficientes para garantir blocos de dados cifrados seguros. Os valores médios encontrados para entropia aplicando-se 20 passos de pré-imagem também estão próximos do valor 1 e são considerados adequados.

Tabela 7.6: Quantidade de pré-imagens necessárias para propagação da perturbação em reticulados $32 \times 32 \times 32$.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
15	0.504492	0.002825	0.944368	0.000567
20	0.499641	0.002684	0.944887	0.000360
25	0.499668	0.002801	0.944835	0.000379
30	0.500305	0.002905	0.944853	0.000369
50	0.500359	0.002600	0.944843	0.000389
100	0.500133	0.002660	0.944871	0.000347

O terceiro tamanho de cubo que foi utilizado possui dimensões à $64 \times 64 \times 64$. É possível observar na Tabela 7.7 que com 20 e 25 passos pré-imagem as porcentagens médias estão altas se considerado o valor esperado de 0.5 com precisão de 3 casas decimais. Esse fato é corroborado pelos desvios padrões que se mostram mais altos de acordo com a coluna C da Tabela 7.7. Observamos que a quantidade de passos para trás que devem ser utilizados, a fim de que a propagação adequada da perturbação seja atingida, é igual a 30. De acordo com a coluna B da Tabela 7.7, observamos que a porcentagem de 0s a partir de 30 passos de pré-imagem está em torno de 50% e a entropia (coluna D) é considerada adequada (próxima a 1). Portanto, esses resultados reafirmam que a quantidade de 30 pré-imagens é suficiente para a difusão da perturbação, como já identificado nos experimentos escritos na Seção 7.2.1 que utilizam reticulados $64 \times 64 \times 64$.

Tabela 7.7: Quantidade de pré-imagens necessárias para propagação da perturbação em reticulados $64 \times 64 \times 64$.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
20	0.585207	0.002928	0.840245	0.004174
25	0.520594	0.001440	0.933294	0.000989
30	0.501455	0.000965	0.953513	0.000131
50	0.499893	0.001010	0.953997	0.000100
100	0.499989	0.001103	0.954024	0.000092

O próximo tamanho de reticulado analisado foi $128 \times 128 \times 128$. De acordo com a coluna

B da Tabela 7.8 podemos observar que as quantidades de 30 e 50 passos de pré-imagem não são suficientes para propagar a perturbação ao longo de todo o reticulado. A quantidade de passos mínimo em nossos experimentos para que se atingisse uma porcentagem de 0s próximo do valor de 0.5 foi de 60 passos de pré-imagem, observado na coluna B Tabela 7.8.

Tabela 7.8: Quantidade de pré-imagens necessárias para propagação da perturbação em reticulados $128 \times 128 \times 128$.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
30	0.669880	0.002975	0.685805	0.005285
50	0.515644	0.000440	0.936350	0.000376
60	0.500677	0.000320	0.952196	0.000032
75	0.500007	0.000335	0.952394	0.000037
100	0.500006	0.000344	0.952399	0.000026

O próximo tamanho analisado foi $256 \times 256 \times 256$. De acordo com a coluna B da Tabela 7.9 a quantidade de 120 passos é suficiente para propagar a perturbação, a partir análise da operação XOR entre os reticulados original e perturbado. Até o valor de 100 pré-imagens calculadas, os valores médios para a porcentagem de 0s estão acima de 50% o que não torna essa quantidade de pré-imagens adequada para uma cifragem de boa qualidade.

Tabela 7.9: Quantidade de pré-imagens necessárias para propagação da perturbação em reticulados $256 \times 256 \times 256$.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
30	0.838091	0.004824	0.348568	0.010927
50	0.718369	0.002813	0.616145	0.005765
100	0.513616	0.000224	0.958412	0.000231
120	0.500444	0.000119	0.966904	0.000027

O último tamanho de bloco testado foi $512 \times 512 \times 512$. A Tabela 7.10 apresenta os resultados de testes obtidos variando-se a quantidade de pré-imagens na cifragem. É

possível observar na Tabela que apenas com 250 passos de pré-imagem foi possível obter um percentual de 0s em torno de 50%.

Tabela 7.10: Quantidade de pré-imagens necessárias para propagação da perturbação em reticulados $512 \times 512 \times 512$.

A	B	C	D	E
Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
30	0.914325	0.004739	0.157249	0.002149
50	0.861161	0.004415	0.192482	0.003542
100	0.713818	0.002519	0.321770	0.001490
250	0.500026	0.000042	0.973370	0.003022

Analisando-se todo o conjunto de testes desde os reticulados de $16 \times 16 \times 16$ até $512 \times 512 \times 512$, pode-se observar que a quantidade de pré-imagens adequada a cada um dos tamanhos de bloco são proporcionais aproximadamente à metade do tamanho de uma das dimensões desse bloco. Ou seja, à medida que a quantidade de bits do bloco é aumentado em 8 vezes em relação ao bloco anterior, a quantidade de pré-imagens necessária para que uma boa propagação, aumenta em 2 vezes.

7.3.2 Experimentos com reticulados irregulares

Nas seções anteriores, foram apresentados experimentos realizados na cifragem de cubos, ou seja, todas as dimensões do reticulado utilizados tinham o mesmo tamanho. Entretanto, uma imagem real normalmente não gera um reticulado tridimensional de tamanhos proporcionais. Isso, se deve ao fato de o reticulado tridimensional gerado por uma imagem de $m \times n$ pixels no padrão RGB, resulta em um reticulado tridimensional de dimensões $24 \times m \times n$ bits. Na análise apresentada nessa seção foi verificada a interferência do eixo escolhido para a sensitividade da regra em relação ao número de pré-imagens que são gastas para cifrar completamente o bloco de dados. Para isso, nenhum vestígio deve ser encontrado no reticulado gerado a partir do XOR entre a imagem original cifrada e a imagem perturbada cifrada. A metodologia para a apuração dos resultados será a mesma adotada nos experimentos anteriores.

Para esse experimento foram utilizadas 100 imagens coloridas no padrão RGB geradas aleatoriamente e as mesmas 10 regras utilizadas nos experimentos anteriores foram utilizadas para cifrar cada um desses reticulados. As dimensões das imagens utilizadas são proporcionais à 128×64 pixels. Essas dimensões resultam em reticulados tridimensionais de $24 \times 128 \times 64$ bits, que corresponde a 196608 células. Foi calculada a entropia do reti-

culado resultante da operação do XOR entre as duas imagens cifradas a partir da imagem original e da imagem perturbada. A entropia foi calculada, separadamente, para cada um dos reticulados que representa canais de cores R, G e B, [Habibipour et al., 2010]. Os resultados desses experimentos são detalhados na Tabela 7.12 e a descrição de cada uma de suas colunas é apresentada na Tabela 7.11.

Nesta análise será verificado qual o grau de interferência da variação do eixo da sensitividade em relação à quantidade de pré-imagens aplicada e consequentemente à qualidade da imagem cifrada. Este experimento também é importante, pois a geração dos reticulados tridimensionais a partir de imagens coloridas no padrão RGB é feita de forma não proporcional. Ou seja, o reticulado gerado tem dimensões proporcionais à $m \times n \times 24$. Dependo do eixo da sensitividade escolhido, essa irregularidade no reticulado pode interferir na quantidade de pré-imagens necessárias para obter uma cifragem de qualidade. O grau dessa interferência já sido investigado em [Magalhaes, 2010] para reticulados bidimensionais.

Tabela 7.11: Especificação detalhada das colunas.

Label	Descrição
A	Dimensão do eixo sensitividade da regra da sensitividade.
B	Quantidade de passos de pré-imagem.
C	Porcentagem média de 0's observada no XOR.
D	Desvio padrão da porcentagem de 0's.
E	Entropia média do canal R.
F	Desvio padrão do canal R.
G	Entropia média do canal G.
H	Desvio padrão do canal G.
I	Entropia média do canal B.
J	Desvio padrão do canal B.

Tabela 7.12: Experimento que avalia qual é a melhor dimensão de cifragem para um reticulado com dimensões irregulares.

Bloco $128 \times 64 \times 24$ com rotação espacial da borda em 4 posições									
A	B	C	D	E	F	G	H	I	J
128	30	0.518153	0.001247	0.935414	0.000926	0.935507	0.000856	0.935551	0.000865
128	35	0.500760	0.001083	0.948348	0.000208	0.948312	0.000259	0.948318	0.000201
128	40	0.500146	0.001208	0.948343	0.000278	0.948250	0.000169	0.948268	0.000207
128	50	0.499985	0.001027	0.948308	0.000206	0.948309	0.000194	0.948266	0.000231
64	30	0.526578	0.005154	0.923225	0.006954	0.923210	0.006984	0.923312	0.006956
64	35	0.502334	0.001680	0.947826	0.000678	0.947754	0.000614	0.947833	0.000678
64	40	0.500390	0.001147	0.948340	0.000171	0.948337	0.000220	0.948372	0.000266
64	50	0.500036	0.001180	0.948290	0.000209	0.948292	0.000215	0.948348	0.000215
24	30	0.524766	0.005637	0.935922	0.004673	0.933242	0.005311	0.938285	0.004048
24	35	0.504201	0.001278	0.948325	0.000233	0.948298	0.000235	0.948254	0.000266
24	40	0.503729	0.001045	0.948311	0.000194	0.948327	0.000251	0.948325	0.000215
24	50	0.499953	0.001172	0.948306	0.000208	0.948309	0.000221	0.948332	0.000223

Na Tabela 7.12, a coluna A indica qual o tamanho do eixo escolhido para a sensitividade da regra. A coluna B indica o número de passos de pré-imagem a coluna C indica o percentual de 0s obtidos a operação XOR entre os dois reticulados cifrados a partir do reticulado original e do reticulado perturbado. Assim como nos experimentos das seções anteriores, o valor esperado para a coluna C também é de 0.5. Assim, observamos que a partir de 35 passos de pré-imagem já é possível que se tenha uma boa cifragem quando a sensitividade da regra é escolhida no eixo da dimensão 128. Entretanto, quando a sensitividade está no eixo de dimensão 64 apenas 35 passos não são suficientes. De acordo com a coluna C da Tabela 7.12 com 40 passos já é possível observar que a cifragem possui boa qualidade. Para o eixo de menor dimensão (24), de acordo com a coluna C, 50 passos já são suficientes para cifrar o reticulado.

As entropias médias de cada um dos canais podem ser observadas respectivamente nas colunas E, G e I da Tabela 7.12, assim como os desvios padrões de cada um desses canais. Pode-se notar que a entropia resultante é proporcional à porcentagem de zeros, ou seja, quanto mais próximo essa porcentagem ficou de 0.5, maior é a aleatoriedade da imagem da propagação da perturbação, resultando dessa forma em valores mais próximos de 1 tendendo a 0,948 para esse tamanho de imagem.

A quantidade de 50 passos de pré-imagem cifra de forma adequada para a sensitividade aplicada em qualquer um dos três eixos. Contudo, com apenas 35 passos de pré-imagem

já é possível obter uma cifragem de boa qualidade se escolhida a maior dimensão do reticulado (128).

A partir das conclusões obtidas no experimento, é necessário definir uma nova forma de calcular o número de ciclos de *clock* em uma arquitetura totalmente paralela. Na Equação 7.3 apresentada anteriormente, é considerado que a sensitividade da regra é sempre no eixo que tem dimensão 24 e os planos da imagem podem ser cifrados de forma síncrona independente do tamanho da imagem $m \times n$. Uma vez que identificamos pelos experimentos descritos na Tabela 7.12 que a escolha do eixo de 24 células nem sempre é o mais vantajoso, iremos definir uma equação que independe das dimensões da imagem, mas que considera que o eixo escolhido é sempre o de maior tamanho. Sendo T a quantidade de passos de pré-imagem utilizados para propagar a perturbação, r o raio que está sendo adotado como tamanho de chave, p a profundidade do reticulado, m a largura da imagem e n a altura da imagem, temos que a Equação 7.3 descreve a quantidade de ciclos necessários para que a imagem seja totalmente encriptada.

$$NC_{C3D} = 4T + \max(p, m, n) - 2r + 1 \tag{7.3}$$

7.3.3 Perturbação de 1 bit do núcleo da regra

Os experimentos apresentados nas seções anteriores verificaram o comportamento do modelo em relação a perturbações provocadas nos reticulados originais. Estes testes verificaram a propriedade de difusão do método. Nessa subseção serão apresentados os testes que verificam o comportamento dos modelos quando a perturbação do bit é provocada na chave. O teste consiste em cifrar 100 reticulados gerados de maneira aleatória com 10 chaves de raio 1 e em seguida cifrar o mesmo reticulado com as 10 chaves de raio 1, alteradas em único bit, sendo que este está posicionado na célula central da chave. A metodologia para apuração dos resultados foi semelhante à aplicada nos testes anteriores.

Neste caso, a entropia tridimensional dos reticulados obtidos no XOR também foram calculados a fim de investigar se o modelo atende o princípio da confusão, ou seja, a cifragem de um mesmo reticulado a partir de chaves similares deve gerar reticulados cifrados completamente diferentes.

A quantidade de passos utilizados para a realização desses experimentos é mostrado na coluna B da Tabela 7.13 e foram definidos a partir dos experimentos da Subseção 7.3.1. Os reticulados utilizados são cubos binários em três tamanhos diferentes de largura: 32, 64 e 128. Os valores utilizados como referência para essa análise, também utilizam a porcentagem média de 0s esperada de 0.5. Os resultados obtidos após essa análise podem ser observados na Tabela 7.13.

Tabela 7.13: Propagação da perturbação de um bit do núcleo da regra/chave criptográfica.

Blocos de Cúbicos Tridimensionais de tamanhos variados					
A	B	C	D	E	F
Tamanho do Reticulado	Quantidade de passos (T)	Porcentagem média de 0's	Desvio padrão (B)	Entropia media	Desvio padrão (D)
$32 \times 32 \times 32$	20	0.499861	0.002898	0.944862	0.000379
$64 \times 64 \times 64$	30	0.500011	0.000932	0.954019	0.000082
$128 \times 128 \times 128$	60	0.499956	0.000134	0.948705	0.000034

Os resultados mostraram que o modelo também é seguro em relação à perturbação de 1 bit da chave, já que o modelo apresentou resultados médios próximos à 0.5 para todos os tamanhos de reticulado apresentados e para as mesmas quantidades de passos de pré-imagens definidos nas seções anteriores. Esses valores também podem ser analisados de acordo com o desvio padrão do percentual de zeros abaixo de 10%, o que significa que a imagem cifrada resultante distribuiu uniformemente os percentuais de zeros no reticulado da diferença. O grau de aleatoriedade do modelo foi bastante alto, visto que, as entropias dos reticulados estão altas, ou seja, com valores próximos a 1.

7.4 Experimentos com imagens reais

Até o momento, apresentamos resultados envolvendo reticulados gerados de maneira aleatória, que não representam imagens reais. A partir dos testes realizados nessa seção, utilizaremos imagens reais no formato "Portable Pixel Map" (*.ppm*) que é um formato simples para imagens a cores. O banco de imagens utilizadas nesses experimentos foram adaptadas manualmente a partir de imagens encontradas em `http://www.cis.temple.edu/~latecki/TestData/mpeg7shapeB.tar.gz` ou criadas diretamente no editor *Gimp* 2.9 e são apresentadas no Apêndice F, além da imagem clássica "Baboon" apresentada na Figura 7.5.

7.4.1 Análise de histogramas

Nessa seção, apresentaremos experimentos com o objetivo de verificar como o processo de cifragem altera a distribuição de cores da imagem original. Em um histograma de uma imagem cifrada, é desejável que as cores disponíveis em cada canal estejam distribuídas uniformemente pelas 256 cores disponíveis em cada canal R, G e B. Por isso, se faz necessário a criação de três histogramas, cada um representando a distribuição de cores em cada canal. As imagens utilizadas nesse experimento utilizam 64×64 pixels.

Na Figura 7.5 (a) temos uma imagem de 64×64 ("Baboon") no padrão RGB e as Figuras 7.5 (b), (c) e (d) representam respectivamente o histograma das cores nos canais R, G e B. A Figura 7.6 (a) apresenta a imagem cifrada a partir da Figura 7.5 (a) e as Figuras 7.6 (b), (c) e (d) representam os histogramas relativos à imagem cifrada. Para a cifragem foram utilizados 30 passos de pré-imagens e uma regra com entropia do núcleo acima de 0,75 que é a regra gerada pelo núcleo de índice 0 na Tabela G.1.

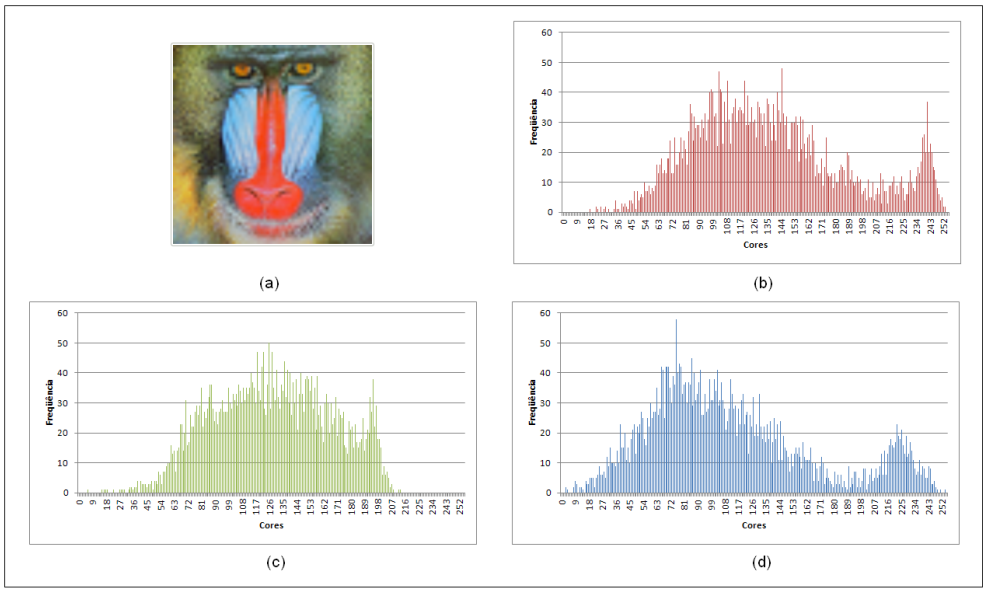


Figura 7.5: (a) Imagem original de 64×64 pixels no padrão RGB. (b) Histograma do canal R. (c) Histograma do canal G. (d) Histograma do canal B.

A imagem original possui tamanho de 64×64 pixels e usa padrão RGB, sendo que a quantidade total de pixels é 4096. É possível observar que a frequência de cores dos histogramas da imagem cifrada está distribuída mais uniformemente ao longo das 256 cores que compõem a paleta de cada canal, quando comparados aos histogramas da imagem cifrada na Figura 7.5. Dessa forma, é nítido que o histograma da imagem cifrada não conserva qualquer informação a respeito da distribuição original das cores da imagem que foi submetida à cifragem.

Para evidenciarmos ainda mais essa característica, uma análise de uma imagem de cor única foi cifrada utilizando as mesmas condições para a realização da cifragem anterior. Como a imagem é toda branca, os canais R, G e B apresentam todos os 4096 pixels com a mesma cor. Os valores dos canais da imagem se concentram na maior intensidade (255). A Figura 7.7 apresenta a imagem branca original e os respectivos histogramas. Na Figura 7.8, a imagem cifrada gerada a partir da Figura 7.7 e também seus respectivos histogramas são apresentados. Nessa imagem é ainda mais nítida a mudança na distribuição de cores nos histogramas dos três canais, que se aproximam de uma distribuição uniforme na imagem cifrada.

A imagem de fundo preto com um círculo amarelo ao centro, apresentada na Figura 7.9 (a) também foi cifrada utilizando as mesmas configurações das duas cifragens anteriores e

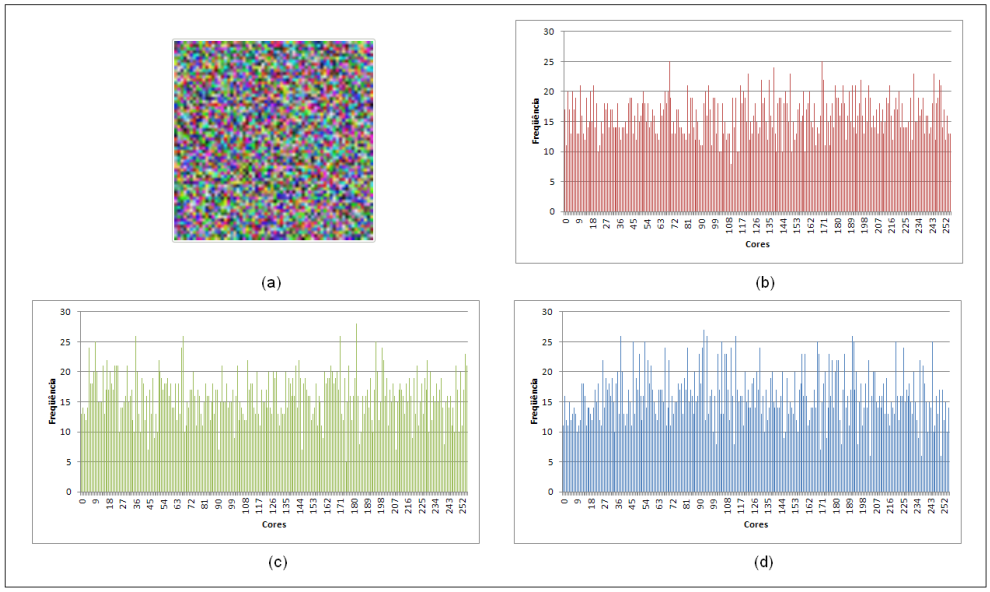


Figura 7.6: (a) Imagem cifrada a partir da imagem da Figura 7.5 padrão RGB. (b) Histograma do canal R. (c) Histograma do canal G. (d) Histograma do canal B.

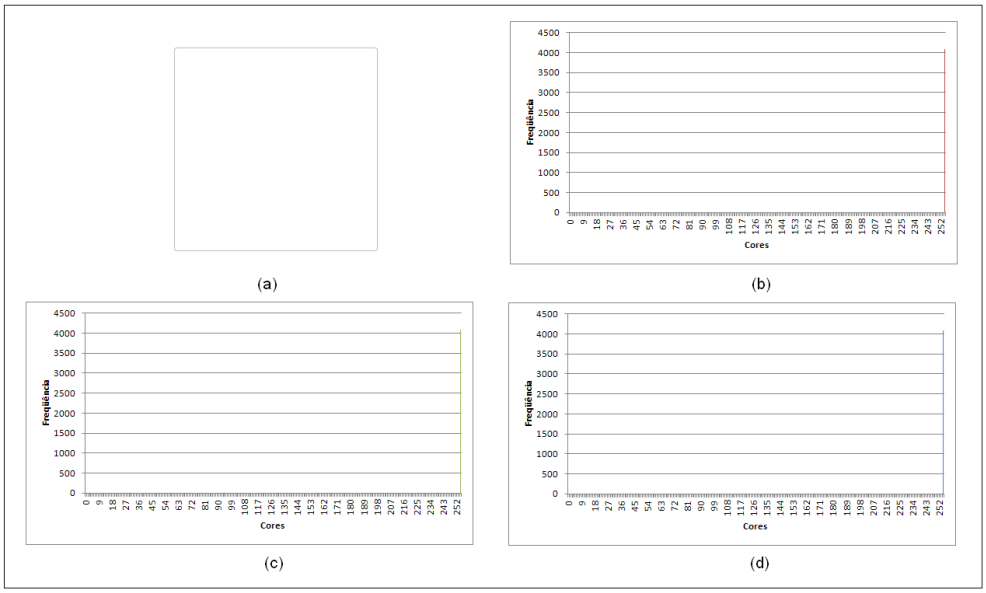


Figura 7.7: (a) Imagem original de 64×64 pixels no padrão RGB. (b) Histograma do canal R. (c) Histograma do canal G. (d) Histograma do canal B.

resultou na imagem cifrada mostrada na Figura 7.10 (a). Assim, é possível perceber que os histogramas na Figura 7.10 (b), (c) e (d) que representam cada um dos canais de cores são bem diferentes dos histogramas na Figura 7.9 (b), (c) e (d,), que possuem seus valores concentrados em no máximo duas cores em cada canal. Assim, é possível concluir que o método não preserva características referentes às cores das imagens, pois tanto na imagem com várias cores quanto nas imagens de uma única cor e com pouca variação de cores, os histogramas das imagens cifradas foram semelhantes entre si e totalmente diferentes dos histogramas das imagens originais.

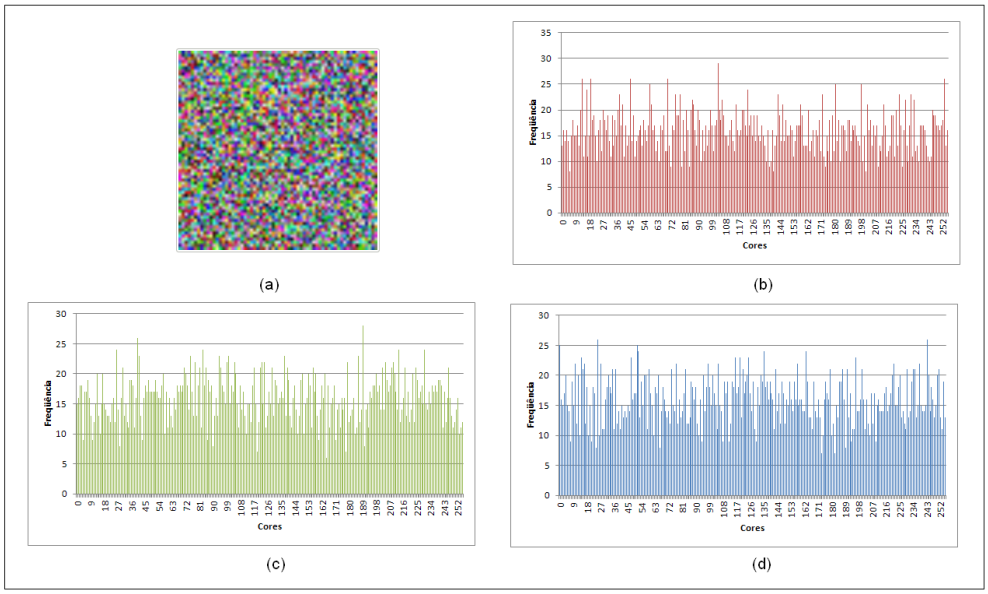


Figura 7.8: (a) Imagem cifrada a partir da imagem da Figura 7.7 padrão RGB. (b) Histograma do canal R. (c) Histograma do canal G. (d) Histograma do canal B.

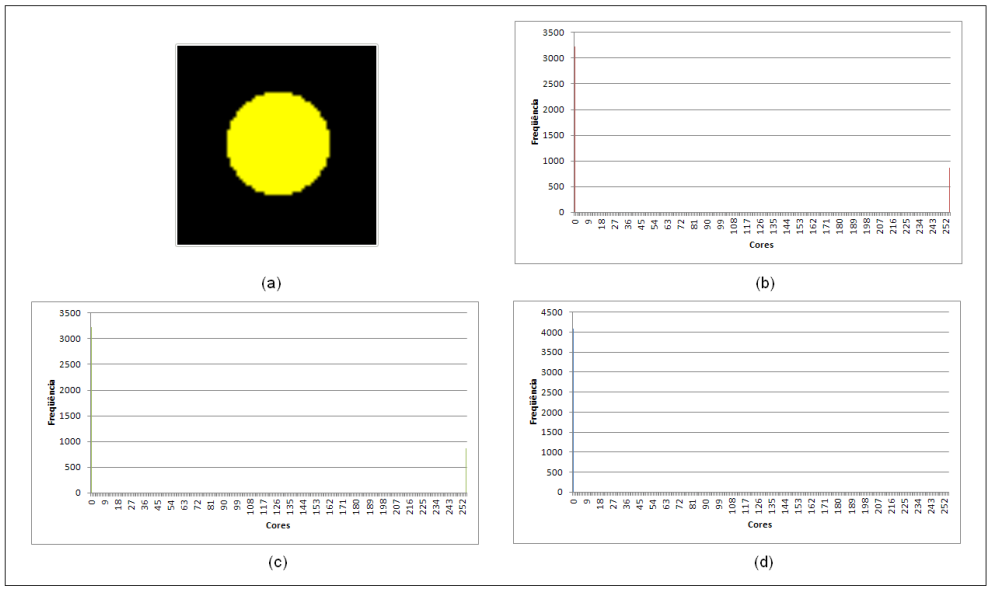


Figura 7.9: (a) Imagem original de 64×64 pixels no padrão RGB. (b) Histograma do canal R. (c) Histograma do canal G. (d) Histograma do canal B.

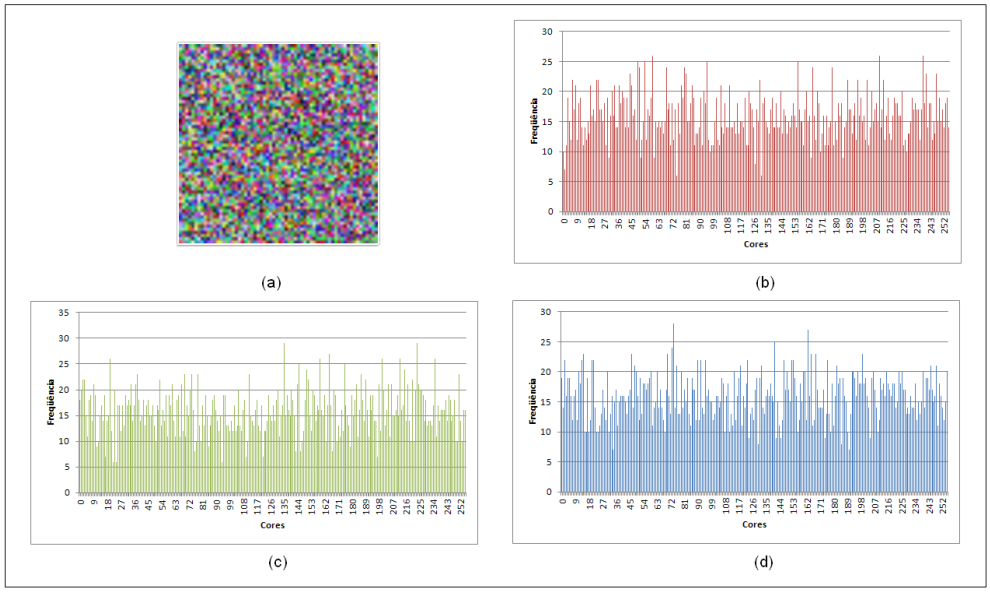


Figura 7.10: (a) Imagem cifrada a partir da imagem da Figura 7.9 no padrão RGB. (b) Histograma do canal R. (c) Histograma do canal G. (d) Histograma do canal B.

7.4.2 Experimentos com um conjunto de regras de alta cardinalidade

Novos experimentos foram realizados com um banco real de 200 imagens apresentadas no Apêndice F. Todas as imagens utilizadas nesse experimento possuem dimensões 64×64 pixels. Cada imagem foi cifrada por 200 regras geradas aleatoriamente e por 20 passos de pré-imagem. O objetivo desse primeiro experimento foi reduzir o conjunto de imagens para realizar um teste mais robusto na cifragem, com um número elevado de regras/chaves criptográficas. Assim, adotamos uma metodologia similar a [Magalhaes, 2010] onde um conjunto reduzido de imagens foi escolhido, identificando-se as 10 imagens mais difíceis de cifrar. Ou seja, buscamos identificar dentro do conjunto inicial de 200 imagens reais, quais delas precisam, em média, de um número maior de passos de cifragem.

Devido ao fato do experimento gerar 40000 linhas de dados, foi inviável exibir todos os resultados nessa dissertação. Assim, a média das métricas computadas na cifragem considerando-se as 200 regras está apresentada na Tabela G.3. As especificações de cada coluna dessa tabela são detalhadas na Tabela G.2. A partir da ordenação decrescente da coluna B da Tabela G.3, a metodologia utilizada para selecionar as imagens mais difíceis de cifrar foi buscar aquelas que retornavam uma porcentagem média de zeros do XOR ($\oplus$) mais alta, ou seja, mais distante do valor esperado de 0,5. Praticamente, todas as imagens apresentaram resultados médios distantes de 0,5, uma vez que só foram utilizados 20 passos de pré-imagem. Pelos testes da Tabela 7.7 já esperávamos que seriam necessários entre 25 e 30 passos para uma cifragem adequadas à imagens 64×64. Entretanto, empregamos um número menor de pré-imagens para identificarmos as imagens mais difíceis

de cifrar. Foi possível perceber que algumas imagens possuem desempenho mais lento a partir do conjunto de regras que foi utilizado e queremos identificar justamente as imagens que precisam de mais passos para serem cifradas. A Figura 7.11 apresenta as imagens selecionadas consideradas os maiores desafios na cifragem. Como resultado a Tabela 7.14 apresenta a média geral para as 200 imagens e a Tabela 7.15 apresenta as médias das 10 imagens selecionadas.

Tabela 7.14: Resultado geral dos 40000 experimentos.

A	B	C	D	E	F	G	H	I
Quantidade de passos (T)	Média % de zeros	Desvio padrão da $\overline{X}$	Entropia canal R	Desvio padrão R	Entropia canal G	Desvio padrão G	Entropia canal B	Desvio padrão B
Média Geral	0.530449	0.002648	0.922837	0.003000	0.922735	0.003044	0.922666	0.003016

Tabela 7.15: Resultado geral dos 40000 experimentos.

A	B	C	D	E	F	G	H
Imagem	Média % de zeros	Desvio simples	Desvio média geral	Desvio média local	Entropia canal R	Entropia canal G	Entropia canal B
53.ppm	0.530981	0.030981	0.000038	0.002294	0.922391	0.922325	0.922186
122.ppm	0.530789	0.030789	0.000024	0.002688	0.922385	0.922442	0.922241
137.ppm	0.530757	0.030757	0.000022	0.002603	0.922644	0.922437	0.922476
147.ppm	0.530757	0.030757	0.000022	0.002870	0.922546	0.922418	0.922409
152.ppm	0.530790	0.030790	0.000024	0.002623	0.922553	0.922471	0.922396
168.ppm	0.530816	0.030816	0.000026	0.002545	0.922327	0.922352	0.922404
180.ppm	0.530967	0.030967	0.000037	0.002716	0.922304	0.922207	0.922285
189.ppm	0.530757	0.030757	0.000022	0.002768	0.922414	0.922361	0.922299
191.ppm	0.530874	0.030874	0.000030	0.002736	0.922554	0.922452	0.922427
198.ppm	0.530934	0.030934	0.000034	0.002561	0.922353	0.922255	0.922287

Uma vez escolhidas as 10 imagens mais difíceis de cifrar, um experimento mais exaustivo foi realizado buscando avaliar um número maior de chaves criptográficas. O objetivo desse experimento é avaliar de forma mais aprofundada se a quantidade de pré-imagens adotada é suficiente para uma quantidade expressiva de regras realizar uma cifragem de qualidade. E principalmente verificar se no meio de um conjunto grande de regras existem chaves que não executam uma cifragem de qualidade. Esse tipo de experimento se faz necessário pois sabe-se que dentro de um espaço de chaves específico (por exemplo, todas as regras de raio 1) existem regras que são um pouco mais lentas para propagar a perturbação.

Em [Macedo, 2007] e [Magalhaes, 2010] testes similares a esse foram realizados. Nesses trabalhos anteriores foi possível utilizar todo o conjunto completo de regras dentro

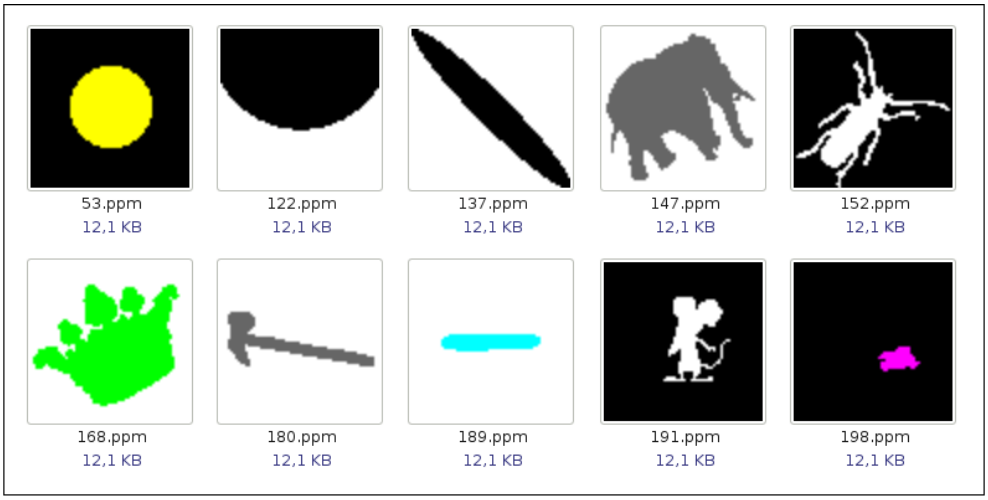


Figura 7.11: Conjunto das 10 piores imagens.

de um raio específico. Em [Macedo, 2007], foram avaliadas todas as regras unidimensionais de raio 2 sensíveis à direita, que formam um conjunto de 2^{16} núcleos diferentes. Em [Magalhaes, 2010] foram avaliados todas as regras bidimensionais (vizinhança do Von Neumman) de raio 1 sensíveis ao norte que também totalizam 2^{16} núcleos possíveis. Embora trabalhando em espaços diferentes, esses dois trabalhos anteriores chegaram a conclusões similares: foi obtido um número de passos mínimo para a cifragem, desde que regras com núcleos de baixa entropia (abaixo de 0,7 em [Magalhaes, 2010] e abaixo de 0,75 em [Macedo, 2007]) fossem excluídos do conjunto de chaves possíveis.

Entretanto, no caso tridimensional, um teste exaustivo do conjunto de chaves em um raio específico se mostrou inviável. As regras tridimensionais de raio 1 são formadas por núcleos de 64 bits. Portanto, no menor raio possível, temos um conjunto de 2^{64} regras com sensitividade ao norte, tornando inviável um teste exaustivo. Assim, elaboramos um teste com 4 conjuntos diferentes de regras de raio 1, cada um formado por 2^{16} regras. Para avaliar o efeito das regras com entropia mais baixa, dois desses conjuntos foram criados de forma a gerar regras mais regulares que também tendem a ter menor entropia, enquanto que os outros dois foram criados de forma a gerar regras mais irregulares. Para isso, cada conjunto foi gerado a partir de máscaras de núcleos de 48 bits fixos, sendo que os demais 16 bits foram preenchidos sequencialmente em todas as suas combinações de 0000000000000000 até 1111111111111111, gerando conjuntos de 2^{16} núcleos diferentes. A Tabela 7.16 apresenta as quatro máscaras utilizadas nesses experimentos, sendo que o "x" representa os bits que são variáveis na geração das regras e os bits especificados são fixos. Por exemplo, utilizando a máscara 1, o primeiro núcleo gerado é 0110100011000010101001010000101000101001100100100000101011100001 e o último é 0111110011000111111101010001111000111101100101110101101011110101, num total de 65536 núcleos. A primeira e a segunda máscaras geram as regras mais irregulares, enquanto que a terceira e a quarta geram os conjuntos de regras um pouco mais regulares. A Tabela 7.17 mostra as frequências das entropias dos núcleos gerados em cada uma das

máscaras utilizadas. Observe que o conjunto das duas máscaras ditas irregulares apresentam apenas núcleos de entropias acima de 0.75 que é o valor recomendado por [Macedo, 2007] e [Magalhaes, 2010]. Por outro lado, as máscaras ditas mais regulares apresentam alguns núcleos com entropia abaixo de 0.7 e nos permitiram avaliar melhor o desempenho dessas chaves, uma vez que o uso de chaves com baixa entropia pode acarretar uma diminuição na velocidade da propagação da perturbação.

Tabela 7.16: Conjunto das quatro máscaras com entropias mais altas.

Índice	Máscara
1	011x1x0011000x1x1x1x0101000x1x10001x1x0110010x1x0x0x1010111x0x01
2	0x11010x0x1101x0101x1x001xx10010x01011x0x1011x001x010x101x0011x0
3	000x1x1111000x1x1x1x10000x001x11000x1x1111000x1x0x0x00111x1110x0
4	10x010x1x100x10101x001x10x10110x1x010x010x10x01101x010x101x10x01

Tabela 7.17: Distribuição de frequências para as regras geradas a partir das quatro máscaras.

Classes das Entropias	Máscaras			
	1	2	3	4
$s \leq 0.65$	0	0	3	0
$0.65 < s \leq 0.70$	0	0	98	87
$0.70 < s \leq 0.75$	0	0	1587	2006
$0.75 < s \leq 0.80$	47	830	11281	14736
$0.80 < s \leq 0.85$	8556	29275	33088	35960
$0.85 < s \leq 0.90$	50509	34890	18891	12732
$s > 0.90$	6424	541	588	15
Total	65536	65536	65536	65536

O experimento consistiu em utilizar cada uma das 2^{16} chaves de cada conjunto na cifragem (cálculo de T pré-imagens consecutivas) de cada uma das 10 imagens selecionadas de 64×64 pixels. Essas imagens formam reticulados tridimensionais de dimensões 64×64×24. A finalidade é constatar qual quantidade de passos é suficiente para que todo o bloco de dados seja completamente cifrado e buscar identificar se existem regras mais lentas dentro de cada conjunto. Para isso, foi utilizada novamente a análise da propagação da perturbação na qual verificamos se o reticulado formado a partir do XOR entre a imagem original cifrada e a imagem perturbada retorna um percentual de zeros próximo a 0,5. A

quantidade de passos utilizados foi estimado a partir dos testes envolvendo o conjunto de reticulados de $64 \times 64 \times 64$ (Tabela 7.7) onde foram obtidos bons resultados com 30 passos e a verificação de que 20 passos de pré-imagem foi insuficiente nos testes iniciais com 200 imagens 64×64 (Tabela 7.14). Assim, foram utilizados 25 passos de pré-imagem nesse teste exaustivo.

A Tabela 7.18 apresenta os resultados médios obtidos por cada um desses conjuntos, identificado pela máscara utilizada (coluna A). A coluna B apresenta a média do percentual de zeros obtido no XOR dos reticulados tridimensionais. É possível observar que os resultados médios obtiveram resultados considerados satisfatórios até essa etapa: 50% de zeros com precisão de 3 casas decimais. Além disso, as entropias médias em cada canal também estão dentro das faixas esperadas. Entretanto, como esse resultado médio foi obtido em um conjunto significativo de regras (2^{16} em cada máscara), os casos que se afastaram da média sofreram uma análise mais minuciosa. As colunas C e D da Tabela 7.18 apresentaram as maiores e menores médias no percentual de zeros do XOR, respectivamente. É possível observar que, as maiores médias se afastaram significativamente da média esperada, sinalizando uma cifragem inadequada, na qual ainda resta um padrão não aleatório no XOR das duas imagens cifradas, similar ao observado no exemplo da imagem apresentada na Figura 7.4.

Tabela 7.18: Resultados médios na cifragem de imagens por 25 passos utilizando-se os 4 conjuntos de 2^{16} regras.

A	B	C	D	E	F	G	H	I	J	K
Índice da Máscara	Média da % de 0s $\overline{X}$	Desvio padrão	Menor valor	Maior valor	Entropia canal R	Desvio padrão R	Entropia canal G	Desvio padrão G	Entropia canal B	Desvio padrão B
1	0.500201	0.001712	0.491862	0.508494	0.939466	0.005396	0.939472	0.005393	0.939472	0.005392
2	0.500211	0.001717	0.492828	0.508270	0.939471	0.005391	0.939479	0.005388	0.939469	0.005394
3	0.500223	0.001729	0.493123	0.508514	0.939469	0.005391	0.939475	0.005390	0.939466	0.005397
4	0.500208	0.001717	0.492930	0.508382	0.939473	0.005388	0.939477	0.005389	0.939478	0.005386

Além disso, foi observado um número bem maior de regras com percentual de zeros no XOR acima de 0,5, do que abaixo dessa média. Os gráficos da Figura 7.12 apresentam para cada máscara (conjunto de 2^{16} regras) a diferença entre o percentual de zeros no XOR e o valor médio esperado (0,5). Para gerar esses gráficos, cada conjunto de regras foi ordenado pelo resultado médio no percentual de zeros no XOR e depois foi calculado o valor: $0.5 - |0.5 -$ percentual de 0s$|$, conforme é apresentado no Algoritmo G.2. O eixo x dos gráficos na Figura 7.12 representa as 655.360 cifragens realizadas em cada conjunto de regras (2^{16} regras por 10 imagens). Devido à aleatoriedade típica de um método da cifragem, seria esperada uma distribuição normal em torno de 0.5: $C = (\frac{n}{2}, 0.5)$. Entretanto, pelos gráficos apresentados é possível observar que as curvas apresentam as linhas do lado esquerdo mais prolongadas, indicando que ocorreu um número maior de

regras com percentual de zeros acima de 0,5. Essa observação é justificada pelo fato de que quando a perturbação não foi totalmente propagada, a parcela nos pares de imagens cifradas que permanece similar, retorna um número de 0s maior.

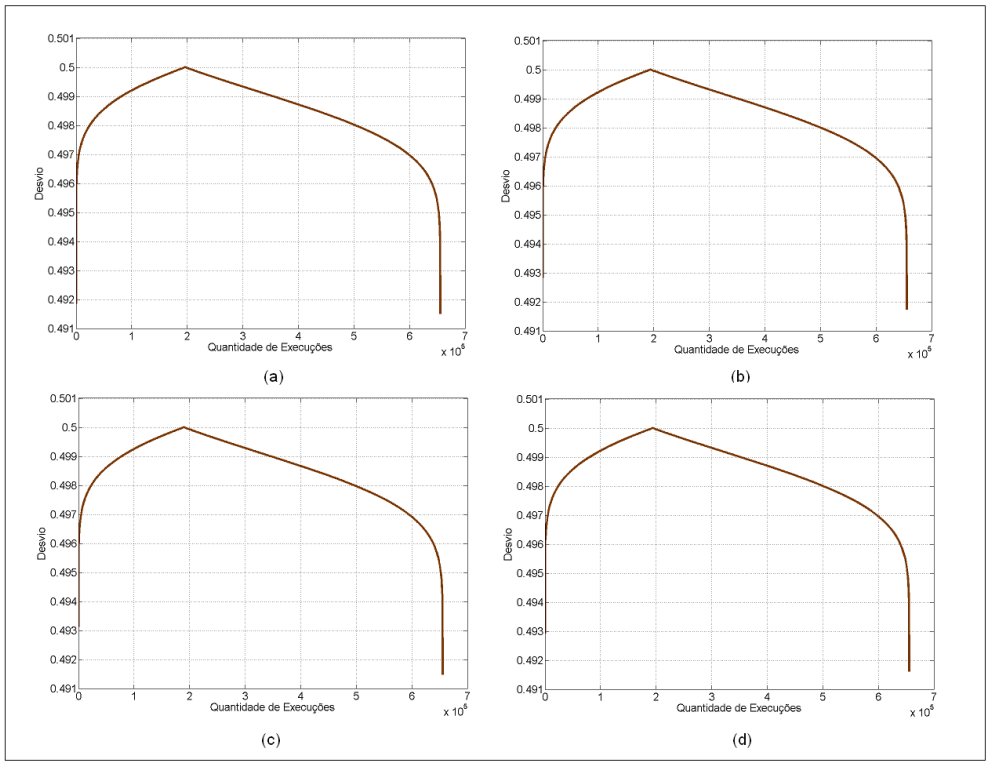


Figura 7.12: Gráficos dos desvios do percentual de zeros em relação a 50% utilizando-se 25 passos na cifragem. (a) Conjunto de núcleos formados a partir da máscara 1. (b) Conjunto de núcleos formados a partir da máscara 2. (c) Conjunto de núcleos formados a partir da máscara 3. (d) Conjunto de núcleos formados a partir da máscara 4.

A Seção G.4 do Apêndice G apresenta dois exemplos de imagens cifrados por 25 passos de tempo, sendo que uma delas realizou a cifragem com sucesso e a outra falhou por apresentar um padrão não-aleatório no XOR realizado entre a imagem original cifrada e a imagem perturbada cifrada. O caso de sucesso foi cifrado utilizando-se uma regra de entropia 0,8535. Por outro lado, o caso de fracasso foi cifrado utilizando-se uma regra de entropia 0,5676.

A partir da análise visual do XOR dos pares de imagens geradas a exemplo da análise apresentada no Apêndice G e a partir dos gráficos apresentados na Figura 7.12 foi possível concluir que uma parte considerável de imagens não foi cifradas adequadamente utilizando apenas $T = 25$ passos de tempo. Essa conclusão foi tomada a partir da distribuição de valores ao longo do gráfico, que nem sempre, apresentaram o ponto central próximo do ponto C esperado. Por outro lado, não foi identificada nenhuma regra crítica, ou seja, não ocorreu nenhum caso de cifragem com percentual de XOR abaixo de 30%. Dessa forma, a principal explicação para esse desempenho não satisfatório de algumas cifragens foi a utilização de um número insuficiente de pré-imagens. Assim, um novo experimento foi

elaborado para verificar se mais cinco passos de pré-imagem seriam suficientes para uma cifragem adequada.

O novo conjunto de dados para este teste foi reduzido devido à quantidade de tempo de processamento dos mesmos e à quantidade de recursos para realizar o mesmo. A quantidade de passos de pré-imagem adotada nesse caso foi de $T = 30$. Foram utilizadas apenas duas imagens para a execução desse teste: 53.ppm e 122.ppm. O número de regras também foi reduzido, nesse caso, foi utilizado um subconjunto de 2^{13} de núcleos gerados para cada uma das máscaras apresentadas na Tabela 7.16. Assim, um conjunto de $n = 16384$ dados foi gerado para cada uma das máscaras (2 imagens e 8192 regras). O resultado médio geral para cada uma das máscaras está apresentado na Tabela 7.19. Essa tabela não apresenta diferença significativa em relação à Tabela 7.18. Entretanto, pelos valores na coluna E, é possível observar que as médias no percentual de zeros do XOR mais altas apresentaram uma ligeira queda. Além disso, foi possível observar uma queda significativa no número de casos que retornaram um percentual de zeros acima de 0.5. Os gráficos da Figura 7.13 confirmam essa informação. Eles foram gerados utilizando-se a mesma metodologia usada para gerar os gráficos da Figura 7.12. É possível perceber uma clara mudança na distribuição dos valores, sendo que nos novos gráficos o percentual de zeros se aproxima bastante de uma distribuição normal. Portanto, com os resultados experimentais foi possível estabelecer que a quantidade mínima de pré-imagens que devem ser adotadas para a realização da cifragem do banco de imagens 64×64 pixels é igual a 30 passos de pré-imagem. Além disso, foi possível observar que nenhuma das 2^{15} regras avaliadas considerando-se as 4 máscaras apresentou uma cifragem de baixa qualidade, apesar de existirem núcleos de baixa entropia nos conjuntos das máscaras 3 e 4.

Tabela 7.19: Resultados médios na cifragem de imagens por 30 passos utilizando-se os 4 conjuntos de 2^{12} regras.

A	B	C	D	E	F	G	H	I	J	K
Índice da Máscara	Média da % de 0s $\overline{X}$	Desvio padrão	Menor valor	Maior valor	Entropia canal R	Desvio padrão R	Entropia canal G	Desvio padrão G	Entropia canal B	Desvio padrão B
1	0.500017	0.001596	0.493886	0.506093	0.944861	0.000361	0.944860	0.000363	0.944856	0.000360
2	0.499984	0.001597	0.492961	0.506887	0.944859	0.000359	0.944851	0.000358	0.944853	0.000362
3	0.500006	0.001590	0.492533	0.507385	0.944861	0.000361	0.944857	0.000359	0.944866	0.000359
4	0.499992	0.001585	0.494344	0.506409	0.944857	0.000363	0.944850	0.000366	0.944854	0.000363

Embora tenhamos concluído o objetivo principal do experimentos que era verificar o número de passos de pré-imagem para uma cifragem de qualidade, considerando-se um conjunto expressivo de regras ainda realizamos uma investigação adicional. Uma vez que tínhamos o conhecimento do descarte de regras com baixa entropia nos testes com os modelos HCA e THCA por apresentarem uma performance indesejada, elaboramos dois conjuntos a partir de máscaras ainda mais regulares (entropias abaixo de 0.7) para

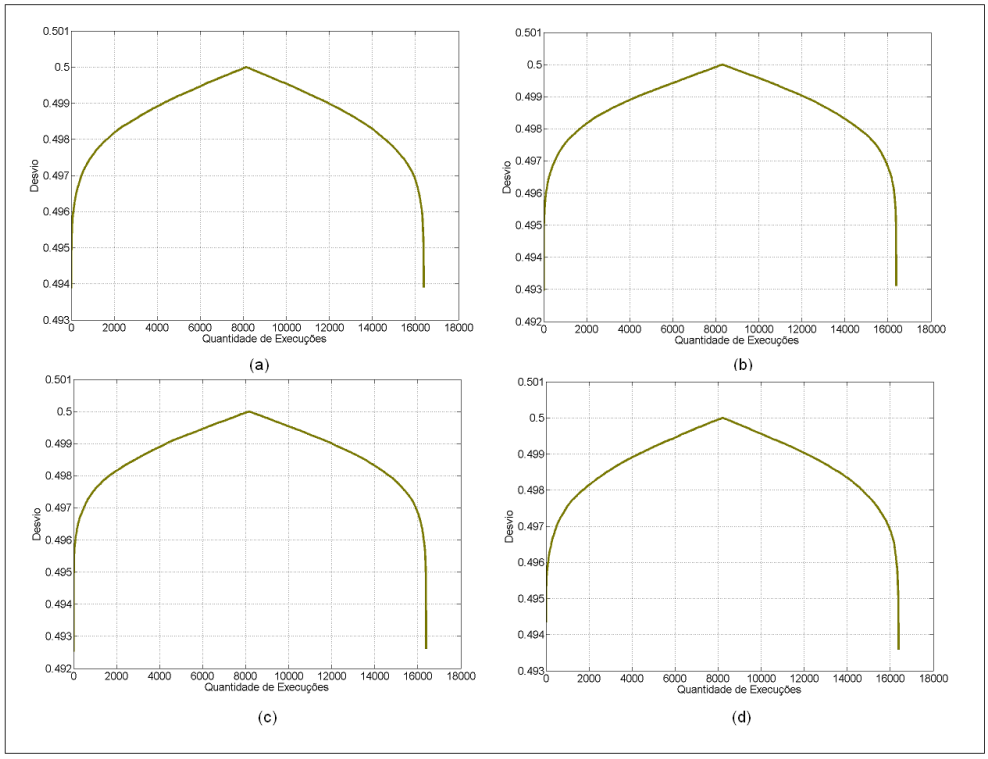


Figura 7.13: Gráficos dos desvios do percentual de zeros em relação a 50% utilizando-se 30 passos na cifragem. (a) Conjunto de núcleos formados a partir da máscara 1. (b) Conjunto de núcleos formados a partir da máscara 2. (c) Conjunto de núcleos formados a partir da máscara 3. (d) Conjunto de núcleos formados a partir da máscara 4.

confirmar as análises realizadas em [Macedo, 2007] e [Magalhaes, 2010]. As máscaras quinto e sexto conjuntos de núcleos estão apresentados na Tabela 7.20 e foram definidos de maneira similar aos conjuntos utilizados nos experimentos anteriores. Entretanto, esses novos conjuntos de núcleos foram definidos com apenas 8 bits livres demarcados por "x" e foram preenchidos sequencialmente desde 00000000 até 11111111, resultando em dois conjuntos de 2^8 chaves. Além disso, um sétimo conjunto de 50 regras bastante regulares e apresentaram entropias muito baixas foi definido manualmente. Esse último conjunto está apresentado no Apêndice G.10. A Tabela 7.21 apresenta a distribuição das entropias dos núcleos. Os resultados médios para cada um dos conjuntos estão apresentados na Tabela 7.22 computados na cifragem das 10 imagens da Figura 7.11 por 25 passos de tempo.

Tabela 7.20: Conjunto das duas máscaras com entropias mais baixas.

Índice	Máscara
5	x0000000x1111111x1111111x0000000x0000000x1111111x1111111x0000000
6	1111111xx00000001111111x0000000xx11111110000000x1111111xx0000000

Tabela 7.21: Distribuição de frequências para as regras mais regulares.

Classes das Entropias	Máscaras		
	5	6	7
$s \leq 0.3$	0	0	24
$0.35 < s \leq 0.40$	0	0	20
$0.40 < s \leq 0.45$	0	0	0
$0.45 < s \leq 0.50$	16	0	4
$0.50 < s \leq 0.55$	66	0	0
$0.55 < s \leq 0.60$	94	108	2
$0.60 < s \leq 0.65$	64	120	0
$s > 0.65$	16	28	0
Total	256	256	50

Tabela 7.22: Resultado médios na cifragem de imagens 64×64 pixels obtidos pelo conjunto das máscaras mais regulares utilizando-se 25 passos de pré-imagem.

A	B	C	D	E	F	G	H	I	J	K
Índice da Máscara	Média da % de 0s $\overline{X}$	Desvio padrão	Menor valor	Maior valor	Entropia canal R	Desvio padrão R	Entropia canal G	Desvio padrão G	Entropia canal B	Desvio padrão B
5	0.503844	0.003763	0.497030	0.530101	0.939466	0.005396	0.939472	0.005393	0.939472	0.005392
6	0.515344	0.007574	0.501078	0.585805	0.937053	0.006592	0.937057	0.007115	0.937040	0.006812
7	0.598880	0.145977	0.502462	0.999908	0.822144	0.244507	0.822558	0.243694	0.822060	0.244407

É possível observar que as máscaras 5 e 6 apresentam entropias bem inferiores e o percentual médio de zeros na imagem do XOR é superior aos apresentados na Tabela 7.18 que também utilizou $T = 25$ passos de tempo para a cifragem. Ou seja, é possível perceber que as regras com menor entropia têm maior tendência a realizar cifragens que propaguem a perturbação de maneira mais lenta. Em relação aos conjuntos gerados com as máscaras 5 e 6 é possível deduzir que essas regras precisariam de um número bem maior de passos para realizar a cifragem de forma adequada. Observamos que o pior caso retornou um percentual de zeros em torno de 58%, indicando uma cifragem bastante inadequada.

O sétimo conjunto de regras utilizado apresentou pior percentual médio de zeros de todos os conjuntos de regras que foram submetidos a testes nessa dissertação. Diferentemente do conjunto das quatro primeiras máscaras adotadas, é possível perceber que o percentual de zeros observado (0,598880) está bem distante do desejado para uma cifragem de boa qualidade. Outro fator que difere o sétimo conjunto de dados dos demais é em relação ao valor da maior porcentagem de zeros observado (0,999908). Isso significa que

nesse conjunto existem regras que não são capazes de cifrar nem mesmo se a quantidade de pré-imagens fosse aumentada. Assim, esses três conjuntos de regras foram descartados e apenas os conjuntos gerados pelas 4 máscaras iniciais foram utilizados nos experimentos com $T = 30$ passos de pré-imagem. Como conclusão final, foi possível confirmar que os valores das entropias dos núcleos das chaves recomendados anteriormente em [Macedo, 2007] também devem ser mantidos no modelo criptográfico tridimensional aqui proposto. Ou seja, as palavras binárias com entropia abaixo de 0,7 não devem ser utilizadas como chaves no sistema 3DHCA.

7.4.3 Teste comparativo entre os modelos tridimensional e bidimensional

Um modelo bidimensional similar ao THCA [Magalhaes, 2010] foi implementado seguindo as mesmas características do modelo final de cifragem tridimensional implementado nessa dissertação. O modelo 2D foi implementado a fim de contrastar com o modelo tridimensional. Um conjunto de 100 regras com entropias acima de 0.7 foram utilizadas em ambos os modelos. Esse teste consiste em duas etapas. A primeira etapa considera um conjunto de imagens de 64×64 pixels no padrão RGB no formato "Portable Pixel Map" (*.ppm*) com pixels somente nas cores preto e branco e estão apresentadas na Figura 7.14 (a). Para cifrá-las utilizando o algoritmo de criptografia tridimensional foi necessário transpor os bits dessa imagem para um reticulado tridimensional de dimensões $64 \times 64 \times 24$ e cifrar por $T = 30$ passos de tempo conforme especificado anteriormente. Para cifrar essas mesmas imagens utilizando o algoritmo de criptografia bidimensional, um reticulado no formato matricial foi adaptado para as seguintes dimensões $(64 \times 8) \times (64 \times 3)$. A segunda etapa consistiu em cifrar 10 imagens binárias (pixels pretos ou brancos) de 64×64 bits no formato "Portable Bit Map" (*.pbm*) que estão mostradas na Figura 7.14 (b). Uma adaptação para a transformação dessa imagem para um reticulado tridimensional se fez necessária resultando em um reticulado de $16 \times 16 \times 16$. A imagem binária 64×64 bits também foi cifrada pelo algoritmo de criptografia bidimensional. A tabela 7.23 indica a quantidade de passos necessários para cifrar em cada um dos modelos é apresentada a seguir.

Tabela 7.23: Comparação da quantidade de passos de pré-imagem nos modelos 3D × 2D

Resultado Geral dos resultados para $T = 30$							
A	B	C	D	E	F	G	H
Algoritmo utilizado	Padrão utilizado	Formato da imagem	Tamanho da imagem	Dimensões do reticulado	Quantidade de passos	Média da % de zeros $\overline{X}$	Desvio padrão
3D	RGB	(.ppm)	64×64 pixels	$64 \times 64 \times 24$	30	0.500074	0.001586
2D				$(512) \times (192)$	120	0.500530	0.003139
3D	Binário	(.pbm)	64×64 bits	$16 \times 16 \times 16$	10	0.500143	0.007707
2D				64×64	60	0.499901	0.007871

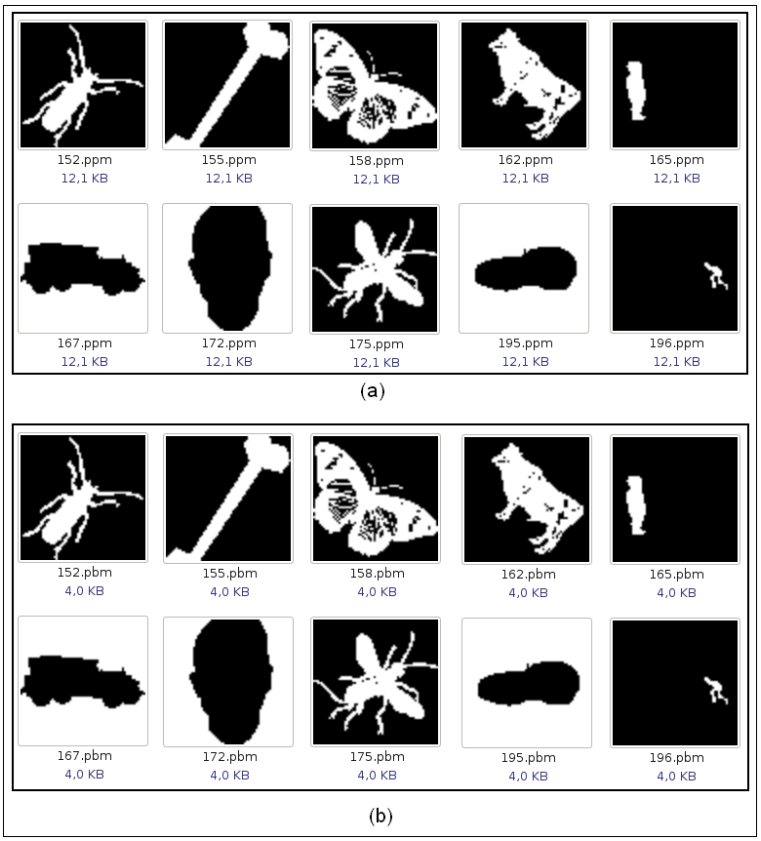


Figura 7.14: (a) Imagens Preto e Branco no padrão RGB. (b) Imagens Binárias.

A Figura 7.15 apresenta os gráficos dos desvios dos modelos de criptografia bidimensional e tridimensional para imagens de 64×64 pixels apresentadas na Figura 7.14 (a). A Figura 7.16 apresenta os gráficos dos desvios para as imagens de 64×64 bits apresentadas na Figura 7.14. Os quatro gráficos apresentados indicam que as quantidades de pré-imagens necessárias para a propagação da perturbação foram suficientes para todas as formas de execuções apresentadas, visto que os gráficos apresentados apresentaram uma distribuição normal em torno do valor 0,5.

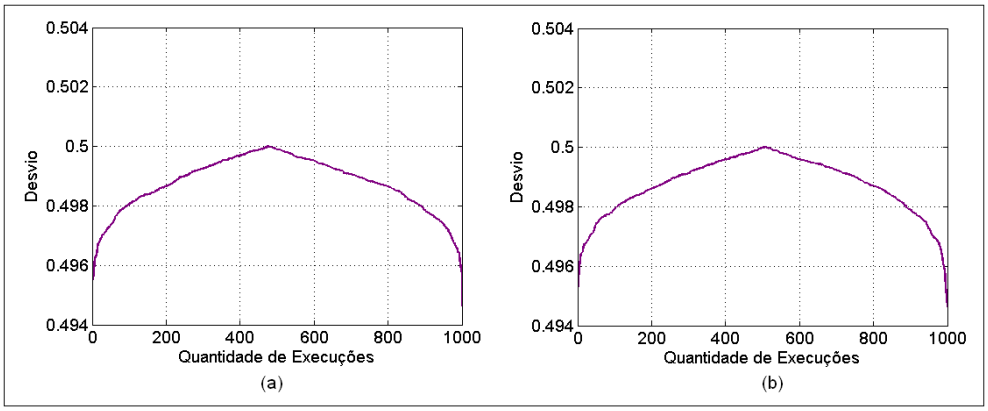


Figura 7.15: (a) Gráfico do desvio do modelo 3D com imagem no padrão RGB ($N = 64 \times 64 \times 24$). (b) Gráfico do desvio do modelo 2D com imagem no padrão RGB ($N = (64 \times 8) \times (64 \times 3)$).

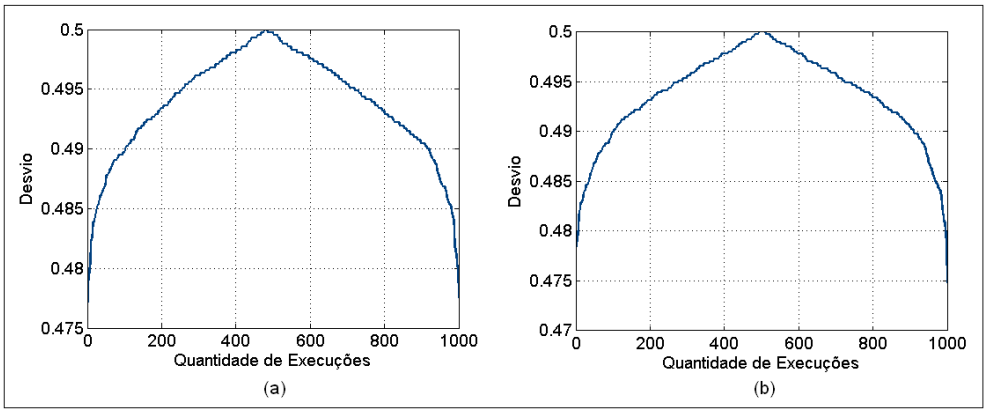


Figura 7.16: (a) Gráfico do desvio do modelo 3D com imagem binária ($N = 16 \times 16 \times 16$). (b) Gráfico do desvio do modelo 2D com imagem binária ($N = 64 \times 64$).

Tomando como base a tabela de resultados, é possível perceber que o algoritmo tridimensional é superior em relação à quantidade de passos de pré-imagens gastos para cifrar a mesma quantidade de dados. Outra característica importante é que o modelo de criptografia 3DHCA pode cifrar com um nível de paralelismo maior, se implementado em uma arquitetura altamente paralelizável. Portanto, na cifragem de um mesmo conjunto de imagens, o método 3DHCA além de utilizar um número menor de passos de pré-imagem, irá utilizar um número bem menor de ciclos de *clock* para realizar o cálculo dessas pré-imagens.

Capítulo 8

Conclusão e trabalhos futuros

Nesta dissertação foi investigado um novo modelo de criptografia baseado em autômatos celulares tridimensionais híbridos que utilizam o cálculo de pré-imagem como processo de cifragem. Esse modelo foi denominado 3DHCA (*Three-dimensional Hybrid Cellular Automata*) e a principal aplicação investigada foi a utilização do mesmo na criptografia de imagens digitais no padrão RGB. Além disso, uma análise do modelo precursor HCA baseado em ACs unidimensionais híbridos foi realizada.

A investigação de modelos de criptografia baseados em Autômatos Celulares tem sido conduzida por diversos pesquisadores. Dentre os modelos analisados que empregam a evolução dos ACs, onde o texto claro corresponde ao reticulado inicial e a chave corresponde às regras empregadas, existem basicamente duas abordagens. Uma delas faz uso das propriedades algébricas de algumas regras aditivas [Nandi et al., 1994], [Ganguly et al., 2000] e [Sen et al., 2002]. A segunda abordagem, na qual este trabalho se insere, refere-se ao uso do cálculo de pré-imagens [Gutowitz, 1995], [Oliveira et al., 2004], [Macedo, 2007] e [Magalhaes, 2010]. A primeira abordagem possui a vantagem de explorar diretamente o paralelismo intrínseco dos ACs, uma vez que se baseia apenas na evolução para frente de ACs periódicos, tanto na cifragem quanto na decifragem. Por outro lado, a característica aditiva das regras utilizadas nesse modelos carrega uma linearidade inerente que foi apontada como fraqueza desses métodos. Na segunda abordagem por outro lado, as regras de cifragem exibem dinâmica caótica, o que tornamos métodos resultantes mais robustos à técnica de criptoanálise linear. Em contrapartida, a evolução para trás precisa ser aliada ao uso das regras caóticas, quebrando o paralelismo intrínseco da evolução para frente dos ACs. Assim, uma preocupação inerente aos métodos que utilizam o cálculo de pré-imagens é a paralelização das etapas, uma vez que esse cálculo possui uma natureza essencialmente sequencial. Nesta dissertação, um novo modelo de criptografia baseado no cálculo de pré-imagens de blocos tridimensionais foi proposto. A principal motivação para o estudo dessas estruturas em criptografia está no fato das mesmas permitirem um alto grau de paralelismo. Embora, o modelo tenha sido implementado e testado na forma sequencial, foi possível mostrar que o mesmo tem forte potencial para implementação em

uma plataforma paralela.

No modelo 3DHCA, o AC utiliza na sua evolução temporal duas regras com vizinhança de Von Neumann tridimensionais: uma chamada regra principal, que provê a caoticidade necessária ao sistema criptográfico, e a regra de contorno, que garante que sempre exista uma pré-imagem válida. Essas duas regras utilizadas no modelo tridimensional são sensíveis a um dos extremos da vizinhança. Essa característica provê ao AC não-homogêneo um comportamento caótico que faz com que o bloco de dados após a cifragem realizada pelo AC seja de boa qualidade, o que pode ser comprovado pela análise de entropia do texto cifrado e pelas análises de perturbação de um bit, tanto no texto claro, quanto na chave criptográfica. Os ACs não-homogêneos baseados em duas regras com sensitividade já haviam sido investigados no modelo de criptografia HCA [Macedo, 2007] registrado no INPI como solicitação de patente (PI0703188-2) em [Oliveira and Macedo, 2007] e no modelo bidimensional THCA [Magalhaes, 2010]. A partir da definição do sistema 3DHCA e dos testes realizados nessa dissertação uma nova patente está sendo elaborada e deve ser registrada no INPI em setembro de 2012.

A principal motivação para estudar o modelo baseado em uma estrutura tridimensional foi sua aplicabilidade na cifragem de imagens. O modelo 3DHCA foi elaborado para cifrar uma imagem digital em um único bloco de dados tridimensional. Ao utilizar um único bloco em uma cifragem de ACs tridimensionais, é possível que uma pequena mudança na célula de um extremo intervenha no estado da célula do centro ou no estado da célula de outro extremo, essa teoria pode ser explicada pela teoria do caos. Caso o bloco único tenha dimensões muito grandes, seria necessário a utilização da quebra do bloco único em pequenos blocos. Cada um desses blocos seria cifrado de forma independente e posteriormente os blocos seriam combinados, utilizando algum modo de operação usual de criptografia (CBC, CBF, OFB, CTR, etc) a fim de gerar o bloco de dados criptografados. Um dos problemas encontrados ao aplicar os modos de operação na criptografia é que a cifragem resultante é de pior qualidade (zonas de texturas) em relação ao modelo que utiliza bloco único. Dessa forma, a eficácia do método criptográfico fica atrelada ao modo de operação utilizado. Além disso, quando utiliza-se algum modo de operação o paralelismo do modelo tridimensional baseado em ACs é quebrado.

A grande vantagem de se utilizar um modelo 3DHCA para a criptografia de imagens é o mapeamento dos dados da imagem para uma dimensão superior que a bidimensional (formato original das imagens digitais no padrão RGB). Assim, ao utilizarmos a terceira dimensão, a cifragem pode ser realizada levando-se em conta apenas a profundidade do bloco e não necessariamente as dimensões da imagem. Ou seja, para o modelo 3DHCA existe paralelismo em dois eixos na cifragem. Assim, a cifragem quando aplicada à imagens possui melhor qualidade, pois o arranjo bidimensional das mesmas é considerado. No caso da decifragem, é permitido paralelismo nos três eixos tornando esse processo muito rápido se implementado em uma arquitetura paralela. Por outro lado, no modelo THCA

o paralelismo no processo de cifragem é observado em um único eixo o que degrada o tempo de processamento desse modelo em dados tridimensionais em relação ao modelo 3DHCA. De acordo com os experimentos realizados nessa dissertação e com a observação de [Magalhaes, 2010] foi constatado que o melhor eixo para realização da cifragem, relacionado à escolha da célula para estabelecer a sensitividade da regra, é aquele de maior dimensão do bloco de dados, visto que com essa técnica é possível utilizar menos passos para a cifragem nos dois modelos.

8.1 Especificação do modelo 3DHCA

Os experimentos realizados indicam que o modelo 3DHCA possui as propriedades de confusão e difusão necessárias a um bom método criptográfico, além de ser robusto à ataques do tipo criptoanálise diferencial. A partir desses testes, pode-se verificar que o processo de cifragem deve possuir um atraso no paralelismo proporcional de quatro ciclos de *clock* (proporcional à $4r$) em relação ao cálculo da próxima pré-imagem, enquanto que o processo de decifragem é totalmente paralelo. As chaves criptográficas do modelo 3DHCA padrão investigado nessa dissertação têm tamanho de 64 bits, ou seja, regras de raio 1. Entretanto, esse modelo pode ser generalizado facilmente para o uso de regras tridimensionais de raio maior, que definem um espaço de chaves de maior cardinalidade. Por exemplo, caso regras com vizinhança de Von Neumann de raio 2 fossem empregadas, o número de células da vizinhança subiria para 13 células, o núcleo seria definido por 2^{12} bits e o espaço de chaves teria cardinalidade 2^{4096}. A Figura 8.1 ilustra o formato da vizinhança tridimensional de raio 2 que poderia ser aplicada no método. Os núcleos/chaves recomendados devem possuir entropia superior à 0,7 para uma boa propagação na perturbação com um número razoável de pré-imagens. As próximas subseções apresentarão as especificações da quantidade de pré-imagens e do arranjo de blocos para cada tipo de arquivo a ser cifrado, considerando-se uma especificação padrão para regras de raio 1.

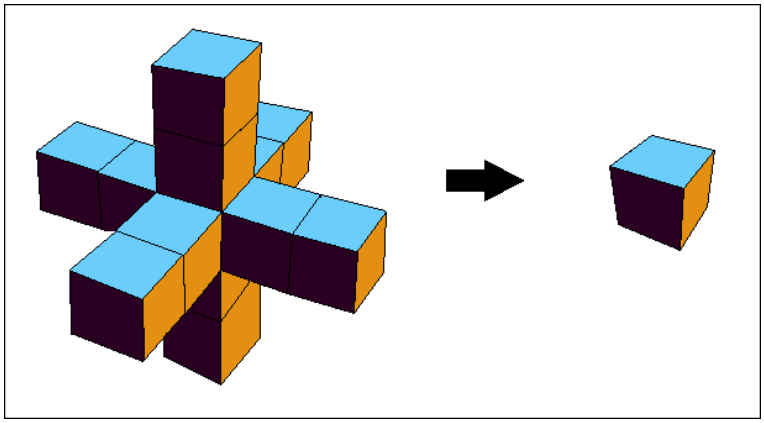


Figura 8.1: Representação da vizinhança da regra de raio 2.

8.1.1 Especificação padrão para o 3DHCA na cifragem de imagens coloridas

A recomendação é aplicar o modelo 3DHCA utilizando a imagem em um único bloco tridimensional. O primeiro arranjo sugerido considera que o bloco de dados é construído apenas transformando uma imagem de $m \times n$ pixels no padrão RGB em um reticulado tridimensional de $m \times n \times 24$ bits, conforme apresentado no Capítulo 6. Os tamanhos de blocos sugeridos para a cifragem de imagens de tamanho $m \times n$ onde $k = \max(m, n)$ são apresentados na Tabela 8.1. Caso seja necessário trabalhar com blocos de dimensões superiores aos apresentados na Tabela 8.1, basta utilizar algum modo de operação: CBC, CBF, OFB, CTR, dentre outros conhecidos na literatura [Stinson, 2006].

Tabela 8.1: Arranjo padrão para cifragem de imagens coloridas no padrão RGB.

$\leq k$	Arranjo	Quantidade de de imagens cifradas	(T) 50% do valor de k
32	$m \times n \times 24$	1	16
64	$m \times n \times 24$	1	32
128	$m \times n \times 24$	1	64
256	$m \times n \times 24$	1	128
512	$m \times n \times 24$	1	256

Um arranjo otimizado para a cifragem de imagens $m \times n$ pixels é apresentado na Tabela 8.2. Com a utilização desse arranjo é possível que se utilize para cada tamanho de k a mesma quantidade de passos de pré-imagem para cifrar uma quantidade maior de dados.

Tabela 8.2: Arranjo otimizado para cifragem de imagens coloridas no padrão RGB.

$\leq k$	Arranjo	Quantidade de de imagens cifradas	Quantidade de passos (T)
32	$m \times n \times 24$	1	16
64	$m \times n \times 48$	2	32
128	$m \times n \times 96$	4	64
256	$\frac{m}{2} \times \frac{n}{2} \times 96$	1	64
512	$\frac{m}{2} \times \frac{n}{2} \times 192$	2	128

8.1.2 Especificação padrão para o 3DHCA na cifragem de imagens na escala cinza

Para a cifragem de imagens na escala cinza faz-se necessária a definição de novos arranjos para os blocos para melhor aproveitamento dos passos de pré-imagem. É sabido que as imagens na escala cinza possuem representatividade de apenas 256 cores (1 *byte* por cor). Assim, uma transformação de uma imagem de $m \times n$ pixels na escala cinza representa um bloco tridimensional de tamanho $m \times n \times 8$. Isso significa que para compor um bloco várias imagens podem ser concatenadas. A concatenação dessas imagens $m \times n$ pixels (escala cinza) para a formação dos reticulados tridimensionais é sugerida de acordo com a Tabela 8.3.

Tabela 8.3: Arranjo para cifragem de imagens na escala cinza.

$\leq k$	Arranjo	Quantidade de de imagens cifradas	Quantidade de passos (T)
32	$m \times n \times 32$	4	20
64	$m \times n \times 64$	8	32
128	$m \times n \times 128$	16	64
256	$\frac{m}{2} \times \frac{n}{2} \times 128$	4	64
512	$\frac{m}{2} \times \frac{n}{2} \times 256$	8	128

8.1.3 Especificação padrão para o 3DHCA na cifragem de imagens binárias

A especificação da quantidade de pré-imagens a ser utilizada na cifragem de imagens binárias de tamanhos $m \times n$ bits onde $k = \max(m, n)$ é baseada na Tabela 8.4.

Tabela 8.4: Arranjo para cifragem de imagens binárias.

$\leq k$	Arranjo	Quantidade de de imagens cifradas	Quantidade de passos (T)
32	$m \times n \times 32$	32	20
64	$m \times n \times 64$	64	32
128	$\frac{m}{2} \times \frac{m}{2} \times 64$	16	32
256	$\frac{m}{2} \times \frac{n}{2} \times 128$	32	64
512	$\frac{m}{4} \times \frac{n}{4} \times 128$	8	64

8.1.4 Especificação padrão para o 3DHCA na cifragem de textos lineares

Apesar dos testes terem sido realizados com imagens, o método proposto também pode ser utilizado para cifragem de textos. A sugestão para a cifragem de textos, é utilizar blocos de 4096 bits, que serão arranjados como blocos tridimensionais de $16 \times 16 \times 16$ bits. De acordo com os experimentos, para essa configuração, são necessários 10 passos de pré-imagem para uma cifragem de boa qualidade.

8.2 Definição formal do modelo de criptografia 3DHCA

A criptografia do modelo 3DHCA pode ser definida por uma 6-tupla($\mathcal{P}$, $\mathcal{K}$, T, $\mathcal{C}$, ε, D) que satisfaz as seguintes condições:

1. $\mathcal{P}$ é um conjunto de reticulados iniciais de tamanho $m \times n \times p$, também denominados blocos originais;
2. T representa a quantidade de passos;
3. $\mathcal{K}$ é o conjunto das chaves possíveis, cada chave $k \in \mathcal{K}$ é composta por uma palavra de 64 bits, sendo este o tamanho da regra principal. A partir do núcleo, uma regra principal sensível a um dos extremos N, S, L, O, K ou F e com entropia recomendada superior à 0,7 deve ser construída. Além disso,e uma regra de contorno que possui a mesma sensitividade da regra principal é escolhida de um conjunto de apenas doze regras possíveis;
4. $\mathcal{C}$ é um conjunto de textos cifrados de tamanho $m \times n \times p$;
5. Para cada $k \in \mathcal{K}$, tem uma regra de criptografia $e_k \in \varepsilon$, dada pelo cálculo de pré-imagem do modelo de AC híbrido tridimensional e aplicando um deslocamento proporcional a $4r$, e uma correspondente regra de descriptografia $d_k \in D$, dada pela evolução para frente do modelo de AC híbrido tridimensional e aplicando um deslocamento proporcional a $4r$. Cada $e_k : \mathcal{P} \to \mathcal{C}$ e $d_k : \mathcal{C} \to \mathcal{P}$ são funções tais que $d_k(e_k(\mathcal{P})) = \mathcal{P}$.

8.3 Trabalhos futuros

Como continuidade dessa dissertação várias investigações e desenvolvimentos podem ser conduzidas tanto com o objetivo de aperfeiçoar o modelo aqui proposto como para a verificação do poder de paralelismo do mesmo na cifragem. A implementação em hardware do método criptográfico proposto é a continuidade mais direta desse trabalho, uma vez que toda a motivação para o emprego de ACs em criptografia reside na possibilidade de

implementação eficiente. Dessa forma, seria empregado de fato o paralelismo alcançado pelo modelo. Durante a disciplina de mestrado "Hardware Reconfigurável" ministrada pelo Prof. Dr. Daniel Mesquita, uma implementação da evolução para frente e dos modelos de criptografia propostos por [Gutowitz, 1995] e [Macedo, 2007] foi realizada. O principal algoritmo testado na placa foi o HCA (sem rotação de núcleo) e foi implementado em VHDL. Os testes foram realizados na placa Altera DE2 com o FPGA Cyclone II, que possui disponíveis 33216 elementos lógicos e 672 pinos. Os resultados foram obtidos fazendo a verificação entre a performance em relação ao tempo entre as linguagens C e VHDL. O cálculo principal da criptografia foi realizado na linguagem VHDL (parte altamente paralelizável) e a parte não-paralelizável foi realizada na linguagem C padrão. A parte feita em C foi necessária para facilitar a quebra em blocos do texto. Como os algoritmos de quebra de textos em blocos são feitos de forma sequencial, essa abordagem não prejudicou o desempenho do algoritmo HCA. De acordo com os experimentos realizados o modelo apresentou o mesmo nível de segurança da implementação sequencial. Nesse caso, o poder de processamento aumentou significativamente, o que tornam as implementações em hardware mais adequadas para os modelos de criptografia baseados no cálculo de pré-imagens de ACs. As estruturas do hardware podem ser utilizadas com sucesso para proteger a informação. Existem outras pesquisas que apontam a utilização de placas FPGA [Donthi and Haggard, 2003], [Zambreno et al., 2005] e [Karmakary and R., 2011] na implementação de métodos criptográficos eficientes.

Embora o método proposto como padrão nesta dissertação tenha apresentado bons resultados em todos os experimentos efetuados, mais testes devem ser realizados para confirmar a robustez do mesmo. Por exemplo, podem ser empregados diferentes métodos para avaliar a aleatoriedade produzida nos textos cifrados, além da entropia que foi largamente explorada nesse trabalho. Poderiam ser avaliadas outras técnicas de criptoanálise, além da diferencial, como a criptoanálise linear. Testes com diferentes tamanhos de blocos, regulares ou irregulares, poderiam ser utilizados para refinar ainda mais o modelo. Além disso, uma comparação entre a versão proposta originalmente em [Macedo, 2007] que utiliza a regra de contorno baseada no primeiro bit da regra principal e a adaptação proposta nesse trabalho que utilizou a alternação da borda em cada uma das pré-imagens calculadas, poderia ser testado para verificação de qual técnica apresenta maior segurança. A investigação futura do método 3DHCA com raios maiores também se faz necessárias, visto que hoje a utilização de chaves de 64 bits já é considerada limitada pela evolução do hardware devido a proposição do método DES [Stallings, 2002]. Assim, outra investigação existe como continuidade do presente trabalho é a implementação e experimentação do método com regras de raio 2 para elaboração de novas especificações de tamanho de bloco.

Uma adaptação do modelo 3DHCA para a criptografia utilizando o padrão de coloração HSI também poderia ser elaborada conforme sugerido pela Profa. Dra. Denise Guliato. O

padrão HSI (*Hue Saturation Intensity*) é a abreviatura para o sistema de cores formadas pelas componentes *Hue* (matiz), *Saturation* (saturação) e *Intensity* (intensidade). Nesse padrão, o parâmetro H é obtido através de operações algébricas a partir canais R, G e B (que representam as cores). Nesse caso, há fortes indícios de que apenas o parâmetro H em uma imagem (padrão HSI) precisaria ser cifrado, diminuindo significativamente a quantidade de dados que seriam utilizados pelo 3DHCA e consequentemente a quantidade de passos para a cifragem de uma única imagem.

Outra investigação que poderia ser conduzida é o estudo do modelo 3DHCA com três blocos. Para minimizar o problema da cifragem resultante com uma quantidade de cores reduzida apontado na Seção 6.3 uma regra distinta e com sensitividade distinta poderia ser aplicada a cada um dos três blocos, fazendo uma rotação do núcleo, por exemplo. Essa modificação faria com que a configuração de cada um dos blocos R, G e B fosse diferente, resultando em uma imagem com pixels coloridos e não em um padrão de cores na escala cinza como o observado com a aplicação de uma única regra no processo de cifragem. Dessa forma, um modelo com maior nível de paralelismo poderia ser obtido. A obtenção de regras distintas para cada canal poderia ser feita a partir de um único núcleo, com rotações distintas, ou mesmo a partir da concatenação de três núcleos diferentes de 64 bits, o que elevaria o tamanho das chaves para 192 bits.

Além disso, testes em objetos e/ou imagens armazenados no formato 3D poderiam ser realizados a fim de verificar o comportamento do modelo 3DHCA na cifragem de arquivos com essas características. Ainda, poderiam ser realizados testes para a verificação do poder de compactação da imagem cifrada gerada pelo algoritmo 3DHCA.

Nos testes realizados nessa dissertação foram utilizados três formatos de imagens: o *ppm* (*Portable Pixmap*), que é um formato simples para imagens a cores, o *pgm* (*Portable Graymap*), que é um formato para imagens na escala cinza e o *pbm* (*Portable Bitmap*) que é um formato para imagens binárias. Todos esses formatos conservam o mesmo tamanho da imagem cifrada e original, pois não há compressão de dados. Por outro lado, ao armazenar ou transmitir dados de forma eficiente, o ideal é reduzir a grande redundância de dados que pode ser alcançada pela compactação dessas imagens. Existem vários formatos de imagem que utilizam essa compactação, como é o caso de imagens *jpeg* e *gif* (compressão com perda de dados) e imagens nos formatos *png* e *tiff* (compressão sem perda de dados). Dessa forma, testes poderiam ser realizados para verificar se as imagens cifradas pelo 3DHCA não possuem redundância de dados, ou seja, a grande parte das imagens apresentadas no Apêndice F, que foram utilizadas nos experimentos, teriam seus tamanhos aumentados após a cifragem, mesmo utilizando os formatos de imagens com compressão de dados.

O estudo baseado na modelagem dos métodos de criptografia como grafos de autômatos finitos com saída deve ser continuado. Esse estudo também poderia contemplar os modelos baseados em grafos para os sistemas criptográficos THCA e 3DHCA, que não foram

modelados nessa dissertação. Além disso, acreditamos que existe a possibilidade de se criar um método baseado no cálculo de pré-imagens para ser empregado em um sistema criptográfico assimétrico, uma vez que foi possível mostrar que o cálculo de pré-imagens pode de ser realizado a partir de uma Máquina de Moore. Conforme foi discutido, a máquina de Moore construída para o cálculo de pré-imagens a partir de uma regra sensível possui um Circuito Hamiltoniano. Sabemos que encontrar esse circuito em um grafo é um problema NP-Completo. Se utilizarmos uma composição de chaves num sistema assimétrico que aborde essa característica importante, uma análise matemática robusta para a segurança do método poderá ser avaliada, assim como foi possível ser avaliada no modelo RSA [Rivest et al., 1978].

Referências Bibliográficas

T. Baignères and S. Vaudenay. Proving the security of aes substitution-permutation network. In *Selected Areas in Cryptography'05*, pages 65–81, 2005.

F. Bao. Cryptoanalysis of a partially known cellular automata cryptosystem. ieee transactions on computers. *IEEE Transactions on Computers*, 11(53):1493 – 1497, 2004.

E. Berlekamp, J. Conway, and R. Guy. Winning ways for your mathematical plays. *Scientific American*, 1992.

S. Blackburn, S. Merphy, and K. Peterson. Comments on theory and applications of cellular automata in criyptography. *IEEE Trans. Computer*, 46(05):637 – 638, 1997.

A. A. Bruen and M. A. Forcinito. *Historical Introduction and the Life and Work of Claude E. Shannon*, pages 1–15. John Wiley e Sons Inc, 2004. ISBN 9781118033296. doi: 10.1002/9781118033296.ch1. URL `http://dx.doi.org/10.1002/9781118033296.ch1`.

L. Chen and J. Lai. Image security system using recursive cellular automata substituition. *Pattern Reconition*, 40(5):1621 – 1631, 2007.

R. Chen, J. Mao, and C. Chui. A symmetric image encryption scheme based on 3d caotic maps. *Fractals, Solitons and Fractals*, 21(3):749 – 761, 2004.

E. Conrad, S. Misenar, and J. Feldman. *Cissp Study Guide*. Newnes, 2012. ISBN 1597499617.

J. H. Conway. Jogo da vida. *Scientific American*, 1970. Jogos matematicos.

J. Daemen and V. Rijmen. Aes proposal: Rijndael, 1998.

S. Donthi and R. Haggard. A survey of dynamically reconfigurable fpga devices. In *Proceedings of 35th Southeastern Symposium*, pages 422 – 426, 2003.

U. Frissh, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equation. *Physical Review Letters*, 56:1505 – 1508, 1988.

N. Ganguly, A. Das, and P. Chaudhuri. Cellular automata model criptosystem. *Cellular Automata Conference*, 2000.

A. Goncalves, A. Martinez, and O. Bruno. Fast, parallel and secure cryptography algorithm using lorenz's attractor. *International Journal of Computer Science and Engineering*, 21(03):365 – 382, 2010.

H. Gutowitz. Cryptography with dynamical systems. *Kluwer Academic Press*, 1995.

M. Habibipour, R. Maarefdoust, M. Yaghobi, and S. Rahati. An image encryption system by 2d memorized cellular automata and chaos mapping. *Digital Content, Multimedia Technology and its Applications (IDC), 2010 6th International Conference on Issue*, (03):331 – 336, 2010.

L. Hernandez, S. Hoya, A. MartaÂn, and G. RodriÂguez. Modelling forest fire spread using hexagonal cellular automata. *Applied Mathematical Modelling*, 31(6):1213 – 1227, 2007. ISSN 0307-904X. doi: 10.1016/j.apm.2006.04.001. URL `http://www.sciencedirect.com/science/article/pii/S0307904X06000916`.

H. Huang, L. Zhang, Y. Guan, and D. Wang. A cellular automata model for population expansion of spartina alterniflora at jiuduansha shoals, shanghai, china. *Estuarine, Coastal and Shelf Science*, 77(1):47 – 55, 2008. ISSN 0272-7714. doi: 10.1016/j.ecss.2007.09.003. URL `http://www.sciencedirect.com/science/article/pii/S0272771407003940`.

J. Jun. Image encryption method based on elementary cellular autoamta. *Southeastcon - IEEE*, pages 345 – 349, 2009.

S. Karmakary and Chowdhury D. R. Nocas : A nonlinear cellular automata based stream cipher. In *AUTOMATA*, pages 135 – 146, 2011.

C. Langton. Artificial life. *Lectures in Complex Systems*, 1992.

W. Li. Parameterizations of cellular automata rule space. *Report: Santa Fe Institute Tech*, 1991.

M. J. L. Lima. Criptografia baseada no calculo generico de pre-imagens de automatos celulares. Master's thesis, Universidade Presbiteriana Mackenzie, 2005.

H. Macedo. Um novo metodo criptografico baseado no calculo de pre-imagens de automatos celulares caoticos, nao-homogeneos e nao-aditivos. Master's thesis, Universidade Federal de Uberlandia, 2007.

M. Machhout, G. Zied, Medien Z., and R. Tourki. Design of reconfigurable image encryption using 2d cellular automata generator. *IJCSA*, 6:43 – 62, 2009.

T. A. J. Magalhaes. Metodo critpgrafico baseado em automatos celulares bidimensionais para cifragem de imagens. Master's thesis, Universidade Federal de Uberlandia, 2010.

F. Maleki, G. Zied, Z. Hashemi, and M. Shiri. Design of reconfigurable image encryption system by cellular automata whith memory. *Availabity, Reliability and Security, International Conference*, 2008.

W. Mao. *Prentice Hall & Hall/CRC*, 2003.

M. Markus and I. Kusch. Cellular automata for modelling the shell pigmentation of molluscs, journal of biological systems. *Journal of Biological Systems*, 3:999 – 1011, 1995.

M. Matsui. Linear cryptoanalysis method for des cipher. *Advances in Cryptography*, pages 386 – 397, 1994.

S. Nandi, B. Kar, and P. Chaudhuri. Theory and applications of cellular automata in cryptography. *IEEE Trans. Computer*, 43:1346 – 1357, 1994.

G. M. B. Oliveira. Automatos celulares: aspectos dinamicos e computacionais. *III Jornada de Mini-cursos em Inteligencia Artificial (MCIA) - Sociedade Brasileira de Computacao*, 8:297 – 345, 2003.

G. M. B. Oliveira and H. Macedo. Sistema criptografico baseado no calculo de pre-imagem em automatos celulares nao-homogeneos, nao-aditivos e com dinamica caotica. *Privilégio de Inovação*, PI0703188-2(INPI 01407000664), 2007.

G. M. B. Oliveira, P. Oliveira, and N. Omar. Definition and applications of a five-parameter characterization of 1d cellular automata rule space. *Artificial Life*, 7(3): 277–301, 2001.

G. M. B. Oliveira, A. Coelho, and L. Monteiro. Cellular automata cryptographic model based on bi-directional toggle rules. *Int. J. of Modern Physics C*, 2004.

G. M. B. Oliveira, M. Lima, H. Macedo, and A. Branquinho. A cryptographic modelo based on the pre-image computation of cellular automata. *Theory and Applications of Cellular Automata*, pages 139 – 155, 2008.

G. M. B. Oliveira, L. Martins, L. Alt, and G. Ferreira. A cellular automata-based cryptographic model with a variable-length ciphertext. *The 2010 International Conference on Scientific Computing*, pages 1 – 10, 2010.

G. G. Powathil, K. E. Gordon, L. A. Hill, and M. A. J. Chaplain. Modelling the effects of cell-cycle heterogeneity on the response of a solid tumour to chemotherapy: Biological insights from a hybrid multiscale cellular automaton model. *Journal of Theoretical Biology*, 308(0):1 – 19, 2012. ISSN 0022-5193. doi: 10.1016/j.jtbi.2012.05.015. URL `http://www.sciencedirect.com/science/article/pii/S0022519312002573`.

J. L. Puliafito. A transport model for the evolution of urban systems. *Applied Mathematical Modelling*, 31(11):2391 – 2411, 2007. ISSN 0307-904X. doi: 10.1016/j.apm.2006.09.005. URL `http://www.sciencedirect.com/science/article/pii/S0307904X06002198`.

R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

Z. Santos, R. Maria, and S. Coutinho. Dynamics of hiv infection: A cellular automata approach. *Phys. Rev. Lett.*, 87:168102, Sep 2001. doi: 10.1103/PhysRevLett.87.168102. URL `http://link.aps.org/doi/10.1103/PhysRevLett.87.168102`.

C. Sarkar and S. A. Abbasi. Enhancing the accuracy of forecasting impact of accidents in chemical process industry by the application of cellular automata technique. *Process Safety and Environmental Protection*, 84(5):355 – 370, 2006. ISSN 0957-5820. doi: 10.1205/psep.04316. URL `http://www.sciencedirect.com/science/article/pii/S0957582006713544`.

S. Sen, C. Shaw, D. Chowdhuro, N. Ganguly, and P. Chaudhuri. Cellular automata based cryptossistem (cac). *Information and Communications Security*, 2513:303 – 314, 2002.

Adi Shamir. A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem. In *23rd Annual Symposium on Foundations of Computer Science, 3-5 November 1982, Chicago, Illinois, USA*, pages 145–152, Chicago, Illinois, 1982. IEEE.

C.E. Shannon. A mathematical theory of communication. Technical Report 27 :379, Bell System technical journal, 1948.

M. Sipser. *Introduction to the Theory of Computation*. Course Technology, 1 edition, December 1996. ISBN 053494728X. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/053494728X`.

D. Socek, S. Li, S. Mangliveras, and B. Furht. Enhanced 1d chaotic key-based algorithm for image encryption. 2005.

William Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson Education, 3rd edition, 2002. ISBN 0130914290.

D. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, 3rd edition, 2006.

K. Sutner. De bruijn graphs and linear cellular automata. 1991.

A Varas, M Cornejo, D Mainemer, B Toledo, J Rogan, V Munoz, and J Valdivia. Cellular automaton model for evacuation process with obstacles. *Physica A: Statistical Mechanics and its Applications*, 382(2):631 – 642, 2007. URL `http://linkinghub.elsevier.com/retrieve/pii/S0378437107003676`.

J. von Neumann and A. Burks. Theory of self-reproducing automata. *University of Illinois Press*, 1966.

S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1 – 35, 1984.

S. Wolfram. *Cellular Automata.* Los Alamos Science., 1986.

S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

Y. Wu, J.P. Noonan, and S. Agaian. Binary data encryption using the sudoku block cipher. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 3915 –3921, oct. 2010. doi: 10.1109/ICSMC.2010.5641801.

A. Wuensche. Complexity in one-d cellular automata: Gliders, basins of attraction and the z parameter. *Cognitive Science Research Papers: CSRP 321.*, 2004.

A. Wuensche. Encryption using cellular automata chain-rules. *Theory and Applications of Cellular Automata*, pages 126 – 136, 2008.

A. Wuensche and M. Lesser. Global dynamics of cellular automata. *Addison-Wesley*, 1992.

L. Yu, Y. Li, and X. Xia. Image encryption algorithm based on self-adaptative symmetrical-coupled toggle cellular automata. *Congress on Image and Signal Processing - IEEE Computer*, 3:32 – 36, 2010. IEEE Computer Society.

J. Zambreno, D. Honbo, and A. Choudary. Exploiting area/delay tradeoffs in an aes fpga implementation. *Lecture Notes in Computer Science*, pages 575 – 585, 2005. Springer Berlin.

M. Zeghid, M. Machhout, L. Khrji, A. Baganne, and R. Tourki. A modified aes based algorithm for image encryption. *International Journal of Computer Science and Modern*, 2007.

Apêndice A

Parâmetro Z

A título de exemplo apresentaremos o cálculo de Z_{left} e Z_{right}. O Z_{left} é calculado verificando se as saídas são sensíveis alterando, primeiramente, o bit mais a esquerda até o bit mais a direita e fazendo os agrupamentos necessários para esse processo. O Z_{right}, é calculado de modo semelhante, mas analisando os bits em ordem inversa (da direita para a esquerda). Tal que Z_{left} pode ser calculado pela Equação A.1, sendo que R representa a saída padronizada do agrupamento analisado.

$$Z_{left} = R_k + \sum_{i=1}^{k-1} R_{k-i} \prod_{j=k-i+1}^{k} (1 - R_j) \tag{A.1}$$

Sabendo que Z_{right} pode ser calculado de maneira análoga e que R_{k-i} pode ser obtido através da Equação A.2. Sabe-se também que D representa o valor de uma saída determinística e ND representa o valor de uma saída não determinística, então D_i é a somatória de todas as vizinhanças determinísticas de uma regra. na Figura A.1

O cálculo de Z_{left} pode ser visualizado na Figura A.1 e é dado por $R_3 + R_2(1 - R_3) + R_1(1 - R_2)(1 - R_3)$, então, $Z_{left} = 0{,}75$. O cálculo de Z_{right} é realizado por um procedimento similar agrupando-se os pares (quádruplas e óctuplas) da direita para a esquerda (Ex: 000 e 100). Ao final tem-se que $Z = \max(Z_{left}, Z_{right})$.

$$R_{k-i} = \frac{n_{k-i}}{2^k} \tag{A.2}$$

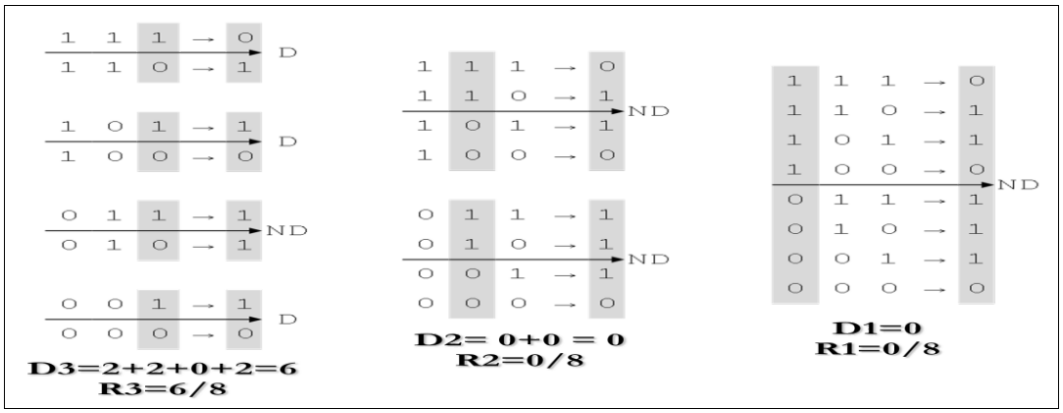


Figura A.1: Cálculo do parâmetro Z da regra 110.

Apêndice B

Criptografia clássica

B.1 Método de deslocamento (*Shift*)

O criptosistema de deslocamento é baseado nas definições teóricas da aritmética modular. Para entender melhor o sistema, devemos considerar a congruência $a \equiv b(\text{mod })$ se m divide $b-a$, ou seja, a é congruente a b módulo m. Devemos definir a aritmética módulo m, como $\mathbb{Z}_m$ que representa o conjunto $\{0, \ldots, m-1\}$ e possui as duas operações soma $(+)$ e multiplicação $(\times)$. Os resultados obtidos por essas operações aritméticas devem ser reduzidos ao módulo de m.

Suponha que se queira computar 11×13 no $\mathbb{Z}_{16}$. Como resultado temos que $11 \times 13 = 143$. Assim devemos reduzir 143 ao módulo 16 e temos que $143 = 8 \times 16 + 15$, então $143 \bmod 16 = 15$, e dessa forma $11 \times 13 = 15$ no $\mathbb{Z}_{16}$. As propriedades matemáticas da aritmética real também são válidas para a aritmética modular.

O Criptosistema de Deslocamanto (*Shift*) pode ser definido formalmente considerando $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$, para $0 \leq k \leq 25$ e temos que:

$$e_k(x) = (x + k) \bmod 26$$

e

$$d_k(y) = (y - k) \bmod 26$$

$(x, y) \in \mathbb{Z}_{26}$

Um sistema famoso, conhecido como *Cifra de Cézar*, foi utilizado por Júlio Cézar para comunicar planos de batalha, tem $k = 3$ como chave criptográfica.

Para ilustrar como um criptosistema de deslocamento funciona, vamos utilizar o exemplo mostrado em [Stinson, 2006], com chave $k = 11$, o texto plano *wewillmeetatmidnight* e o alfabeto ilustrado na Figura B.1. Nesse caso, as letras maiúsculas foram utilizadas apenas para diferenciar o texto claro do texto cifrado.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Figura B.1: Alfabeto português [Stinson, 2006].

Primeiramente, devemos verificar o número correspondente a cada uma das letras da mensagem que está sendo enviada. O decimal correspondente a cada uma dessas letras pode ser ilustrada na Figura B.2 (a). Como foi especificado anteriormente, devemos adicionar o valor da chave $k = 11$ em cada um dos valores da Figura B.2 (a) e aplicar o módulo sobre o valor encontrado, nesse caso, estamos utilizando $\mathbb{Z}_{26}$. Como exemplo, suponha o valor 22 que corresponde à letra w do alfabeto. Para cifrar essa letra devemos computar $(22 + 11) \bmod 26 = 33 \bmod 26 = 7$. Assim, a letra w deverá ser substituída pela letra H. Aplicando a mesma regra para as demais letras do texto plano, obtemos a numeração apresentada na Figura B.2 o seguinte texto cifrado $HPHTWWXPPELEXTOYTRSE$. Para decifrar o processo é inverso, primeiramente, transforma-se cada caractere do texto cifrado em um número, e em seguida deve-se subtrair o valor da chave do valor que corresponde ao caractere cifrado. Para exemplificar, suponha H que equivale ao inteiro 7, deve-se calcular $7 - 11 \bmod 26 = -4 \bmod 26 = 22$ e finalmente, obter o texto plano novamente.

22	4	22	8	11	11	12	4	4	19
0	19	12	8	3	13	8	6	7	19

(a)

7	15	7	19	22	22	23	15	15	4
11	4	23	19	14	24	19	17	18	4

(b)

Figura B.2: (a) Numeração correspondente ao texto plano. (b) Numeração correspondente ao texto cifrado [Stinson, 2006].

Para que um criptosistema seja usado na prática, ele deve satisfazer algumas propriedades, podemos enumerar duas delas:

1. Cada função de criptografia e_k e cada função de descriptografia d_k devem ser eficientemente computáveis.
2. Um criptoanalista (oponente), vendo o texto cifrado y, não pode conseguir determinar o texto plano ou a chave k que foi utilizada para cifrar a string x.

Observe que a criptografia de deslocamento não é segura, pois, é fácil para um criptoanalista fazer uma busca exaustiva no conjunto de chaves. Como existem apenas 26

chaves possíveis, verificar todas as possibilidades não é uma tarefa computacionalmente difícil.

B.2 Método de substituição

A criptografia de substituição é bastante conhecida e foi utilizada por centenas de anos. Esse criptosistema considera $\mathcal{P}$ e $\mathcal{C}$ como sendo as 26 letras do alfabeto. Esse algoritmo de criptografia difere do modelo anterior pelo fato de que nesse consideramos que o processo de cifragem e decifragem é permutação (π) das letras do alfabeto enquanto que no modelo anterior esse processo ocorria através de operações algébricas.

Na Figura B.3, [Stinson, 2006], temos um permutação de letras do alfabeto que podem ser entendidas como uma função de encriptação. Assim, temos que $e_\pi(a) = X$, $e_\pi(b) = N$, etc. A função de descriptação é a permutação inversa, onde $d_\pi(A) = d$, $d_\pi(N) = b$, $d_\pi(X) = a$, etc.

a	b	c	d	e	f	g	h	i	j	k	l	m
X	N	Y	A	H	P	O	G	Z	Q	W	B	T

n	o	p	q	r	s	t	u	v	w	x	y	z
S	F	L	R	C	V	M	U	E	K	J	D	I

Figura B.3: Função de encriptação do sistema de substituição [Stinson, 2006].

A chave de substituição consiste numa permutação dos 26 caracteres do alfabeto. Nesse caso o espaço de chaves corresponde a 26!. Esse grande número de chaves possíveis é inviável até mesmo para um computador. Mesmo essa busca exaustiva de chaves sendo inviável, existe um método intitulado análise da frequência que consegue quebrar esse algoritmo. O ataque explora o fato da linguagem natural possuir um elevado volume de redundância, [Mao, 2003]. Na língua portuguesa, sabe-se que a frequência média da letra "A" é bem maior que a ocorrência da letra "B", como mostra a Figura B.4. Assim, um texto cifrado com alta índice de repetição da letra "W", sugere que W corresponda à letra "A".

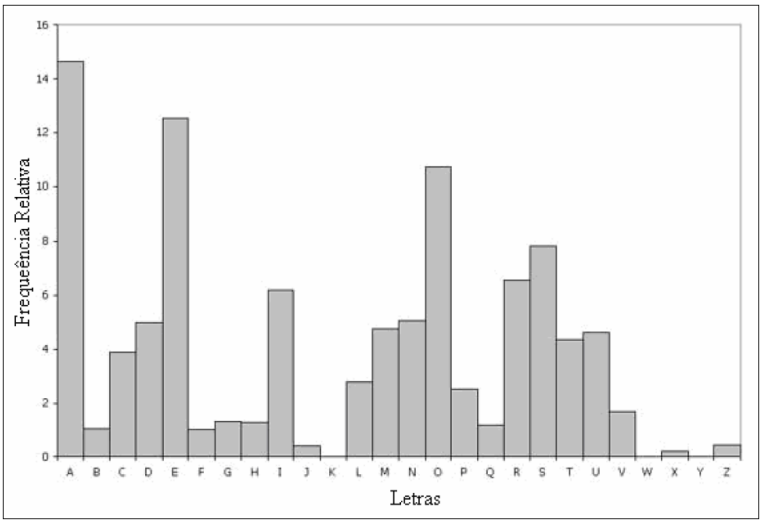


Figura B.4: Frequência do uso de letras na língua portuguesa [Macedo, 2007].

B.3 Método de transposição

O método de transposição também é um método de cifra clássica. Nesse método, uma permutação entre as letras do texto claro é realizada. Como exemplo, suponha o texto claro ($x \in \mathcal{P}$) "RETORNE HOJE PARA BASE UM" este deve ser estruturado em forma de matriz, onde o número de colunas é dado pelo tamanho da chave (k) e são utilizadas tantas linhas quanto forem necessárias para representar o texto claro. Seja a chave "431256", o texto cifrado é dado pela leitura dos caracteres nas colunas, de acordo com a ordenação da chave.

Chave	4	3	1	2	5	6	7
Texto original	R	E	T	O	R	N	E
		H	O	J	E		P
	A	R	A		B	A	S
	E		U	M	X	Y	Z

Figura B.5: Exemplo do método de transposição [Macedo, 2007].

Neste caso o texto cifrado será: "TOAUOJ MEHR R AEREBXN AYESPZ". Segundo [Stallings, 2002], quando a cifragem do texto é realizadas utilizando o método de transposição esta é facilmente identificadas porque o texto claro e o texto cifrado possuem a mesma frequência de letras. Uma criptoanálise para o exemplo de transposição citado acima consistiria em ajustar o texto cifrado em uma matriz e tentar reorganizar o texto claro fazendo uso de pares de letras que ocorrem com frequência, como por exemplo, dígrafos, e então analisar os possíveis anagramas. Quando executadas por mais de um estágio de tranposição este método se torna mais seguro. Assim o texto "TOAUOJ MEHR

R AEREBXN AYESPZ" foi cifrado mais de uma vez usando a mesma técnica e nesse caso utilizou-se chaves diferentes, o que resultaria em uma permutação mais complexa.

As técnicas de substituição e transposição são conhecidas como métodos de criptografia clássica. Esses métodos, se usados sozinhos, possuem vulnerabilidades, mas quando usadas em conjunto com outras técnicas, são ferramentas poderosas na construção de um sistema criptográfico como, por exemplo, o sistema DES (*Data Encryption Standard*) e o sistema AES (*Advanced Encryption Standard*).

Apêndice C

Criptografia moderna

C.1 Criptografia simétrica

C.1.1 Estrutura de Feistel

Os métodos de cifragem por blocos ficaram bastante conhecidos após a Segunda Guerra Mundial. Na Estrutura de Feistel, o bloco de mensagem do texto plano, convertido em bits, é dividido em duas partes (L_1, R_1). Cada uma dessas metades passa por n etapas (*rounds*) e ao final são novamente combinadas produzindo assim, o bloco de texto cifrado. Essas etapas consistem em aplicar uma operação de substituição ou permutação em R_1, juntamente com a função F e uma sub-chave correspondente àquela etapa. A sub-chave é derivada da chave do sistema criptográfico. Posteriormente, aplica-se a operação XOR entre L_1 e a saída da função F. Essa estrutura é ilustrada na Figura C.1. Uma troca entre os bits de R_1 e L_1 é realizada ao final de cada etapa. Os parâmetros do sistema tais como tamanho do bloco, tamanho da chave e número de etapas podem ser alterados conforme a necessidade imposta pelo sistema.

C.1.2 Sistema de criptografia DES

O sistema de criptografia DES (*Data Encryption Standard*) é o sistema mais estudado e mais citado por pesquisadores. Foi desenvolvido na década de 70 pela equipe da IBM. O algoritmo do DES se baseia na estrutura de Feistel utilizando 16 *rounds* e possuir uma permutação ao início e uma ao final do processo de cifragem.

Os blocos de entrada no DES possuem tamanho de 64 bits. Como veremos após uma etapa equivalente a um round na estrutura de Feistel, um bloco será transformado em outro bloco de 64 bits. Para tal, a chave de 56 bits é usada a cada round.

Cada um das 16 etapas gera uma nova sub-chave diferente, de tamanho igual a 48 bits produzida a partir da sub-chave original. Uma permutação dos 56 bits da chave é realizada. Em seguida, os bits da chave são divididos em duas metades de 28 bits cada,

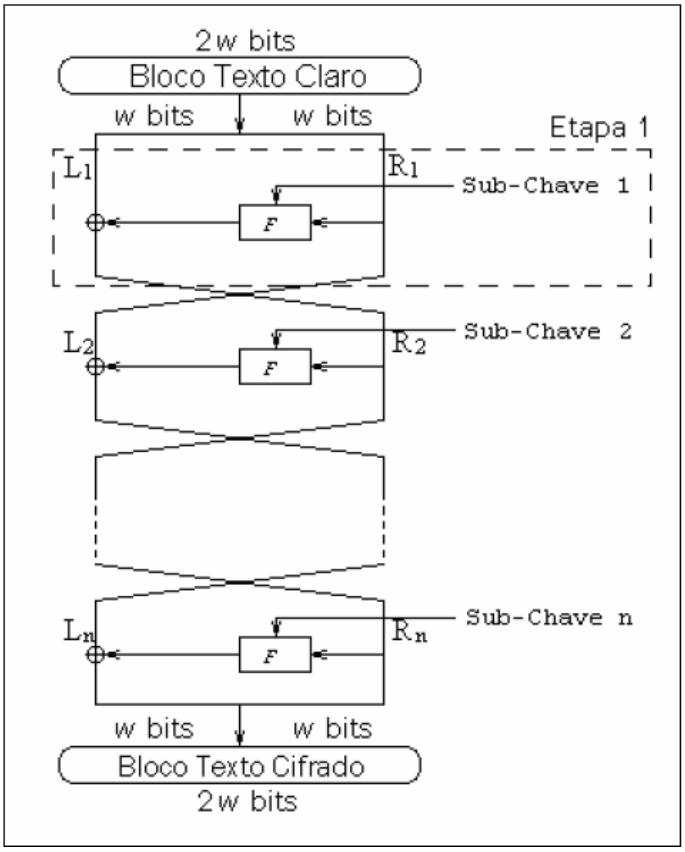


Figura C.1: Estrutura de Feistel [Macedo, 2007].

essas metades são denominadas E_0 e D_0. A cada etapa, E_i e D_i sofrem separadamente um deslocamento circular de duas posições, exceto nas etapas 1, 2, 9 e 16, que deslocam apenas uma posição. Esse deslocamento, serve como entrada para geração da sub-chave do próxima entrada e também para entrada da segunda permutação que produz 48 bits de saída. Esses 48 bits correspondem à sub-chave que também é utilizada para como parâmetro para a função F.

Antes de ser processado pelas 16 etapas, o texto claro passa por uma permutação inicial. Ao final das 16 etapas, troca-se a metade dos bits do lado direito com a metade dos bits do lado esquerdo. Dessa forma, é possível que o mesmo algoritmo seja utilizado tanto no processo de cifragem quanto no processo de decifragem.

O bloco de texto de 64 bits é dividido em duas partes de 32 bits (L,R). Visto que, R possui 32 bits, uma tabela de permutação/expansão é utilizada para gerar 16 bits adicionais, resultando em uma palavra de 48 bits. Os 48 bits sofrem a operação XOR com os 48 bits da sub-chave. Esses bits resultantes são submetidos a 8 funções que são conhecidas como tabelas *S-box*. Essas tabelas *S-box* possuem 6 bits de entrada e 4 bits de saída. A figura C.2 apresenta um *S-box* S_5 utilizada no DES que será explicado a seguir. Assim, os 48 bits resultarão em 32 bits, após ter sido aplicada a tabela *S-box*. Cada um dos 32 bits de saída das tabelas *S-box* passam por uma permutação e então sofrem operação XOR com os bits de L_{i-1}, resultando, assim, os bits de R. Uma etapa do DES

é ilustrada na Figura C.3 e também a estrutura de Feistel utilizada.

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

Figura C.2: Valores definidos pela *S-box 6* (S_6) no DES [Macedo, 2007].

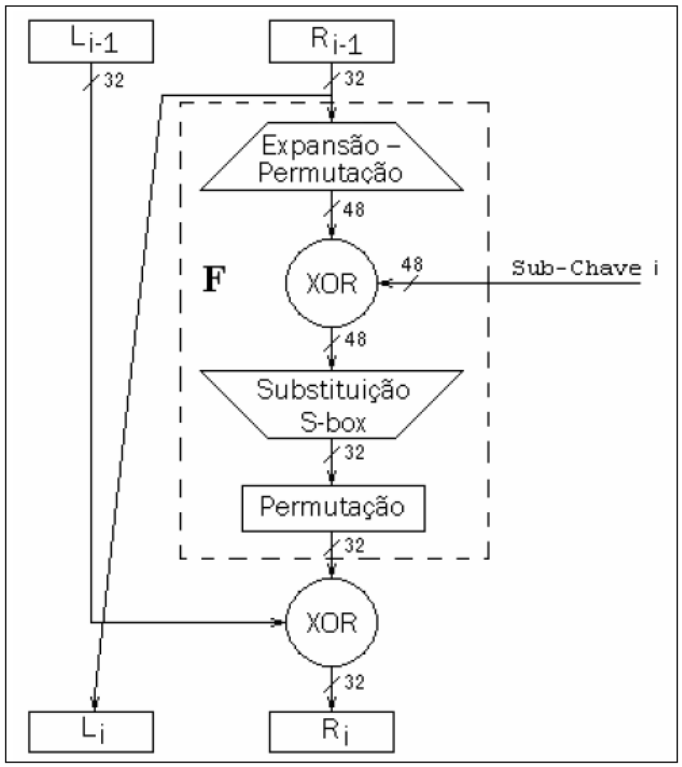


Figura C.3: Representação de uma etapa no DES [Macedo, 2007].

A Tabela S-*box* retorna um valor de saída a partir de uma consulta na linha e coluna desta tabela. Cada uma das 8 tabelas S-*box* possuem 4 linhas e 16 colunas, sendo que, os 4 bits de saída são obtidos a partir de 6 bits de entrada (Figura 6). A linha é definida pelo primeiro e o último bit da entrada (2 bits), que combinados formam um número na base decimal. Dentre os 6 bits de entrada, o valor da coluna é obtido pelos 4 bits do meio convertidos na base decimal. Os valores das tabelas S-*box* podem variar de 0 a 15. Os bits de saída podem variar de 0000 a 1111. Considere o exemplo da Figura C.2 do processo em uma tabela S-*box*, considere os 6 bits de entrada 001001 sendo aplicados à tabela S-*box* (S_6). Os bits da extremidade 01, convertidos para base decimal, correspondem à linha 1 da tabela S-*box*. Os 4 bits do meio, referentes aos 6 bits de entrada, são os bits 0100, que na base decimal representam a coluna 4. Ao consultar a linha 1 e coluna 4 na tabela S-box é obtido o número 15, que convertido para base binária produz os 4 bits de saída 1111. Portanto, a entrada 001001 é convertida na saída 1111, se submetida à tabela S-*box* 6 do DES.

As tabelas de permutação, assim como os valores das S-boxes, podem ser encontrados em [Stinson, 2006].

Atualmente, o DES não é fortemente utilizado devido ao fato de que com o advento de computadores cada vez mais sofisticados, algoritmos de força bruta podem facilmente quebrar um sistema criptográfico com chaves de 56 bits. Novas versões surgiram, dentre elas podemos citar o 2DES e o 3DES que operam com chaves de 112 bits e 168 bits respectivamente.

C.1.3 Sistema de criptografia AES

O sistema criptográfico AES (*Advanced Encryption Standard*) [Daemen and Rijmen, 1998] trabalha com blocos de tamanho 128 bits e utiliza chave de 128, 192 e 256 bits. O AES faz uso de uma estrutura conhecida como rede de substituição e permutação (SPN) [Baignères and Vaudenay, 2005]. Ao contrário do seu predecessor DES que utiliza uma rede de Feistel, uma SPN utiliza etapas (*rounds*) contendo caixas de substituição conhecidas como S-*box* e caixas de permutação conhecidas como P-*box*, onde cada etapa geralmente é combinada com a chave através de alguma operação de grupo, como por exemplo, a operação XOR.

O número de etapas executadas depende do tamanho da chave. Para chave de 128, 196 ou 256 bits, são executados 10, 12 e 14 etapas, respectivamente. O bloco de texto é formado por 128 bits que será copiado para uma matriz. Existem quatro funções principais no AES que operam sobre essa matriz $S_{4\times4}$.

C.2 Criptografia assimétrica

C.2.1 O Criptosistema RSA

O primeiro algoritmo de criptografia de chave pública foi criado por Diffie and Hellman em 1976. Posteriormente, o algoritmo de criptografia RSA foi criado por [Rivest et al., 1978]. Se por um lado a função de criptografia e_k do RSA é facilmente calculada, por outro, a função de descriptografia (função inversa) é de difícil computabilidade. Nessa seção apresentaremos novamente a definição do RSA associada a um exemplo prático e ao algoritmo responsável pela geração de parâmetros.

O criptosistema RSA pode ser descrito utilizando computações em $\mathbb{Z}_n$, onde n é o produto de dois primos ímpares p e q. Para esse inteiro n temos a função tociente de Euler, representada por ϕ, é definida para um número natural n como sendo igual à quantidade de números menores que n co-primos com respeito a ele. Assim, temos que, $\phi(n) = (p-1)(q-1)$. O criptosistema é descrito a seguir:

Seja $n = p \times q$, onde p e q são primos. Seja $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$, define-se:

$$\mathcal{K} = (n, p, q, a, b)\text{: } ab \equiv (1 \bmod \phi(n)) \text{ e } \gcd(b, \phi(n)) = 1.$$

ou seja, devem ser escolhidos dois inteiros a e b de forma que b e $\phi(n)$ sejam co-primos, isto é, o máximo divisor comum entre b e $\phi(n)$ deve ser igual a 1.

Um exemplo utilizado em [Stinson, 2006], será mostrado a seguir para ilustrar como o algoritmo RSA funciona.

Suponha que Bob escolha $p = 101$ e $q = 113$. Assim, temos que $n = 11413$ e $\phi(n) = 100 \times 112 = 11200$. Como $11200 = 2^6 5^2 7$, um inteiro b deve ser escolhido, de tal forma que não deve ser divisível por $2, 5, 7$, ou seja, Bob deve verificar se $\gcd(\phi(n), b) = 1$. Ao mesmo tempo é computado b^{-1}, suponha que Bob escolha $b = 3533$. Então,

$$b^{-1} \bmod 11200 = 6597$$

Dessa forma, temos que o valor de a é dado por: $a = 6597$.

Uma vez definidos os parâmetros n, p, q, a, b que definem a chave $k = (n, p, q, a, b)$, o processo de cifragem e_k e de decifragem d_k é dado por:

$$e_K(x) = x^b \bmod n$$

e

$$d_K(y) = y^a \bmod n$$

De tal forma que $(x, y \in \mathbb{Z}_n)$. Os valores n e b compreendem a chave pública e os valores p, q e a formam a chave privada. O Algoritmo 4 define o esquema de geração dessas chaves.

Algoritmo 4 Gerador dos parâmetros do RSA

1. Gere dois primos grandes p e q, de tal forma que $p \neq q$.
2. $n \leftarrow pq$ e $\phi(n) \leftarrow (p-1)(q-1)$
3. Escolha b randômico, $1 < b < \phi(n)$ tal que $\gcd(b, \phi(n)) = 1$
4. $a \leftarrow b^{-1} \bmod \phi(n)$
5. A chave pública é (n, b) e a chave privada é (p, q, a).

Voltando ao exemplo anterior, vejamos como ficaria a cifragem de uma mensagem enviada de Alice para Bob. Bob publica $n = 11413$ e $b = 3533$ em um diretório. Suponha que Alice quer encriptar o texto plano 9726 para enviar a Bob. Ela deve computar:

$$9726^{3533} \bmod 11413 = 5761$$

e envia o o texto cifrado 5761 ao longo de um canal. Quando Bob receber o texto cifrado 5761, ele deve usar seu expoente secreto para computar o texto plano:

$$5761^{6597} \bmod 11413 = 9726$$

C.2.2 Considerações sobre criptosistemas de chave pública

Mostrar a segurança de modelos criptográficos é uma tarefa muito importante ao se modelar um criptosistema. A redução é uma ferramenta que permite provar a segurança de criptosistemas. Entretanto, muitos criptosistemas não permitem uma forma elegante de provar sua segurança se baseando nas noções matemáticas do problema da redução.

Em criptosistemas de chave pública, como por exemplo o RSA, cada indivíduo somente precisa estabelecer um único par de chaves, a de encriptação b e a de decriptação a. O individuo mantem a secreta, mas publica b. Se um outro indivíduo deseja enviá-lo uma mensagem, este deve buscar b no diretório público, encriptar a mensagem utilizando b, e a envia para o primeiro indivíduo. O primeiro indivíduo é o único que conhece a, portanto somente ele pode decriptar aquela mensagem.

Para mostrar a segurança do RSA precisaremos dos conceitos sobre funções unidirecionais e funções de alçapão. A existência da primeira permite a construção de sistemas de chave-privada seguros, enquanto que, a existência da segunda, permite construir criptosistemas de chave-pública.

Dizemos que uma função é unidirecional (*one-way*) se ela é fácil de computar mas quase sempre difícil de inverter. Para isso ela deve atender algumas restrições, por exemplo, deve ser uma função comprimento-preservante, onde a cadeia de entrada e saída possuem o mesmo comprimento. Uma função comprimento-preservante é uma permutação se ela nunca mapeia duas cadeias para o mesmo lugar. Para isso, devemos supor que existe um algoritmo polinomial probabilístico que tenta inverter a função unidirecional f. Um algoritmo polinomial determinístico é aquele que dá à máquina duas possibilidades a cada passo de tempo, aceitação ou não aceitação de uma string qualquer, com uma probabilidade de erro. Para funções unidirecionais, para qualquer algoritmo de tempo polinomial probabilístico, é improvável que ele seja capaz de encontrar qualquer saída que mapeia a sua entrada.

A definição da função unidirecional não é suficiente para permitir a construção de um criptosistema assimétrico. A construção pode ser dada a partir da função de alçapão, essa função é facilmente invertida na presença de uma informação especial, que em sistema de criptografia assimétrica, essa informação especial é a chave privada. A segurança no RSA é baseada no fato de que função de cifragem é uma função de *one-way*. Essas funções são largamente conhecidas por serem computacionalmente inviáveis para um oponente decifrar o texto plano que é enviado. De tal forma que se um oponente conhecer a fatoração $n = pq$, então ele conhece $\phi(n) = (p-1)(q-1)$ e consequentemente, vai computar facilmente o expoente de decriptação a utilizando o $\gcd(\phi(n), b)$.

Apêndice D

Máquina de Moore

Nesse apêndice um algoritmo que foi construído para gerar as máquinas de Moore do Capítulo 5 será apresentado. O Algoritmo 5 apresenta os passos para a criação de uma Máquina de Moore dada uma regra de transição para a execução *backward*. Utilizando-se o Algoritmo 5 e a Tabela Padrão D.1, as tabelas de transição para as regras 01111000 (sensível à esquerda) e 10011010 (sensível à direita) foram construídas e estão indicadas na Tabela D.2. A tabela padrão de uma regra sensível à direita é aquela em que o bit de saída e o bit que representa a vizinhança da sensitividade possuem o mesmo valor. A regra padrão para a sensitividade à esquerda é a regra 240. A regra pedrão da sensitividade à direita é a regra 170. Assim é fácil construir uma tabela que indica a transição das duas regras (170 e 240). Construindo as tabelas para qualquer regra a partir do algoritmo mostrado e das tabelas padrão, então a construção Máquina de Moore se torna simples. Dois exemplos dessa construção foram mostrados na Figura 5.10 (a) e 5.10 (b).

Algoritmo 5 Máquina de Moore que realiza o cálculo de pré-imagem a partir de uma regra qualquer.

```
1.  Se "regra possui sensitividade à esquerda":
2.      Variável: i = 0
3.      Array: v[tamanho da regra]
4.      Para cada "vizinhança da regra" faça leitura da esquerda para direita:
5.          "concatene: bit de saída da regra + todos os bits da vizinhança exceto o bit sensível."
6.          v[i] = Tabela 1(a)["transforme a string concatenada no binário correspondente"]
7.          i = i + 1
8.  Se "regra possui sensitividade à direita":
9.      Variável: i = 0
10.     Array: v[tamanho da regra]
11.     Para cada "vizinhança da regra" faça leitura da esquerda para direita:
12.         "concatene: todos os bits da vizinhança exceto o bit sensível + bit de saída da regra"
13.         v[i] = Tabela 1(b)["transforme a string concatenada no binário correspondente"]
14.         i = i + 1
```

Tabela D.1: (a) Tabela padrão para regras com sensitividade à esquerda. (b) Tabela padrão para regras com sesitividade à direita.

State	0	1	2	3	4	5	6	7	Rule
Read 0	Go to 0	Go to 0	Go to 1	Go to 1	Go to 2	Go to 2	Go to 3	Go to 3	240
Read 1	Go to 4	Go to 4	Go to 5	Go to 5	Go to 6	Go to 6	Go to 7	Go to 7	(a)
Read 0	Go to 0	Go to 2	Go to 4	Go to 6	Go to 0	Go to 2	Go to 4	Go to 6	170
Read 1	Go to 1	Go to 3	Go to 5	Go to 7	Go to 1	Go to 3	Go to 5	Go to 7	(b)

Tabela D.2: (a) Construção da tabela de movimentos para uma regra com sensitividade à esquerda a partir da tabela padrão. (b) Construção da tabela de movimentos para uma regra com sensitividade à direita a partir da tabela padrão.

State	0	1	2	3	4	5	6	7	Rule
Read 0	Go to 0	Go to 2	Go to 3	Go to 3	Go to 2	Go to 0	Go to 1	Go to 1	30
Read 1	Go to 4	Go to 6	Go to 7	Go to 7	Go to 6	Go to 4	Go to 5	Go to 5	(a)
Read 0	Go to 2	Go to 0	Go to 4	Go to 6	Go to 2	Go to 0	Go to 6	Go to 4	89
Read 1	Go to 3	Go to 1	Go to 5	Go to 7	Go to 3	Go to 1	Go to 7	Go to 5	(b)

Apêndice E

Entropia

Nesse apêndice, será apresentado a forma de calcular a entropia nos modelos unidimensional e bidimensional. Primeiramente, um exemplo do modelo unidimensional utilizado em [Macedo, 2007] será apresentado. Como exemplo de cálculo, suponha a palavra binária 0100101110011101 de 16 bits. Nesse caso, temos j = log2 16 = 4. A Figura E.1 apresenta a forma periódica de se obter as 16 janelas que devem ser analisadas para verificar o número de ocorrências de cada tipo de sequência de 4 bits possível. A partir da frequência de cada uma das janelas indicadas na tabela E, podemos calcular S conforme detalhado abaixo.

$$S = -\left\{ \begin{array}{llll} \left(\frac{1}{16}\log_2\frac{1}{16}\right)+ & \left(\frac{1}{16}\log_2\frac{1}{16}\right)+ & \left(\frac{1}{16}\log_2\frac{1}{16}\right)+ & \left(\frac{2}{16}\log_2\frac{2}{16}\right)+ \\ \left(\frac{2}{16}\log_2\frac{2}{16}\right)+ & \left(\frac{2}{16}\log_2\frac{2}{16}\right)+ & \left(\frac{2}{16}\log_2\frac{2}{16}\right)+ & \left(\frac{1}{16}\log_2\frac{1}{16}\right)+ \\ \left(\frac{1}{16}\log_2\frac{1}{16}\right)+ & \left(\frac{1}{16}\log_2\frac{1}{16}\right)+ & \left(\frac{2}{16}\log_2\frac{2}{16}\right) & \end{array} \right\} = 3.375$$

Normalizando o valor da entropia (entre 0 e 1), deve-se dividir o valor obtido pelo tamanho da janela ($j = 4$). Assim, temos para a sequência {0100101110011101}, o valor da entropia normalizada (s) é igual a 0.84375.

Na Tabela E podemos observar algumas sequências binárias lineares e suas respectivas

Tabela E.1: Ocorrências das janelas.

Janela	Ocorrências	Janela	Ocorrências
0000	0	1000	0
0001	0	1001	2
0010	1	1010	2
0011	1	1011	1
0100	1	1100	1
0101	2	1101	1
0110	0	1110	2
0111	2	1111	0

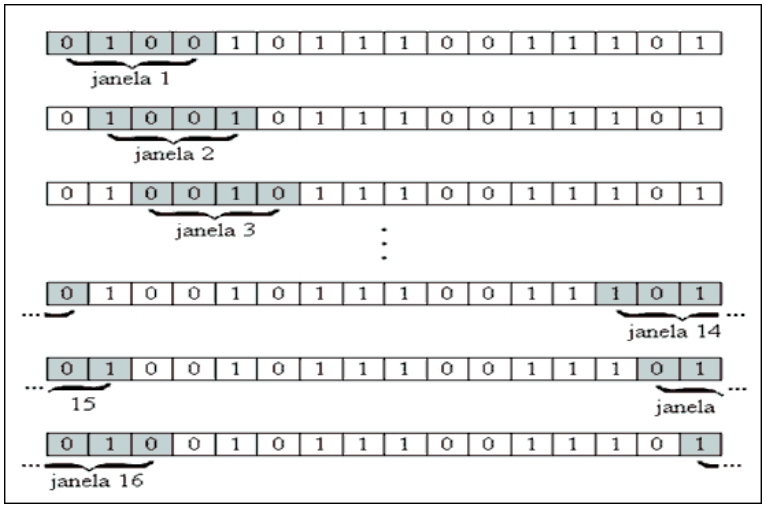


Figura E.1: Janelas da sequência binária {0100101110011101}.

entropias.

Tabela E.2: Exemplos de entropias [Macedo, 2007].

Sequência Binária	S	s
0101010101010101	1.000000	0.250000
0000000100000000	1.311278	0.327820
0110000000000000	1.621641	0.405410
0011001100110011	2.000000	0.500000
0111010101000011	3.452820	0.863205
1101001011000101	3.625000	0.906250
0111110101001000	3.625000	0.906250
1010110000100111	4.000000	1.000000

Como exemplo, para um reticulado bidimensional de $16 \times 16 = 256$, o número de bits da janela deve ser igual a $j = \log_2 256 = 8$. Portanto, existem duas configurações de janelas bidimensionais possíveis, a primeira é utilizar 4×2 e a segunda é utilizar 2×4. Todas as configurações de janelas 2×4 são apresentadas na Figura E.2. A construção das janelas do modelo bidimensional pode ser observada na Figura E.3. O cálculo da entropia é similar ao proposto para o caso unidimensional, considerando-se o número de ocorrências de cada janela bidimensional possível

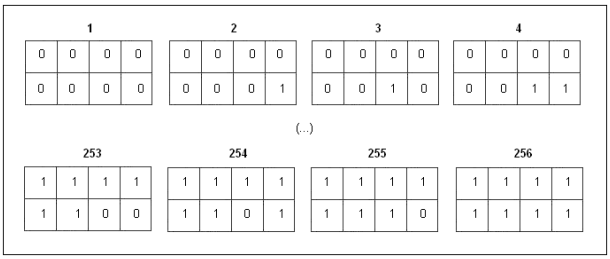


Figura E.2: Janelas 2×4 possíveis.

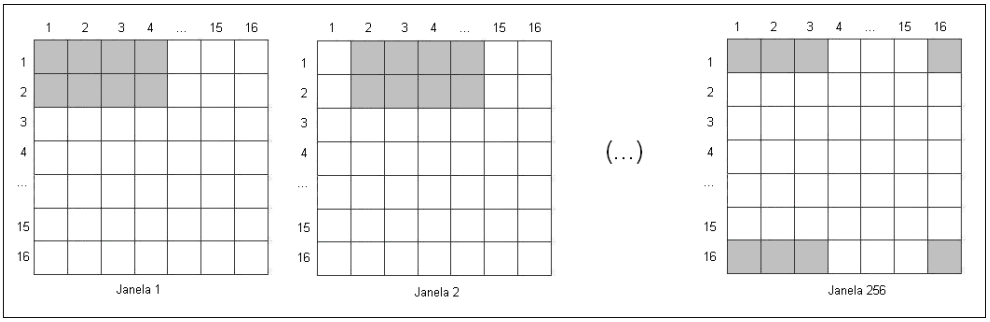


Figura E.3: Construção das janelas de um reticulado de ordem 16×16.

Apêndice F

Banco de imagens

Apêndice G

Resultados obtidos a partir do banco de imagens

G.1 Conjunto dos 10 núcleos

Tabela G.1: Conjunto dos dez núcleos utilizados para os experimentos e suas respectivas entropias.

Índice	Núcleo	Entropia
0	10111100110101100000101100011110001110101111010010101001000111101	0.895144
1	01011001110111110110001111000110001001111100010001011000110100 11	0.852528
2	01001000100011010111100011110000000110000001011000110101010011111	0.872345
3	11001111010000100111100110000001001100101101010001111110010001 10	0.904284
4	11000001110000000000011110110101001101011111111111110101111101 01	0.776629
5	11010101001111000010100011001110010100011000101010000101001010 00	0.810172
6	11111101011110100000010010101010000110101010010110110111100101 10	0.859962
7	00011101111101100011101001011110011111100011101010100100111011	0.833921
8	10110100000101111001101110011100111001001111010111101010101011 00	0.852788
9	10010111111101010010001100010011100111101011110001001111110000 11	0.876277

G.2 Algoritmo para geração dos gráficos

Algoritmo 6 Gráficos construídos no Matlab para exibição dos desvios.

1. **load** nome_do_arquivo.txt;
2. coluna = **sort**(nome_do_arquivo(:, índice_coluna));
3. desvio = 0.5− **abs**(0.5− coluna);
4. **axes**('FontSize', 20);
5. **plot**(desvio, 'r-', 'LineWidth', 4);
6. **xlabel**('Quantidade de Execuções');
7. **ylabel**('Desvio');

G.3 Experimentos para a escolha das imagens mais difíceis de cifrar

Tabela G.2: Especificação detalhada das colunas da Tabela G.3

Label	Descrição
A	Índice da imagem.
B	Média da porcentagem de zeros da imagem.
C	Desvio simples da média em relação ao valor esperado de 0,50.
D	Desvio padrão das 200 médias por regra de cada imagem em relação à média global 7.14.
E	Desvio padrão das 200 médias por regra de cada imagem em relação à média local (Coluna B)
F	Média da entropia do canal R por imagem.
G	Média da entropia do canal G por imagem.
H	Média da entropia do canal B por imagem.

Tabela G.3: Resultados da cifragem das 200 regras nas 200 imagens

A	B	C	D	E	F	G	H
0	0.529327	0.029327	0.000080	0.002490	0.924932	0.924500	0.924534
1	0.529240	0.029240	0.000086	0.002407	0.925186	0.924821	0.924841
2	0.530252	0.030252	0.000014	0.002527	0.923125	0.923038	0.922986
3	0.530478	0.030478	0.000002	0.002776	0.922605	0.922612	0.922544
4	0.530552	0.030552	0.000007	0.002492	0.922643	0.922617	0.922662
5	0.530560	0.030560	0.000008	0.002677	0.922538	0.922455	0.922411
6	0.530521	0.030521	0.000005	0.002596	0.922722	0.922618	0.922482
7	0.530506	0.030506	0.000004	0.002446	0.923038	0.922985	0.922858
8	0.530678	0.030678	0.000016	0.002424	0.922910	0.922791	0.922672
9	0.530398	0.030398	0.000004	0.002570	0.922922	0.922777	0.922663
10	0.530537	0.030537	0.000006	0.002641	0.922853	0.922841	0.922624
11	0.530429	0.030429	0.000001	0.002424	0.922805	0.922709	0.922613
12	0.530496	0.030496	0.000003	0.002641	0.922608	0.922453	0.922497
13	0.530455	0.030455	0.000000	0.002692	0.922988	0.922810	0.922771
14	0.530266	0.030266	0.000013	0.002690	0.923223	0.922974	0.922850
15	0.530547	0.030547	0.000007	0.002598	0.922725	0.922628	0.922556
16	0.530522	0.030522	0.000005	0.002568	0.922455	0.922457	0.922376
17	0.530555	0.030555	0.000008	0.002528	0.922743	0.922608	0.922462
18	0.530738	0.030738	0.000020	0.002728	0.922641	0.922394	0.922381
19	0.530503	0.030503	0.000004	0.002937	0.922728	0.922602	0.922477
20	0.530273	0.030273	0.000012	0.002560	0.923162	0.922958	0.922881
21	0.530651	0.030651	0.000014	0.002806	0.922417	0.922394	0.922265
22	0.530560	0.030560	0.000008	0.002579	0.922513	0.922337	0.922367
23	0.530380	0.030380	0.000005	0.002686	0.922830	0.922700	0.922687
24	0.530254	0.030254	0.000014	0.002862	0.922601	0.922444	0.922489
25	0.530691	0.030691	0.000017	0.002658	0.922675	0.922528	0.922455
26	0.530664	0.030664	0.000015	0.002622	0.922603	0.922514	0.922529
27	0.530424	0.030424	0.000002	0.002572	0.922826	0.922620	0.922560
28	0.530372	0.030372	0.000005	0.002611	0.922751	0.922806	0.922688
29	0.530732	0.030732	0.000020	0.002891	0.922627	0.922389	0.922514
30	0.530114	0.030114	0.000024	0.002952	0.923336	0.923292	0.923142
31	0.530532	0.030532	0.000006	0.002521	0.923055	0.922984	0.922855
32	0.530270	0.030270	0.000013	0.002665	0.923150	0.923069	0.922908
33	0.530315	0.030315	0.000010	0.002626	0.923141	0.923093	0.923002
34	0.530400	0.030400	0.000003	0.002554	0.923068	0.922929	0.922925
35	0.530366	0.030366	0.000006	0.002523	0.923160	0.923072	0.923065

36	0.530185	0.030185	0.000019	0.002541	0.923353	0.923113	0.923020
37	0.530347	0.030347	0.000007	0.002555	0.923283	0.923147	0.923086
38	0.530522	0.030522	0.000005	0.002396	0.923117	0.922984	0.922908
39	0.530243	0.030243	0.000015	0.002898	0.923172	0.923039	0.922994
40	0.530184	0.030184	0.000019	0.002590	0.923279	0.923286	0.923052
41	0.530023	0.030023	0.000030	0.002299	0.923527	0.923417	0.923326
42	0.530001	0.030001	0.000032	0.002662	0.923505	0.923478	0.923295
43	0.530104	0.030104	0.000024	0.002497	0.923552	0.923510	0.923376
44	0.530326	0.030326	0.000009	0.002679	0.923244	0.923135	0.922945
45	0.530112	0.030112	0.000024	0.002628	0.923252	0.923185	0.923122
46	0.530379	0.030379	0.000005	0.002621	0.923311	0.923251	0.923075
47	0.530542	0.030542	0.000007	0.002633	0.922718	0.922552	0.922569
48	0.530632	0.030632	0.000013	0.002951	0.922466	0.922462	0.922361
49	0.530796	0.030796	0.000025	0.002427	0.922578	0.922500	0.922365
50	0.530168	0.030168	0.000020	0.002434	0.923087	0.922935	0.922865
51	0.530631	0.030631	0.000013	0.002812	0.922659	0.922473	0.922465
52	0.530393	0.030393	0.000004	0.002850	0.922942	0.922921	0.922783
53	0.530981	0.030981	0.000038	0.002294	0.922391	0.922325	0.922186
54	0.530294	0.030294	0.000011	0.002636	0.923429	0.923314	0.923228
55	0.530403	0.030403	0.000003	0.002783	0.922843	0.922820	0.922688
56	0.530555	0.030555	0.000008	0.002575	0.922577	0.922573	0.922496
57	0.530414	0.030414	0.000002	0.002539	0.922942	0.922843	0.922693
58	0.530578	0.030578	0.000009	0.002466	0.922670	0.922632	0.922341
59	0.530536	0.030536	0.000006	0.002691	0.922584	0.922500	0.922400
60	0.530566	0.030566	0.000008	0.002660	0.922739	0.922535	0.922519
61	0.530392	0.030392	0.000004	0.002639	0.922773	0.922679	0.922644
62	0.530553	0.030553	0.000007	0.002898	0.922500	0.922498	0.922434
63	0.530219	0.030219	0.000016	0.002388	0.923322	0.923190	0.923227
64	0.530344	0.030344	0.000007	0.002655	0.922986	0.922973	0.922775
65	0.530508	0.030508	0.000004	0.002710	0.922963	0.922859	0.922857
66	0.530573	0.030573	0.000009	0.002759	0.922963	0.922956	0.922876
67	0.530470	0.030470	0.000001	0.002526	0.922811	0.922747	0.922615
68	0.530142	0.030142	0.000022	0.002440	0.923041	0.923016	0.922954
69	0.530313	0.030313	0.000010	0.002513	0.923214	0.923126	0.923030
70	0.530313	0.030313	0.000010	0.002539	0.923015	0.922903	0.922829
71	0.530181	0.030181	0.000019	0.002612	0.923180	0.923261	0.923141
72	0.530623	0.030623	0.000012	0.002922	0.922751	0.922634	0.922619
73	0.530382	0.030382	0.000005	0.002993	0.922911	0.922855	0.922807

74	0.530057	0.030057	0.000028	0.002719	0.923224	0.923057	0.923137
75	0.530405	0.030405	0.000003	0.002732	0.922804	0.922747	0.922711
76	0.530521	0.030521	0.000005	0.002900	0.923038	0.922969	0.922964
77	0.530393	0.030393	0.000004	0.002839	0.922767	0.922801	0.922661
78	0.530337	0.030337	0.000008	0.002679	0.923422	0.923266	0.923252
79	0.530173	0.030173	0.000020	0.002535	0.923183	0.923105	0.923088
80	0.530549	0.030549	0.000007	0.002758	0.922449	0.922363	0.922243
81	0.530663	0.030663	0.000015	0.002800	0.922528	0.922393	0.922333
82	0.530465	0.030465	0.000001	0.002497	0.922872	0.922726	0.922637
83	0.530528	0.030528	0.000006	0.002638	0.922650	0.922634	0.922601
84	0.530384	0.030384	0.000005	0.002512	0.923045	0.923079	0.922877
85	0.530495	0.030495	0.000003	0.002752	0.922793	0.922697	0.922610
86	0.530689	0.030689	0.000017	0.002682	0.922673	0.922651	0.922404
87	0.530659	0.030659	0.000015	0.003023	0.922572	0.922609	0.922436
88	0.530627	0.030627	0.000013	0.002838	0.922700	0.922559	0.922523
89	0.530576	0.030576	0.000009	0.002606	0.922787	0.922747	0.922665
90	0.530398	0.030398	0.000004	0.002458	0.922972	0.922888	0.922718
91	0.530329	0.030329	0.000009	0.002427	0.923129	0.922968	0.922969
92	0.530499	0.030499	0.000004	0.002620	0.922628	0.922564	0.922388
93	0.530037	0.030037	0.000029	0.002587	0.923199	0.923041	0.923149
94	0.530409	0.030409	0.000003	0.002318	0.922802	0.922804	0.922599
95	0.530329	0.030329	0.000009	0.002427	0.923129	0.922968	0.922969
96	0.530247	0.030247	0.000014	0.002688	0.923037	0.922917	0.922742
97	0.530654	0.030654	0.000015	0.002961	0.922432	0.922374	0.922325
98	0.530220	0.030220	0.000016	0.002582	0.922989	0.922895	0.922837
99	0.530381	0.030381	0.000005	0.002693	0.922814	0.922740	0.922540
100	0.530247	0.030247	0.000014	0.002688	0.923037	0.922917	0.922742
101	0.530692	0.030692	0.000017	0.002662	0.922481	0.922532	0.922220
102	0.530004	0.030004	0.000032	0.002612	0.923351	0.923120	0.922985
103	0.530131	0.030131	0.000023	0.002602	0.922672	0.922631	0.922582
104	0.530514	0.030514	0.000005	0.002587	0.922658	0.922518	0.922628
105	0.530515	0.030515	0.000005	0.002652	0.922657	0.922518	0.922442
106	0.530727	0.030727	0.000020	0.002706	0.922350	0.922252	0.922304
107	0.530453	0.030453	0.000000	0.003091	0.922822	0.922591	0.922597
108	0.530449	0.030449	0.000000	0.002699	0.922634	0.922416	0.922435
109	0.530593	0.030593	0.000010	0.002676	0.922707	0.922544	0.922498
110	0.530447	0.030447	0.000000	0.002572	0.922689	0.922560	0.922555
111	0.530593	0.030593	0.000010	0.002676	0.922707	0.922544	0.922498

112	0.530399	0.030399	0.000004	0.002710	0.922683	0.922636	0.922508
113	0.530418	0.030418	0.000002	0.002528	0.922691	0.922604	0.922541
114	0.530486	0.030486	0.000003	0.002550	0.922621	0.922553	0.922508
115	0.530701	0.030701	0.000018	0.002669	0.922628	0.922471	0.922441
116	0.530570	0.030570	0.000009	0.002566	0.922438	0.922516	0.922392
117	0.530418	0.030418	0.000002	0.002528	0.922691	0.922604	0.922541
118	0.530465	0.030465	0.000001	0.002513	0.922701	0.922565	0.922593
119	0.530500	0.030500	0.000004	0.002368	0.922710	0.922589	0.922576
120	0.530475	0.030475	0.000002	0.002492	0.922728	0.922536	0.922478
121	0.530611	0.030611	0.000011	0.002731	0.922677	0.922598	0.922444
122	0.530789	0.030789	0.000024	0.002688	0.922385	0.922442	0.922241
123	0.530479	0.030479	0.000002	0.002731	0.922780	0.922633	0.922576
124	0.530506	0.030506	0.000004	0.002749	0.922720	0.922568	0.922519
125	0.530494	0.030494	0.000003	0.002742	0.922584	0.922557	0.922397
126	0.530551	0.030551	0.000007	0.002415	0.922686	0.922582	0.922569
127	0.530359	0.030359	0.000006	0.002390	0.922853	0.922632	0.922681
128	0.530276	0.030276	0.000012	0.002593	0.922755	0.922756	0.922651
129	0.530319	0.030319	0.000009	0.002827	0.922790	0.922595	0.922493
130	0.530448	0.030448	0.000000	0.002502	0.922969	0.922819	0.922736
131	0.530583	0.030583	0.000009	0.002741	0.922464	0.922340	0.922300
132	0.530454	0.030454	0.000000	0.002699	0.922889	0.922847	0.922617
133	0.530564	0.030564	0.000008	0.002609	0.922740	0.922609	0.922570
134	0.530292	0.030292	0.000011	0.002544	0.922903	0.922735	0.922781
135	0.530510	0.030510	0.000004	0.002567	0.922888	0.922723	0.922712
136	0.530347	0.030347	0.000007	0.002542	0.922763	0.922645	0.922655
137	0.530757	0.030757	0.000022	0.002603	0.922644	0.922437	0.922476
138	0.530605	0.030605	0.000011	0.002464	0.922710	0.922690	0.922566
139	0.530595	0.030595	0.000010	0.002666	0.922634	0.922563	0.922492
140	0.530305	0.030305	0.000010	0.002919	0.923181	0.923036	0.922960
141	0.530412	0.030412	0.000003	0.002542	0.922787	0.922634	0.922547
142	0.530661	0.030661	0.000015	0.002769	0.922439	0.922389	0.922339
143	0.530563	0.030563	0.000008	0.002633	0.922699	0.922741	0.922619
144	0.530581	0.030581	0.000009	0.002615	0.922610	0.922525	0.922455
145	0.530362	0.030362	0.000006	0.002708	0.922986	0.922906	0.922815
146	0.530485	0.030485	0.000003	0.002876	0.922537	0.922450	0.922447
147	0.530757	0.030757	0.000022	0.002870	0.922546	0.922418	0.922409
148	0.530265	0.030265	0.000013	0.002482	0.922973	0.922881	0.922789
149	0.530569	0.030569	0.000008	0.002553	0.922667	0.922432	0.922553

150	0.530624	0.030624	0.000012	0.002738	0.922489	0.922426	0.922436
151	0.530562	0.030562	0.000008	0.002573	0.922862	0.922844	0.922693
152	0.530790	0.030790	0.000024	0.002623	0.922553	0.922471	0.922396
153	0.530437	0.030437	0.000001	0.002628	0.922525	0.922555	0.922486
154	0.530503	0.030503	0.000004	0.002589	0.922673	0.922724	0.922550
155	0.530592	0.030592	0.000010	0.002803	0.922602	0.922465	0.922411
156	0.529915	0.029915	0.000038	0.002451	0.923580	0.923450	0.923452
157	0.530539	0.030539	0.000006	0.002632	0.922670	0.922607	0.922553
158	0.530424	0.030424	0.000002	0.002575	0.922768	0.922628	0.922537
159	0.530700	0.030700	0.000018	0.002588	0.922836	0.922676	0.922571
160	0.530604	0.030604	0.000011	0.002664	0.922715	0.922651	0.922496
161	0.530335	0.030335	0.000008	0.002827	0.922830	0.922597	0.922603
162	0.530666	0.030666	0.000015	0.002750	0.922480	0.922317	0.922358
163	0.530605	0.030605	0.000011	0.002583	0.922749	0.922516	0.922611
164	0.530279	0.030279	0.000012	0.002572	0.922739	0.922545	0.922625
165	0.528974	0.028974	0.000105	0.002622	0.925145	0.924866	0.924777
166	0.530615	0.030615	0.000012	0.002694	0.922525	0.922284	0.922329
167	0.530486	0.030486	0.000003	0.002560	0.922612	0.922609	0.922552
168	0.530816	0.030816	0.000026	0.002545	0.922327	0.922352	0.922404
169	0.530525	0.030525	0.000005	0.002532	0.922803	0.922646	0.922641
170	0.530675	0.030675	0.000016	0.002720	0.922329	0.922301	0.922111
171	0.530512	0.030512	0.000004	0.002844	0.922612	0.922381	0.922474
172	0.530393	0.030393	0.000004	0.002574	0.922848	0.922679	0.922558
173	0.530507	0.030507	0.000004	0.002709	0.922775	0.922595	0.922562
174	0.530214	0.030214	0.000017	0.002635	0.923049	0.922953	0.922899
175	0.530671	0.030671	0.000016	0.002735	0.922516	0.922449	0.922450
176	0.530491	0.030491	0.000003	0.002563	0.922644	0.922614	0.922508
177	0.530481	0.030481	0.000002	0.002589	0.922640	0.922531	0.922588
178	0.530123	0.030123	0.000023	0.002557	0.923256	0.923019	0.922991
179	0.530437	0.030437	0.000001	0.002504	0.922827	0.922746	0.922625
180	0.530967	0.030967	0.000037	0.002716	0.922304	0.922207	0.922285
181	0.530523	0.030523	0.000005	0.002705	0.922858	0.922673	0.922612
182	0.530616	0.030616	0.000012	0.002680	0.922808	0.922706	0.922677
183	0.530227	0.030227	0.000016	0.002649	0.923077	0.923032	0.922962
184	0.530911	0.030911	0.000033	0.002784	0.922301	0.922140	0.922091
185	0.530403	0.030403	0.000003	0.002779	0.922458	0.922499	0.922390
186	0.530433	0.030433	0.000001	0.002654	0.922915	0.922868	0.922837
187	0.530482	0.030482	0.000002	0.002638	0.922711	0.922619	0.922597

188	0.530636	0.030636	0.000013	0.002661	0.922615	0.922481	0.922442
189	0.530757	0.030757	0.000022	0.002768	0.922414	0.922361	0.922299
190	0.530713	0.030713	0.000019	0.002663	0.922477	0.922436	0.922289
191	0.530874	0.030874	0.000030	0.002736	0.922554	0.922452	0.922427
192	0.530349	0.030349	0.000007	0.002468	0.922665	0.922704	0.922581
193	0.530680	0.030680	0.000016	0.002928	0.922453	0.922308	0.922272
194	0.530356	0.030356	0.000007	0.002371	0.923006	0.922894	0.922854
195	0.530645	0.030645	0.000014	0.002613	0.922463	0.922338	0.922193
196	0.530732	0.030732	0.000020	0.002766	0.922641	0.922490	0.922553
197	0.530494	0.030494	0.000003	0.002812	0.922659	0.922598	0.922481
198	0.530934	0.030934	0.000034	0.002561	0.922353	0.922255	0.922287
199	0.530173	0.030173	0.000020	0.002564	0.923034	0.922900	0.922909

G.4 Análise visual da qualidade da imagem cifrada

Nessa seção um teste visual para verificar a qualidade de cifragem será apresentado. Como exemplo, exemplificaremos dois casos de cifragem com 25 passos sendo que no primeiro caso a cifragem foi feita com sucesso e no segundo ainda resta um padrão. A Figura G.1 apresenta a imagem original que foi utilizada nas cifragens e que teve um único bit perturbado na parte central da imagem. Cada uma dessas imagens foi cifrada por 25 passos de tempo por duas regras (Tabela G.4): uma que apresenta entropia alta (0.853477) e outra que apresenta entropia um pouco mais baixa (0.567607). A execução detalhada da cifragem de cada uma dessas regras ao longo de 25 passos de tempo está apresentada na Tabelas G.6 e na Tabela G.7 e a especificação de suas colunas está apresentada na Tabela G.5. A primeira regra gerou duas cifragens que são exibidas na Figura G.2, cifragem da imagem original, e a Figura G.3 representa a cifragem da imagem perturbada. A diferença dessas imagens realizadas com o operador XOR ($\oplus$) mostrada na Figura G.4 indica que a cifragem com $T = 25$ passos de tempo possui boa qualidade (padrão aleatório), indicando que não existem indícios da imagem original na cifrada. O mesmo não pode ser observado na cifragem da imagem G.1 a partir de uma regra com entropia mais baixa ao longo de $T = 25$ passos de tempo. A Figura G.6 apresenta a cifragem da imagem original e a Figura G.7 apresenta a cifragem da imagem perturbada. É possível observar na imagem da diferença gerada, Figura G.5 que grande parte da imagem original foi congelada em relação à cifragem da imagem perturbada (triângulos em preto). Isso indica a cifragem utilizando uma regra de baixa entropia não teve boa qualidade.

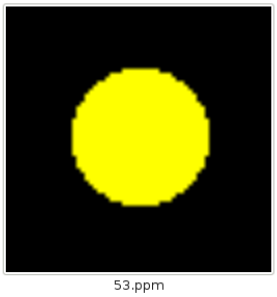


Figura G.1: Imagem original utilizada na cifragem (53.ppm) de 64×64 pixels no padrão RGB e no formato de "Portable Pixel Map".

G.4.1 Regras utilizadas

Tabela G.4: Núcleos de alta e baixa entropia utilizados na cifragem.

Índice	Núcleo
0	000011111100011010111000010010110000101111000011000100111111000
1	000000001111111100000000111111110000000011111111000000001111111

G.4.2 Especificação das colunas da tabela de resultados

Tabela G.5: Especificação detalhada das colunas.

Label	Descrição
A	Quantidade de passos de pre-imagem. (T)
B	Numero da figura utilizada (53.ppm).
C	Índice da regra utilizada.
D	Entropia da regra utilizada.
E	Quantidade de 1s observados.
F	Quantidade de 0s observados.
G	Porcentagem de 0s encontrados.
H	Entropia do canal R.
I	Entropia do canal G.
J	Entropia do canal B.

G.4.3 Resultados da análise visual

Tabela G.6: Resultado passo a passo da cifragem por $t = 1, 2, ..., 25$ passos de tempo a partir de uma regra com alta entropia.

A	B	C	D	E	F	G	H	I	J
1	53	0	0.853477	94256	4048	0.958822	0.135438	0.138762	0.139435
2	53	0	0.853477	92042	6262	0.936300	0.186585	0.182173	0.182540
3	53	0	0.853477	90019	8285	0.915721	0.227855	0.230883	0.229219
4	53	0	0.853477	87999	10305	0.895172	0.274224	0.282085	0.278317
5	53	0	0.853477	85398	12906	0.868713	0.333502	0.334867	0.335257
6	53	0	0.853477	82955	15349	0.843862	0.386489	0.387944	0.391848
7	53	0	0.853477	79853	18451	0.812307	0.444513	0.444635	0.448115
8	53	0	0.853477	77247	21057	0.785797	0.500866	0.498751	0.500015
9	53	0	0.853477	74636	23668	0.759237	0.548819	0.549664	0.548446
10	53	0	0.853477	72250	26054	0.734965	0.596120	0.596257	0.596454
11	53	0	0.853477	69743	28561	0.709462	0.643808	0.643578	0.642649
12	53	0	0.853477	67452	30852	0.686157	0.688962	0.690209	0.687798
13	53	0	0.853477	64897	33407	0.660166	0.734192	0.734826	0.733902
14	53	0	0.853477	62505	35799	0.635834	0.775021	0.775333	0.773606
15	53	0	0.853477	60340	37964	0.613810	0.812585	0.811266	0.812635
16	53	0	0.853477	58369	39935	0.593760	0.844606	0.843892	0.844162
17	53	0	0.853477	56245	42059	0.572154	0.870489	0.869866	0.869814
18	53	0	0.853477	54615	43689	0.555573	0.893629	0.893138	0.893528
19	53	0	0.853477	53243	45061	0.541616	0.909684	0.911282	0.910877
20	53	0	0.853477	51922	46382	0.528178	0.923884	0.924876	0.924498
21	53	0	0.853477	50750	47554	0.516256	0.934442	0.934607	0.934562
22	53	0	0.853477	50361	47943	0.512299	0.940005	0.940622	0.940985
23	53	0	0.853477	49694	48610	0.505513	0.943672	0.943423	0.943922
24	53	0	0.853477	49766	48538	0.506246	0.944880	0.943623	0.944171
25	53	0	0.853477	49299	49005	0.501495	0.944673	0.945513	0.944382

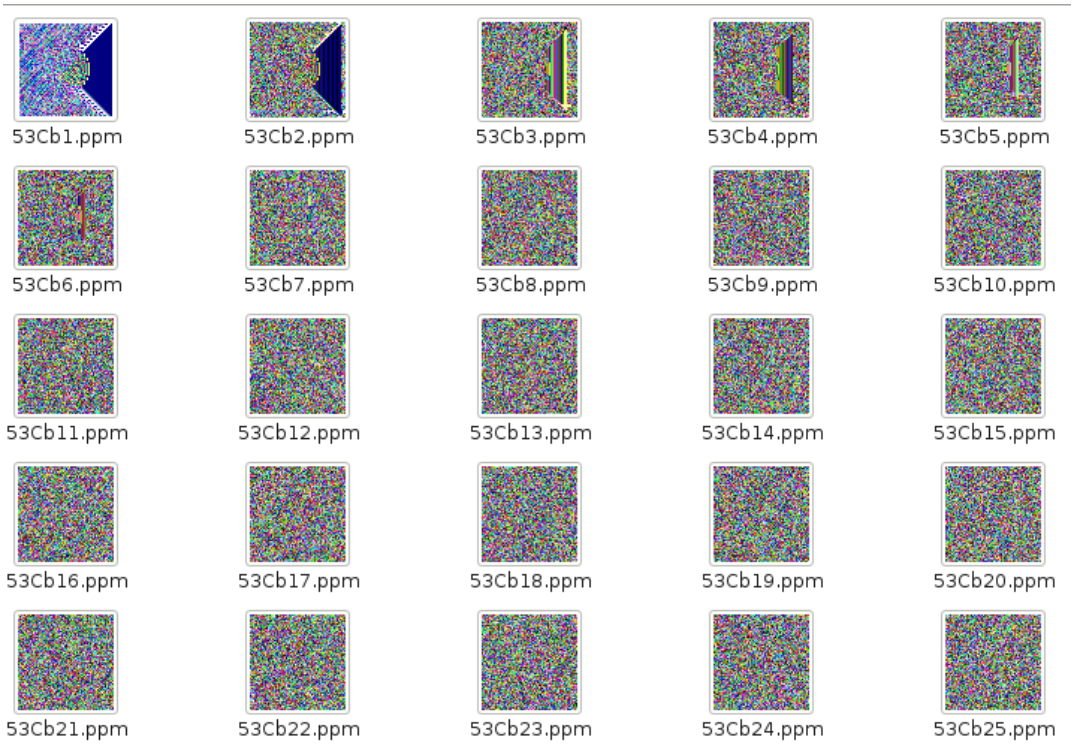


Figura G.2: Evolução da imagem original sendo cifrada por t = 1, 2, ..., 25 passos de tempo.

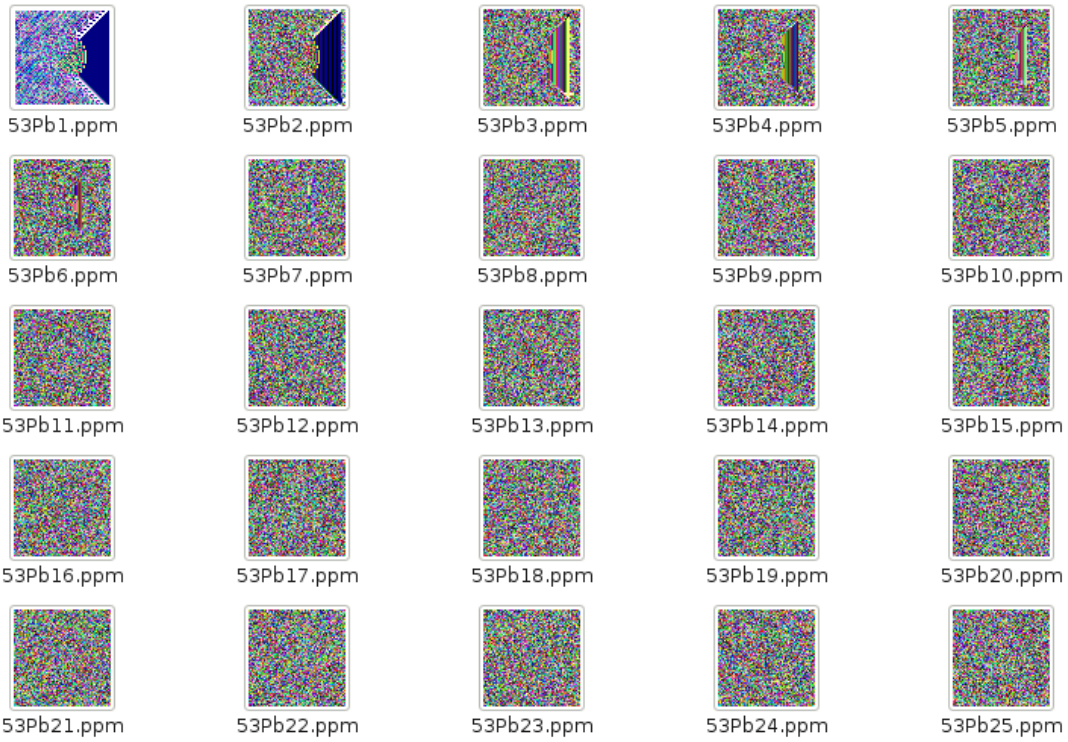


Figura G.3: Evolução da imagem perturbada sendo cifrada por t = 1, 2, ..., 25 passos de tempo.

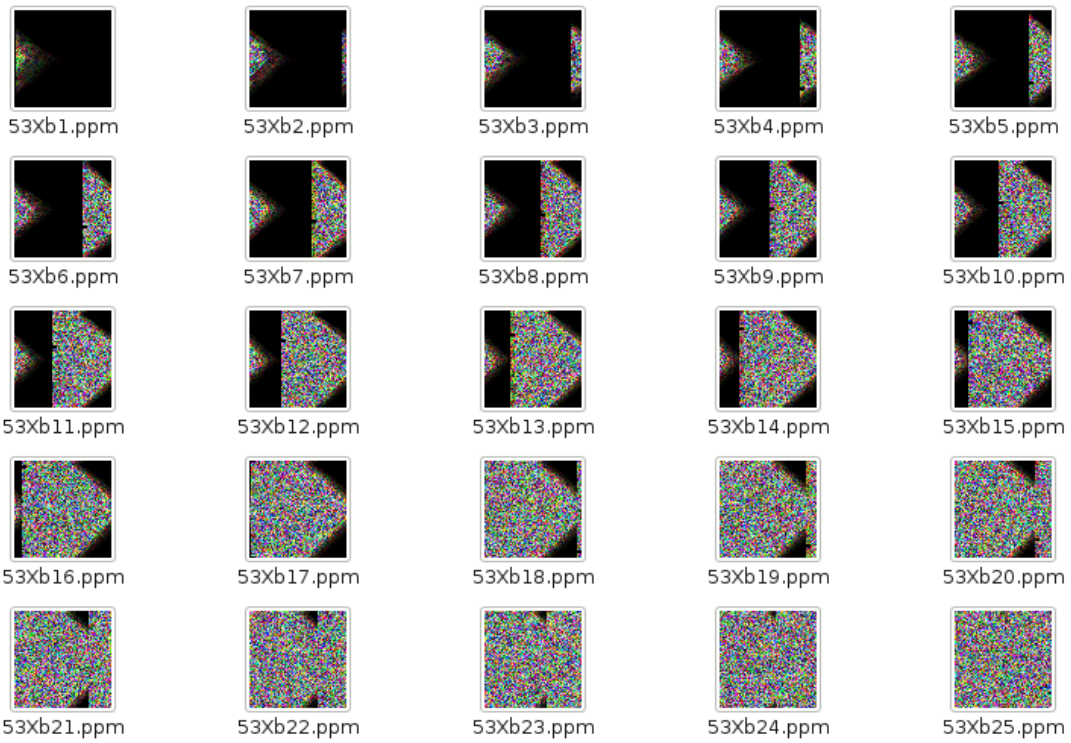


Figura G.4: Evolução da imagem XOR ao longo de t = 1, 2, ..., 25 passos de tempo.

Tabela G.7: Resultado passo a passo da cifragem por $t = 1, 2, ..., 25$ passos de tempo a partir de uma regra com baixa entropia.

A	B	C	D	E	F	G	H	I	J
1	53	1	0.567607	98273	31	0.999685	0.010859	0.000000	0.000000
2	53	1	0.567607	98133	171	0.998260	0.034101	0.000000	0.000000
3	53	1	0.567607	96076	2228	0.977336	0.099009	0.084192	0.090079
4	53	1	0.567607	95378	2926	0.970235	0.118294	0.107187	0.113624
5	53	1	0.567607	94305	3999	0.959320	0.155106	0.141237	0.145709
6	53	1	0.567607	93330	4974	0.949402	0.182742	0.169947	0.178750
7	53	1	0.567607	91705	6599	0.932872	0.219344	0.206215	0.214742
8	53	1	0.567607	90672	7632	0.922363	0.252106	0.234519	0.242593
9	53	1	0.567607	89272	9032	0.908122	0.275256	0.258972	0.267328
10	53	1	0.567607	87998	10306	0.895162	0.311265	0.293952	0.303616
11	53	1	0.567607	85475	12829	0.869497	0.360908	0.340255	0.350636
12	53	1	0.567607	83359	14945	0.847972	0.415051	0.390054	0.403767
13	53	1	0.567607	80823	17481	0.822174	0.474044	0.441630	0.470107
14	53	1	0.567607	78741	19563	0.800995	0.528529	0.499448	0.517418
15	53	1	0.567607	76364	21940	0.776815	0.571623	0.546837	0.562147
16	53	1	0.567607	74221	24083	0.755015	0.608790	0.586949	0.601785
17	53	1	0.567607	72348	25956	0.735962	0.644350	0.618887	0.635528
18	53	1	0.567607	70449	27855	0.716644	0.683931	0.663814	0.676155
19	53	1	0.567607	67827	30477	0.689972	0.721043	0.704559	0.713022
20	53	1	0.567607	65839	32465	0.669749	0.754170	0.736079	0.744617
21	53	1	0.567607	63487	34817	0.645823	0.785993	0.768897	0.778610
22	53	1	0.567607	61819	36485	0.628855	0.811332	0.796324	0.804674
23	53	1	0.567607	60477	37827	0.615204	0.832145	0.821061	0.828898
24	53	1	0.567607	58960	39344	0.599772	0.853302	0.842166	0.848844
25	53	1	0.567607	57438	40866	0.584290	0.869646	0.858196	0.866450

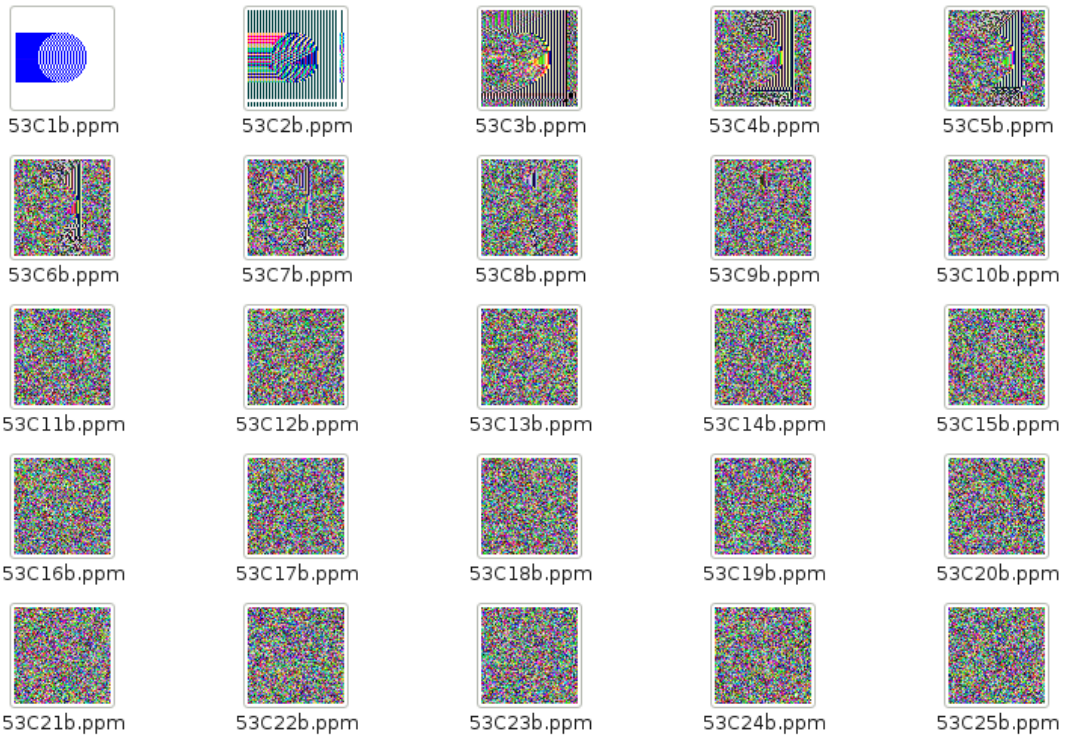


Figura G.5: Evolução da imagem original sendo cifrada por t = 1, 2, ..., 25 passos de tempo.

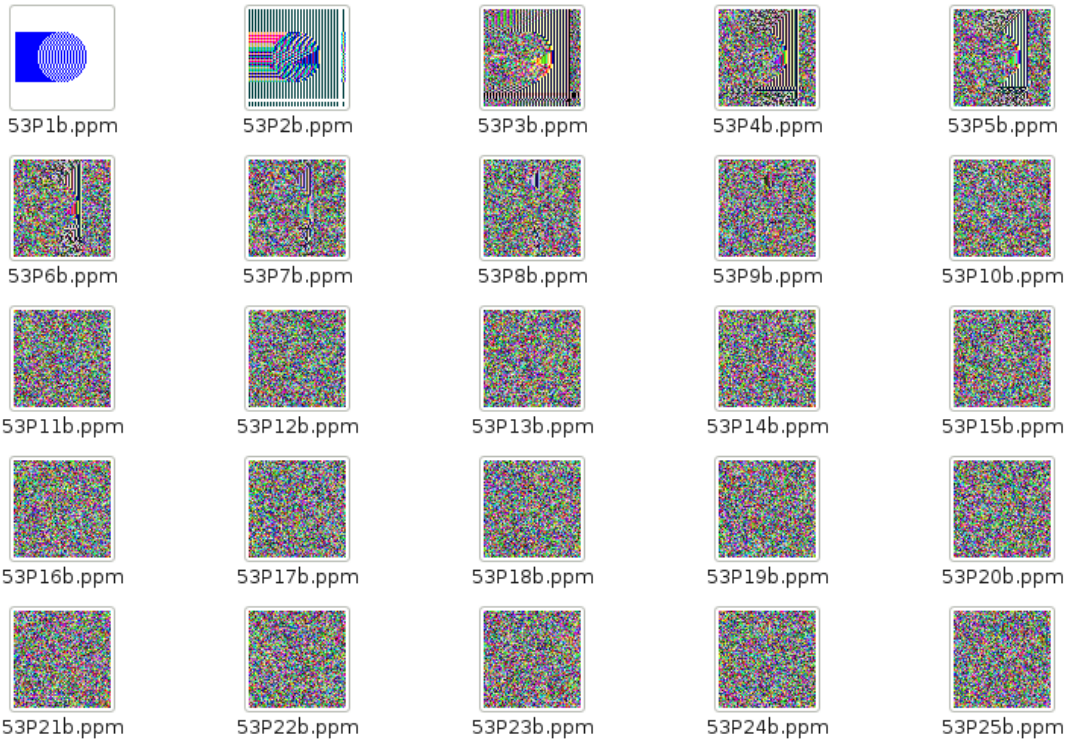


Figura G.6: Evolução da imagem perturbada sendo cifrada por t = 1, 2, ..., 25 passos de tempo.

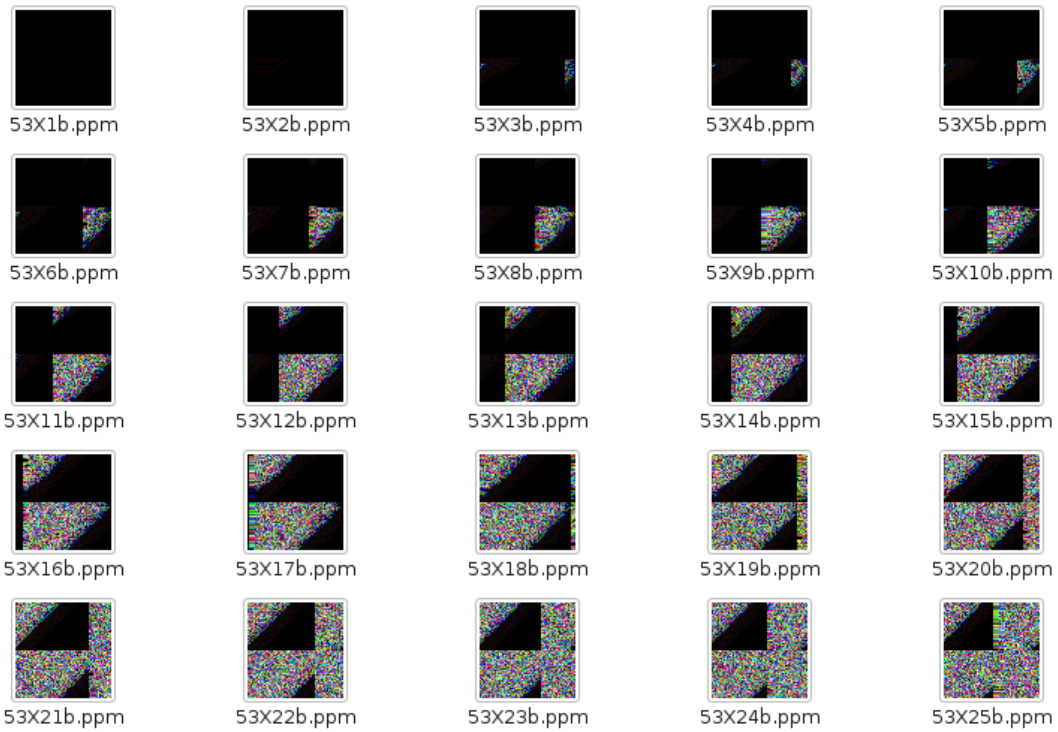


Figura G.7: Evolução da imagem XOR ao longo de t = 1, 2, ..., 25 passos de tempo.

G.5 Experimentos com as regras definidas manualmente

Tabela G.8: Conjunto das 50 regras de baixa que foram definidas manualmente.

Índice	Núcleo
0	00000000111111110000000011111111000000001111111100000000111111111
1	11111111000000001111111100000000111111110000000011111111000000000
2	11111111000000000000000011111111111111110000000000000000111111111
3	00000000111111111111111100000000000000001111111111111111000000000
4	11111111111111100000000000000000111111111111111100000000000000000
5	00000000000000011111111111111110000000000000000111111111111111111
6	00001111111111111111111111111111111111111111111111111111111110000
7	00000000111111111111111111111111111111111111111111111111111000000000
8	00000000000011111111111111111111111111111111111111111000000000000
9	00000000000000011111111111111111111111111111111100000000000000000
10	00000000000000000001111111111111111111111110000000000000000000000
11	00000000000000000000001111111111111110000000000000000000000000000
12	00000000000000000000000000111111110000000000000000000000000000000
13	11110000000000000000000000000000000000000000000000000000000001111
14	11111111000000000000000000000000000000000000000000000000011111111
15	11111111111100000000000000000000000000000000000000111111111111
16	11111111111111100000000000000000000000000000000111111111111111
17	11111111111111111110000000000000000000000001111111111111111111
18	11111111111111111111100000000000000001111111111111111111111111
19	11111111111111111111111100000000111111111111111111111111111111
20	00001111111111111111111111111111111111111111111111111111111111
21	00000000111111111111111111111111111111111111111111111111111111
22	00000000000011111111111111111111111111111111111111111111111111
23	00000000000000011111111111111111111111111111111111111111111111
24	00000000000000000011111111111111111111111111111111111111111111
25	00000000000000000000011111111111111111111111111111111111111111
26	00000000000000000000000011111111111111111111111111111111111111
27	00000000000000000000000000011111111111111111111111111111111111
28	00000000000000000000000000000011111111111111111111111111111111
29	00000000000000000000000000000000111111111111111111111111111111
30	00000000000000000000000000000000011111111111111111111111111111
31	00000000000000000000000000000000000000000000000011111111111111
32	00000000000000000000000000000000000000000000000000111111111111
33	00000000000000000000000000000000000000000000000000011111111111
34	00000000000000000000000000000000000000000000000000000000001111
35	11110000000000000000000000000000000000000000000000000000000000000
36	11111111000000000000000000000000000000000000000000000000000000000
37	11111111111100000000000000000000000000000000000000000000000000000
38	11111111111111110000000000000000000000000000000000000000000000000
39	11111111111111111111000000000000000000000000000000000000000000000

40	11111111111111111111111000000000000000000000000000000000000000
41	11111111111111111111111110000000000000000000000000000000000000
42	11111111111111111111111111110000000000000000000000000000000000
43	11111111111111111111111111111110000000000000000000000000000000
44	11111111111111111111111111111111110000000000000000000000000000
45	11111111111111111111111111111111111110000000000000000000000000
46	11111111111111111111111111111111111111110000000000000000000000
47	11111111111111111111111111111111111111111110000000000000000000
48	11111111111111111111111111111111111111111111100000000000000000
49	11111111111111111111111111111111111111111111111110000000000000

G.5.1 Resultados obtidos com a utilização de regras com baixa entropia

Tabela G.9: Especificação detalhada das colunas da Tabela.

Label	Descrição
A	Imagem utilizada.
B	Regra utilizada $\{0, 1, 2, ..., 49\}$.
C	Entropia da regra utilizada..
D	Quantidade de 1's observados.
E	Quantidade de 0's observados.
F	Porcentagem de 0's na imagem do XOR observados.
G	Entropia do canal R
H	Entropia do canal G.
I	Entropia do canal B.

Tabela G.10: Resultados experimentos com as 50 regras regulares

A	B	C	D	E	F	G	H	I
53	0	0.567607	57438	40866	0.584290	0.869646	0.858196	0.866450
53	1	0.567607	57355	40949	0.583445	0.871567	0.858347	0.863909
53	2	0.436940	51119	47185	0.520009	0.940214	0.937814	0.938943
53	3	0.436940	52111	46193	0.530101	0.935944	0.932464	0.934321
53	4	0.436769	49769	48535	0.506276	0.945018	0.943872	0.944806
53	5	0.436769	49952	48352	0.508138	0.943360	0.943824	0.943257
53	6	0.234249	74358	23946	0.756409	0.634409	0.632496	0.637733
53	7	0.293273	51547	46757	0.524363	0.933867	0.933595	0.934179
53	8	0.322303	50348	47956	0.512166	0.939212	0.938774	0.940161
53	9	0.331344	49805	48499	0.506643	0.942775	0.943052	0.943057
53	10	0.322303	50191	48113	0.510569	0.939957	0.939023	0.939714
53	11	0.293273	55172	43132	0.561239	0.918031	0.919517	0.921128
53	12	0.234248	79602	18702	0.809753	0.590502	0.599294	0.597287
53	13	0.234248	89265	9039	0.908051	0.351228	0.369832	0.351048
53	14	0.293273	55168	43136	0.561198	0.920335	0.919337	0.919026
53	15	0.322303	50691	47613	0.515656	0.938225	0.938625	0.938998
53	16	0.331344	50109	48195	0.509735	0.941592	0.942843	0.941984
53	17	0.322303	50311	47993	0.511790	0.939958	0.939516	0.940594
53	18	0.293273	51061	47243	0.519419	0.937624	0.936216	0.936850
53	19	0.234249	57665	40639	0.586599	0.898899	0.896542	0.898442
53	20	0.171941	97464	840	0.991455	0.058724	0.056381	0.051700
53	21	0.234249	66374	31930	0.675191	0.775738	0.777521	0.777620
53	22	0.268966	56025	42279	0.569916	0.900705	0.898910	0.900927
53	23	0.293273	50574	47730	0.514465	0.941331	0.941243	0.941683
53	24	0.310588	51349	46955	0.522349	0.930794	0.931290	0.930194
53	25	0.322303	50108	48196	0.509725	0.939773	0.940909	0.940103
53	26	0.329109	50072	48232	0.509359	0.942312	0.943076	0.942614
53	27	0.331205	50339	47965	0.512075	0.943494	0.942923	0.942279
53	28	0.329109	50344	47960	0.512126	0.940645	0.940480	0.942139
53	29	0.322303	50640	47664	0.515137	0.940570	0.941084	0.939650
53	30	0.310588	51935	46369	0.528310	0.934116	0.934750	0.934293
53	31	0.293273	52939	45365	0.538523	0.932410	0.933794	0.933284
53	32	0.268966	64537	33767	0.656504	0.833593	0.838611	0.829292
53	33	0.234248	90577	7727	0.921397	0.333124	0.333859	0.328431
53	34	0.171941	97437	867	0.991180	0.054402	0.056508	0.060374
53	35	0.171941	98295	9	0.999908	0.001360	0.002358	0.000000
53	36	0.234248	97737	567	0.994232	0.043439	0.040541	0.041026
53	37	0.268966	63659	34645	0.647573	0.856318	0.865798	0.860356
53	38	0.293273	52198	46106	0.530986	0.935609	0.938487	0.936661
53	39	0.310588	52181	46123	0.530813	0.922094	0.923937	0.923052
53	40	0.322303	50378	47926	0.512471	0.941587	0.941051	0.941859
53	41	0.329109	50287	48017	0.511546	0.940822	0.940809	0.940240

53	42	0.331205	49657	48647	0.505137	0.943611	0.942715	0.942963
53	43	0.329109	50052	48252	0.509155	0.943318	0.944529	0.944151
53	44	0.322303	50226	48078	0.510925	0.941691	0.942468	0.940584
53	45	0.310588	50555	47749	0.514272	0.941785	0.940556	0.940783
53	46	0.293273	50275	48029	0.511424	0.942586	0.942903	0.943301
53	47	0.268966	53119	45185	0.540354	0.917727	0.918720	0.916321
53	48	0.234249	58364	39940	0.593709	0.885809	0.880011	0.880143
53	49	0.171941	85136	13168	0.866048	0.440435	0.448425	0.450756
122	0	0.567607	57058	41246	0.580424	0.872599	0.859570	0.866467
122	1	0.567607	57180	41124	0.581665	0.869320	0.858746	0.864282
122	2	0.436940	51504	46800	0.523926	0.940002	0.937215	0.938092
122	3	0.436940	51441	46863	0.523285	0.939394	0.935894	0.938676
122	4	0.436769	49394	48910	0.502462	0.944217	0.945228	0.944993
122	5	0.436769	50074	48230	0.509379	0.943054	0.943576	0.943329
122	6	0.234249	53817	44487	0.547455	0.924120	0.924254	0.925236
122	7	0.293273	50307	47997	0.511749	0.942576	0.942067	0.942036
122	8	0.322303	50272	48032	0.511393	0.940238	0.940248	0.940228
122	9	0.331344	49989	48315	0.508514	0.942246	0.941741	0.942274
122	10	0.322303	50633	47671	0.515065	0.940114	0.940598	0.941490
122	11	0.293273	55529	42775	0.564870	0.917004	0.918690	0.917992
122	12	0.234248	92952	5352	0.945557	0.228121	0.240465	0.235346
122	13	0.234248	72408	25896	0.736572	0.743434	0.755212	0.749784
122	14	0.293273	54104	44200	0.550374	0.926213	0.929898	0.927374
122	15	0.322303	50049	48255	0.509125	0.940919	0.940945	0.940875
122	16	0.331344	50017	48287	0.508799	0.942380	0.943059	0.942144
122	17	0.322303	50053	48251	0.509165	0.941514	0.942727	0.941375
122	18	0.293273	50948	47356	0.518270	0.940552	0.939716	0.939939
122	19	0.234249	63477	34827	0.645721	0.820881	0.821936	0.818489
122	20	0.171941	93392	4912	0.950033	0.225848	0.223364	0.215799
122	21	0.234249	62461	35843	0.635386	0.844068	0.844171	0.843821
122	22	0.268966	54147	44157	0.550812	0.916329	0.914960	0.915422
122	23	0.293273	51047	47257	0.519277	0.937275	0.938313	0.937805
122	24	0.310588	50436	47868	0.513062	0.939748	0.939088	0.939886
122	25	0.322303	50359	47945	0.512278	0.940104	0.939642	0.939255
122	26	0.329109	50476	47828	0.513468	0.939052	0.938580	0.939350
122	27	0.331205	49967	48337	0.508291	0.942778	0.943300	0.943365
122	28	0.329109	50405	47899	0.512746	0.939036	0.939636	0.940121
122	29	0.322303	50368	47936	0.512370	0.940760	0.940659	0.940319
122	30	0.310588	51531	46773	0.524200	0.938233	0.938491	0.938423
122	31	0.293273	52729	45575	0.536387	0.932052	0.931989	0.932843
122	32	0.268966	62151	36153	0.632233	0.856159	0.861607	0.858612
122	33	0.234248	76637	21667	0.779592	0.647782	0.651099	0.649648
122	34	0.171941	93596	4708	0.952108	0.212460	0.205528	0.201206
122	35	0.171941	95801	2503	0.974538	0.127687	0.131322	0.136381

122	36	0.234248	83260	15044	0.846965	0.499015	0.523144	0.516161
122	37	0.268966	61629	36675	0.626923	0.873810	0.880974	0.873814
122	38	0.293273	52379	45925	0.532827	0.931724	0.930366	0.930016
122	39	0.310588	51226	47078	0.521098	0.931811	0.931444	0.931482
122	40	0.322303	50798	47506	0.516744	0.935854	0.934576	0.935813
122	41	0.329109	50344	47960	0.512126	0.937948	0.938485	0.937100
122	42	0.331205	50177	48127	0.510427	0.942963	0.942585	0.942560
122	43	0.329109	50395	47909	0.512644	0.940718	0.941109	0.940694
122	44	0.322303	50258	48046	0.511251	0.942266	0.942308	0.942664
122	45	0.310588	50580	47724	0.514526	0.939691	0.940736	0.940209
122	46	0.293273	50654	47650	0.515279	0.939305	0.940417	0.939801
122	47	0.268966	51342	46962	0.522278	0.937533	0.937030	0.937980
122	48	0.234249	54769	43535	0.557139	0.919049	0.918033	0.916774
122	49	0.171941	92383	5921	0.939768	0.244069	0.245706	0.242232
137	0	0.567607	57599	40705	0.585927	0.870210	0.860036	0.864353
137	1	0.567607	57215	41089	0.582021	0.871727	0.860336	0.865552
137	2	0.436940	51963	46341	0.528595	0.937594	0.932189	0.936122
137	3	0.436940	51580	46724	0.524699	0.939244	0.934617	0.938708
137	4	0.436769	49969	48335	0.508311	0.943694	0.944059	0.944323
137	5	0.436769	49885	48419	0.507456	0.944342	0.944230	0.944123
137	6	0.234249	75090	23214	0.763855	0.652622	0.649635	0.647627
137	7	0.293273	50960	47344	0.518392	0.938346	0.938033	0.938822
137	8	0.322303	50115	48189	0.509796	0.942288	0.942733	0.942439
137	9	0.331344	49898	48406	0.507589	0.942917	0.943259	0.942946
137	10	0.322303	50906	47398	0.517843	0.936093	0.937189	0.937368
137	11	0.293273	55879	42425	0.568431	0.910359	0.911632	0.911119
137	12	0.234248	89375	8929	0.909169	0.354726	0.367728	0.351683
137	13	0.234248	72090	26214	0.733337	0.764786	0.769939	0.754057
137	14	0.293273	53566	44738	0.544902	0.931837	0.933728	0.933084
137	15	0.322303	50408	47896	0.512777	0.941915	0.941503	0.940928
137	16	0.331344	50368	47936	0.512370	0.942064	0.941338	0.941917
137	17	0.322303	50093	48211	0.509572	0.941297	0.940829	0.941295
137	18	0.293273	51077	47227	0.519582	0.936792	0.938031	0.936787
137	19	0.234249	58676	39628	0.596883	0.886025	0.888784	0.887501
137	20	0.171941	94850	3454	0.964864	0.158058	0.159733	0.156925
137	21	0.234249	58278	40026	0.592834	0.884808	0.883963	0.881996
137	22	0.268966	52710	45594	0.536194	0.928287	0.928584	0.928545
137	23	0.293273	51331	46973	0.522166	0.936515	0.937634	0.936850
137	24	0.310588	50827	47477	0.517039	0.938431	0.937221	0.937933
137	25	0.322303	50298	48006	0.511658	0.940604	0.941338	0.941342
137	26	0.329109	49892	48412	0.507528	0.942843	0.941908	0.942472
137	27	0.331205	49946	48358	0.508077	0.942352	0.942753	0.942288
137	28	0.329109	49995	48309	0.508575	0.941419	0.941098	0.941610
137	29	0.322303	50570	47734	0.514425	0.939329	0.939567	0.940223

137	30	0.310588	51303	47001	0.521881	0.937579	0.938805	0.937310
137	31	0.293273	52257	46047	0.531586	0.935175	0.937492	0.936479
137	32	0.268966	59683	38621	0.607127	0.887543	0.886926	0.884812
137	33	0.234248	77234	21070	0.785665	0.640466	0.650778	0.641469
137	34	0.171941	93504	4800	0.951172	0.213140	0.207765	0.207958
137	35	0.171941	96574	1730	0.982402	0.103562	0.114115	0.108648
137	36	0.234248	87427	10877	0.889353	0.409343	0.421934	0.410521
137	37	0.268966	65139	33165	0.662628	0.833076	0.835282	0.834369
137	38	0.293273	52469	45835	0.533742	0.930112	0.930345	0.929827
137	39	0.310588	51405	46899	0.522919	0.931578	0.931611	0.930954
137	40	0.322303	50620	47684	0.514933	0.941187	0.941904	0.939269
137	41	0.329109	50345	47959	0.512136	0.940081	0.939754	0.938340
137	42	0.331205	50254	48050	0.511210	0.941370	0.943151	0.940079
137	43	0.329109	50416	47888	0.512858	0.940713	0.940158	0.940417
137	44	0.322303	50145	48159	0.510101	0.941714	0.942123	0.942048
137	45	0.310588	50652	47652	0.515259	0.939758	0.939900	0.938549
137	46	0.293273	50370	47934	0.512390	0.940726	0.941688	0.940478
137	47	0.268966	52216	46088	0.531169	0.928884	0.929953	0.928636
137	48	0.234249	55244	43060	0.561971	0.912582	0.913988	0.913728
137	49	0.171941	79558	18746	0.809306	0.568354	0.571677	0.573545
147	0	0.567607	57679	40625	0.586741	0.870509	0.858883	0.864616
147	1	0.567607	57205	41099	0.581919	0.870247	0.858989	0.864390
147	2	0.436940	50863	47441	0.517405	0.941762	0.938387	0.940079
147	3	0.436940	51070	47234	0.519511	0.941004	0.937736	0.939976
147	4	0.436769	49851	48453	0.507111	0.944165	0.943831	0.943549
147	5	0.436769	49725	48579	0.505829	0.944651	0.945104	0.944553
147	6	0.234249	53489	44815	0.544118	0.930775	0.931088	0.930073
147	7	0.293273	50512	47792	0.513835	0.941983	0.941633	0.941569
147	8	0.322303	49893	48411	0.507538	0.942509	0.942467	0.942331
147	9	0.331344	49676	48628	0.505330	0.943511	0.943554	0.943329
147	10	0.322303	50558	47746	0.514303	0.938893	0.939605	0.939655
147	11	0.293273	56078	42226	0.570455	0.910463	0.911310	0.911938
147	12	0.234248	87095	11209	0.885976	0.406295	0.417886	0.412352
147	13	0.234248	88456	9848	0.899821	0.376312	0.392628	0.388141
147	14	0.293273	54675	43629	0.556183	0.923800	0.923773	0.923534
147	15	0.322303	50248	48056	0.511149	0.942708	0.942221	0.942337
147	16	0.331344	49976	48328	0.508382	0.942650	0.943307	0.942604
147	17	0.322303	50446	47858	0.513163	0.939046	0.939688	0.939135
147	18	0.293273	50977	47327	0.518565	0.938177	0.938436	0.937978
147	19	0.234249	66484	31820	0.676310	0.774375	0.777603	0.772819
147	20	0.171941	93697	4607	0.953135	0.208726	0.209484	0.211354
147	21	0.234249	62341	35963	0.634165	0.833603	0.834565	0.835711
147	22	0.268966	55258	43046	0.562113	0.903680	0.906703	0.905217
147	23	0.293273	51446	46858	0.523336	0.936310	0.936525	0.934777

147	24	0.310588	50201	48103	0.510671	0.941995	0.941812	0.941723
147	25	0.322303	50164	48140	0.510295	0.939714	0.940790	0.940015
147	26	0.329109	50176	48128	0.510417	0.939977	0.940424	0.939516
147	27	0.331205	49834	48470	0.506938	0.943335	0.943233	0.943339
147	28	0.329109	50327	47977	0.511953	0.941659	0.941360	0.942050
147	29	0.322303	50501	47803	0.513723	0.940785	0.940322	0.940116
147	30	0.310588	51392	46912	0.522786	0.938400	0.939413	0.938507
147	31	0.293273	53171	45133	0.540883	0.929740	0.931417	0.929720
147	32	0.268966	56783	41521	0.577627	0.918708	0.918348	0.917998
147	33	0.234248	76063	22241	0.773753	0.659485	0.656378	0.653490
147	34	0.171941	94029	4275	0.956512	0.190308	0.190246	0.187215
147	35	0.171941	95611	2693	0.972605	0.143331	0.142572	0.144449
147	36	0.234248	86454	11850	0.879456	0.439811	0.445088	0.441558
147	37	0.268966	63617	34687	0.647146	0.851510	0.859164	0.854652
147	38	0.293273	52722	45582	0.536316	0.922273	0.925646	0.923426
147	39	0.310588	51164	47140	0.520467	0.935134	0.935649	0.936529
147	40	0.322303	50382	47922	0.512512	0.942868	0.943008	0.943223
147	41	0.329109	49940	48364	0.508016	0.942093	0.941561	0.941847
147	42	0.331205	49820	48484	0.506795	0.943843	0.944071	0.944544
147	43	0.329109	50350	47954	0.512187	0.940613	0.939708	0.941225
147	44	0.322303	49940	48364	0.508016	0.941929	0.941368	0.941720
147	45	0.310588	50386	47918	0.512553	0.938678	0.939433	0.939310
147	46	0.293273	50893	47411	0.517710	0.939270	0.938154	0.937657
147	47	0.268966	52798	45506	0.537089	0.922162	0.922623	0.921122
147	48	0.234249	59607	38697	0.606354	0.873774	0.875025	0.876266
147	49	0.171941	89336	8968	0.908773	0.330140	0.338281	0.334271
152	0	0.567607	57258	41046	0.582458	0.872892	0.859896	0.865144
152	1	0.567607	57143	41161	0.581289	0.870662	0.859299	0.867493
152	2	0.436940	51347	46957	0.522329	0.940494	0.938291	0.939622
152	3	0.436940	51012	47292	0.518921	0.941698	0.936811	0.939712
152	4	0.436769	49400	48904	0.502523	0.944602	0.944525	0.944996
152	5	0.436769	49975	48329	0.508372	0.943019	0.943893	0.943450
152	6	0.234249	54726	43578	0.556702	0.917263	0.921455	0.920891
152	7	0.293273	51019	47285	0.518992	0.937478	0.937980	0.938695
152	8	0.322303	50550	47754	0.514221	0.939843	0.939579	0.940521
152	9	0.331344	50141	48163	0.510061	0.944152	0.943421	0.943401
152	10	0.322303	50539	47765	0.514109	0.940476	0.940414	0.940217
152	11	0.293273	55524	42780	0.564819	0.918748	0.920585	0.919753
152	12	0.234248	93635	4669	0.952504	0.222400	0.231233	0.233164
152	13	0.234248	73974	24330	0.752502	0.717868	0.730459	0.728401
152	14	0.293273	54995	43309	0.559438	0.918902	0.920408	0.919193
152	15	0.322303	50205	48099	0.510712	0.942671	0.942844	0.942372
152	16	0.331344	50013	48291	0.508759	0.942766	0.943818	0.943938
152	17	0.322303	50425	47879	0.512950	0.940626	0.940013	0.940736

152	18	0.293273	51700	46604	0.525920	0.931452	0.932114	0.931105
152	19	0.234249	63660	34644	0.647583	0.822909	0.829074	0.822560
152	20	0.171941	96769	1535	0.984385	0.090308	0.086343	0.085524
152	21	0.234249	63027	35277	0.641144	0.827318	0.833725	0.826966
152	22	0.268966	53391	44913	0.543121	0.924815	0.925495	0.923410
152	23	0.293273	51131	47173	0.520131	0.940582	0.940985	0.939265
152	24	0.310588	50311	47993	0.511790	0.938278	0.938912	0.938159
152	25	0.322303	50469	47835	0.513397	0.938876	0.939452	0.939091
152	26	0.329109	50255	48049	0.511220	0.939265	0.939967	0.939655
152	27	0.331205	50227	48077	0.510935	0.942271	0.941746	0.942618
152	28	0.329109	50533	47771	0.514048	0.940628	0.940865	0.940582
152	29	0.322303	50432	47872	0.513021	0.941808	0.942694	0.942870
152	30	0.310588	51837	46467	0.527313	0.934491	0.934621	0.934564
152	31	0.293273	52910	45394	0.538228	0.930979	0.930191	0.929305
152	32	0.268966	59184	39120	0.602051	0.895450	0.897466	0.895553
152	33	0.234248	73656	24648	0.749268	0.704244	0.717472	0.709195
152	34	0.171941	96139	2165	0.977977	0.124616	0.118484	0.127285
152	35	0.171941	98253	51	0.999481	0.005413	0.005748	0.006277
152	36	0.234248	98168	136	0.998617	0.013080	0.012134	0.015691
152	37	0.268966	74018	24286	0.752950	0.679716	0.691270	0.674964
152	38	0.293273	59623	38681	0.606517	0.847365	0.847123	0.845389
152	39	0.310588	91493	6811	0.930715	0.209553	0.213119	0.202177
152	40	0.322303	51406	46898	0.522929	0.931313	0.931443	0.930994
152	41	0.329109	50723	47581	0.515981	0.937907	0.937201	0.937506
152	42	0.331205	50161	48143	0.510264	0.939415	0.939903	0.939911
152	43	0.329109	50383	47921	0.512522	0.942626	0.942249	0.941629
152	44	0.322303	50420	47884	0.512899	0.941035	0.943096	0.942441
152	45	0.310588	50810	47494	0.516866	0.935695	0.936071	0.937043
152	46	0.293273	50970	47334	0.518494	0.938739	0.937172	0.936570
152	47	0.268966	52440	45864	0.533447	0.930592	0.930125	0.930928
152	48	0.234249	59409	38895	0.604340	0.872734	0.877621	0.873445
152	49	0.171941	83488	14816	0.849284	0.482482	0.482659	0.482390
168	0	0.567607	57371	40933	0.583608	0.866991	0.855722	0.862856
168	1	0.567607	57434	40870	0.584249	0.867372	0.858200	0.863758
168	2	0.436940	51588	46716	0.524780	0.939509	0.935099	0.937042
168	3	0.436940	50781	47523	0.516571	0.940650	0.937742	0.939423
168	4	0.436769	49832	48472	0.506917	0.944235	0.944818	0.945072
168	5	0.436769	49901	48403	0.507619	0.944497	0.944876	0.944714
168	6	0.234249	56116	42188	0.570841	0.908700	0.908370	0.910580
168	7	0.293273	50982	47322	0.518616	0.938087	0.938151	0.938135
168	8	0.322303	49788	48516	0.506470	0.943110	0.942328	0.942400
168	9	0.331344	50021	48283	0.508840	0.943135	0.942007	0.942417
168	10	0.322303	50596	47708	0.514689	0.937870	0.938913	0.939970
168	11	0.293273	55546	42758	0.565043	0.915522	0.915163	0.915138

168	12	0.234248	79061	19243	0.804250	0.597406	0.609801	0.599773
168	13	0.234248	72364	25940	0.736125	0.743198	0.751027	0.740077
168	14	0.293273	54863	43441	0.558095	0.922020	0.922793	0.921241
168	15	0.322303	50035	48269	0.508982	0.942361	0.941737	0.941949
168	16	0.331344	50105	48199	0.509694	0.942320	0.942918	0.940987
168	17	0.322303	50117	48187	0.509816	0.942809	0.942022	0.941836
168	18	0.293273	50794	47510	0.516703	0.940678	0.940650	0.940928
168	19	0.234249	57454	40850	0.584452	0.899818	0.903038	0.901761
168	20	0.171941	97355	949	0.990346	0.069611	0.059722	0.058815
168	21	0.234249	68103	30201	0.692780	0.742138	0.738035	0.738286
168	22	0.268966	54344	43960	0.552816	0.916976	0.917559	0.916102
168	23	0.293273	51690	46614	0.525818	0.934706	0.935163	0.933207
168	24	0.310588	50799	47505	0.516754	0.938049	0.936573	0.938235
168	25	0.322303	50513	47791	0.513845	0.939624	0.938538	0.940046
168	26	0.329109	50238	48066	0.511047	0.940440	0.940495	0.939941
168	27	0.331205	50346	47958	0.512146	0.940743	0.941229	0.941576
168	28	0.329109	50602	47702	0.514750	0.938729	0.939117	0.938838
168	29	0.322303	50008	48296	0.508708	0.943167	0.942518	0.942683
168	30	0.310588	52333	45971	0.532359	0.929122	0.930879	0.931154
168	31	0.293273	53957	44347	0.548879	0.925832	0.926457	0.924420
168	32	0.268966	63584	34720	0.646810	0.837952	0.839643	0.839635
168	33	0.234248	79148	19156	0.805135	0.595704	0.601255	0.599537
168	34	0.171941	98254	50	0.999491	0.006009	0.005193	0.005708
168	35	0.171941	98123	181	0.998159	0.020351	0.015737	0.014595
168	36	0.234248	95988	2316	0.976440	0.137357	0.144641	0.129698
168	37	0.268966	72443	25861	0.736928	0.706390	0.711443	0.702115
168	38	0.293273	52548	45756	0.534546	0.932424	0.933192	0.933169
168	39	0.310588	52076	46228	0.529744	0.924709	0.924705	0.925222
168	40	0.322303	50524	47780	0.513957	0.938823	0.939484	0.940079
168	41	0.329109	49982	48322	0.508443	0.942394	0.943037	0.942623
168	42	0.331205	50135	48169	0.510000	0.941771	0.942414	0.941027
168	43	0.329109	50302	48002	0.511698	0.941607	0.941853	0.941893
168	44	0.322303	49928	48376	0.507894	0.942925	0.944007	0.943745
168	45	0.310588	51018	47286	0.518982	0.936488	0.935358	0.935324
168	46	0.293273	51059	47245	0.519399	0.938429	0.938365	0.938030
168	47	0.268966	52816	45488	0.537272	0.927705	0.926223	0.925205
168	48	0.234249	58175	40129	0.591787	0.890819	0.890319	0.889731
168	49	0.171941	89410	8894	0.909526	0.319848	0.314986	0.330354
180	0	0.567607	57397	40907	0.583872	0.870453	0.856699	0.862938
180	1	0.567607	57538	40766	0.585307	0.871752	0.858837	0.866632
180	2	0.436940	51673	46631	0.525645	0.937818	0.933526	0.935739
180	3	0.436940	51715	46589	0.526072	0.939520	0.935794	0.937545
180	4	0.436769	49772	48532	0.506307	0.944230	0.943745	0.944656
180	5	0.436769	49853	48451	0.507131	0.943834	0.944382	0.943774

180	6	0.234249	66591	31713	0.677399	0.757323	0.767458	0.764858
180	7	0.293273	51022	47282	0.519023	0.939557	0.938735	0.939869
180	8	0.322303	50216	48088	0.510824	0.941516	0.942473	0.941997
180	9	0.331344	49546	48758	0.504008	0.944449	0.944531	0.944053
180	10	0.322303	50611	47693	0.514842	0.938234	0.939507	0.938682
180	11	0.293273	54663	43641	0.556061	0.923487	0.925600	0.924261
180	12	0.234248	95744	2560	0.973958	0.148588	0.155194	0.145152
180	13	0.234248	75738	22566	0.770447	0.688189	0.694001	0.683783
180	14	0.293273	54706	43598	0.556498	0.923044	0.924426	0.923363
180	15	0.322303	50397	47907	0.512665	0.939938	0.940627	0.940218
180	16	0.331344	49821	48483	0.506805	0.942921	0.942747	0.942720
180	17	0.322303	50195	48109	0.510610	0.941252	0.941582	0.941588
180	18	0.293273	51248	47056	0.521322	0.936519	0.935205	0.935829
180	19	0.234249	70869	27435	0.720917	0.694150	0.699957	0.695740
180	20	0.171941	97463	841	0.991445	0.056185	0.054700	0.050865
180	21	0.234249	64576	33728	0.656901	0.801791	0.806270	0.811928
180	22	0.268966	55626	42678	0.565857	0.900939	0.903415	0.904974
180	23	0.293273	51508	46796	0.523966	0.935476	0.937182	0.936094
180	24	0.310588	50413	47891	0.512828	0.938959	0.938019	0.938635
180	25	0.322303	50397	47907	0.512665	0.939458	0.938829	0.940654
180	26	0.329109	50615	47689	0.514882	0.940169	0.939576	0.939081
180	27	0.331205	50167	48137	0.510325	0.941476	0.941285	0.941384
180	28	0.329109	50429	47875	0.512990	0.941148	0.941640	0.941237
180	29	0.322303	50449	47855	0.513194	0.940256	0.940354	0.940924
180	30	0.310588	51765	46539	0.526581	0.935263	0.935431	0.934540
180	31	0.293273	53016	45288	0.539307	0.931266	0.932607	0.932024
180	32	0.268966	58496	39808	0.595052	0.897652	0.900229	0.900538
180	33	0.234248	76518	21786	0.778381	0.647219	0.663224	0.655709
180	34	0.171941	97100	1204	0.987752	0.073024	0.075457	0.071525
180	35	0.171941	97565	739	0.992482	0.050639	0.045537	0.046525
180	36	0.234248	91469	6835	0.930471	0.296328	0.308739	0.300553
180	37	0.268966	64474	33830	0.655863	0.859187	0.860164	0.860519
180	38	0.293273	56349	41955	0.573212	0.889282	0.888403	0.890556
180	39	0.310588	51564	46740	0.524536	0.932814	0.931461	0.934221
180	40	0.322303	50325	47979	0.511932	0.940674	0.939981	0.939341
180	41	0.329109	50354	47950	0.512227	0.938880	0.939681	0.939303
180	42	0.331205	49988	48316	0.508504	0.942503	0.943081	0.942931
180	43	0.329109	49958	48346	0.508199	0.942112	0.942229	0.942078
180	44	0.322303	49982	48322	0.508443	0.942482	0.943100	0.942404
180	45	0.310588	51004	47300	0.518840	0.936603	0.937402	0.936355
180	46	0.293273	51533	46771	0.524221	0.932446	0.934154	0.935177
180	47	0.268966	52477	45827	0.533824	0.927975	0.927985	0.923923
180	48	0.234249	55507	42797	0.564646	0.914786	0.916655	0.916499
180	49	0.171941	91757	6547	0.933400	0.262044	0.266012	0.265668

189	0	0.567607	57409	40895	0.583995	0.869071	0.859218	0.865580
189	1	0.567607	57188	41116	0.581746	0.868307	0.860210	0.866573
189	2	0.436940	51343	46961	0.522288	0.939932	0.936147	0.938519
189	3	0.436940	51367	46937	0.522532	0.939744	0.935395	0.937770
189	4	0.436769	50197	48107	0.510630	0.943169	0.942957	0.943016
189	5	0.436769	49991	48313	0.508535	0.944763	0.943353	0.944093
189	6	0.234249	64971	33333	0.660919	0.798692	0.796504	0.797428
189	7	0.293273	50746	47558	0.516215	0.938687	0.938666	0.940049
189	8	0.322303	50070	48234	0.509338	0.942945	0.943377	0.943331
189	9	0.331344	49588	48716	0.504435	0.943066	0.943218	0.942563
189	10	0.322303	50751	47553	0.516266	0.939340	0.938732	0.939274
189	11	0.293273	55287	43017	0.562408	0.918462	0.921170	0.921089
189	12	0.234248	84591	13713	0.860504	0.482718	0.487116	0.471784
189	13	0.234248	76472	21832	0.777913	0.676725	0.681373	0.673458
189	14	0.293273	55788	42516	0.567505	0.914383	0.913499	0.914128
189	15	0.322303	49948	48356	0.508097	0.943461	0.943371	0.943569
189	16	0.331344	49862	48442	0.507222	0.944170	0.943981	0.944126
189	17	0.322303	50177	48127	0.510427	0.942376	0.942682	0.943096
189	18	0.293273	51003	47301	0.518829	0.937681	0.937078	0.936955
189	19	0.234249	60679	37625	0.617259	0.856763	0.863027	0.860034
189	20	0.171941	97733	571	0.994191	0.040670	0.035947	0.037722
189	21	0.234249	65976	32328	0.671143	0.785226	0.784595	0.782680
189	22	0.268966	55195	43109	0.561473	0.907082	0.906255	0.906924
189	23	0.293273	51053	47251	0.519338	0.938615	0.939379	0.937845
189	24	0.310588	51308	46996	0.521932	0.936757	0.937028	0.935114
189	25	0.322303	50850	47454	0.517273	0.937328	0.936589	0.936840
189	26	0.329109	50397	47907	0.512665	0.939593	0.939851	0.939484
189	27	0.331205	50066	48238	0.509298	0.942988	0.942894	0.942507
189	28	0.329109	50072	48232	0.509359	0.941767	0.941861	0.941306
189	29	0.322303	50724	47580	0.515991	0.938600	0.939143	0.938840
189	30	0.310588	52214	46090	0.531148	0.932195	0.933451	0.932573
189	31	0.293273	53585	44719	0.545095	0.927464	0.927305	0.927322
189	32	0.268966	63438	34866	0.645325	0.842864	0.844257	0.841666
189	33	0.234248	91259	7045	0.928335	0.286569	0.302907	0.297477
189	34	0.171941	98008	296	0.996989	0.023440	0.024972	0.028521
189	35	0.171941	98001	303	0.996918	0.024043	0.025117	0.027158
189	36	0.234248	97964	340	0.996541	0.027144	0.025063	0.027877
189	37	0.268966	66555	31749	0.677032	0.821950	0.827248	0.821801
189	38	0.293273	55114	43190	0.560649	0.901779	0.901960	0.901426
189	39	0.310588	51359	46945	0.522451	0.931723	0.930404	0.931343
189	40	0.322303	51341	46963	0.522268	0.933813	0.932578	0.933358
189	41	0.329109	50074	48230	0.509379	0.941467	0.941248	0.940750
189	42	0.331205	50145	48159	0.510101	0.942701	0.942975	0.943564
189	43	0.329109	50207	48097	0.510732	0.940710	0.940985	0.939820

189	44	0.322303	50248	48056	0.511149	0.942170	0.942122	0.942214
189	45	0.310588	50694	47610	0.515686	0.938974	0.939847	0.939791
189	46	0.293273	50705	47599	0.515798	0.940940	0.941789	0.941367
189	47	0.268966	52568	45736	0.534749	0.924871	0.924806	0.924434
189	48	0.234249	58424	39880	0.594320	0.882829	0.885958	0.889611
189	49	0.171941	94345	3959	0.959727	0.182368	0.184122	0.189314
190	0	0.567607	57621	40683	0.586151	0.865743	0.859221	0.863065
190	1	0.567607	57433	40871	0.584239	0.869224	0.860157	0.864624
190	2	0.436940	51763	46541	0.526560	0.938371	0.934168	0.936913
190	3	0.436940	51092	47212	0.519735	0.940544	0.936734	0.939498
190	4	0.436769	50123	48181	0.509878	0.943004	0.944114	0.944549
190	5	0.436769	49948	48356	0.508097	0.944355	0.943341	0.943148
190	6	0.234249	56265	42039	0.572357	0.907636	0.906887	0.905052
190	7	0.293273	50789	47515	0.516652	0.941905	0.941730	0.942701
190	8	0.322303	50048	48256	0.509115	0.942168	0.941658	0.941257
190	9	0.331344	49864	48440	0.507243	0.943981	0.943226	0.943891
190	10	0.322303	50283	48021	0.511505	0.941374	0.942764	0.942333
190	11	0.293273	58855	39449	0.598704	0.887020	0.882144	0.887150
190	12	0.234248	86142	12162	0.876282	0.442271	0.452154	0.442117
190	13	0.234248	84634	13670	0.860942	0.482103	0.487408	0.483747
190	14	0.293273	53367	44937	0.542877	0.934901	0.934619	0.932448
190	15	0.322303	49667	48637	0.505239	0.943812	0.943378	0.943765
190	16	0.331344	50138	48166	0.510030	0.941346	0.941795	0.941238
190	17	0.322303	50468	47836	0.513387	0.938304	0.938773	0.938192
190	18	0.293273	50962	47342	0.518412	0.937078	0.937150	0.938020
190	19	0.234249	70071	28233	0.712799	0.726496	0.733309	0.736462
190	20	0.171941	90668	7636	0.922323	0.294518	0.295335	0.291311
190	21	0.234249	65526	32778	0.666565	0.798706	0.799096	0.794823
190	22	0.268966	53587	44717	0.545115	0.921729	0.923583	0.923898
190	23	0.293273	51284	47020	0.521688	0.937490	0.936346	0.937258
190	24	0.310588	50897	47407	0.517751	0.936890	0.936941	0.936834
190	25	0.322303	50139	48165	0.510040	0.941685	0.941560	0.940587
190	26	0.329109	50101	48203	0.509654	0.941732	0.941728	0.940972
190	27	0.331205	49732	48572	0.505900	0.943485	0.943353	0.942399
190	28	0.329109	50140	48164	0.510050	0.942819	0.942366	0.942470
190	29	0.322303	50600	47704	0.514730	0.939304	0.939234	0.939103
190	30	0.310588	50957	47347	0.518361	0.939385	0.939918	0.940182
190	31	0.293273	53280	45024	0.541992	0.928308	0.928978	0.928562
190	32	0.268966	58737	39567	0.597504	0.894313	0.895526	0.896648
190	33	0.234248	77632	20672	0.789714	0.651113	0.648706	0.649741
190	34	0.171941	92946	5358	0.945496	0.234448	0.232282	0.225661
190	35	0.171941	97431	873	0.991119	0.057259	0.061611	0.059185
190	36	0.234248	89230	9074	0.907695	0.351952	0.367728	0.360864
190	37	0.268966	63366	34938	0.644592	0.862401	0.865815	0.859950

190	38	0.293273	53252	45052	0.541707	0.922628	0.924170	0.922540
190	39	0.310588	51480	46824	0.523682	0.931618	0.932383	0.930986
190	40	0.322303	50613	47691	0.514862	0.939317	0.937944	0.938264
190	41	0.329109	50014	48290	0.508769	0.941173	0.940547	0.941616
190	42	0.331205	50173	48131	0.510386	0.940061	0.940481	0.940612
190	43	0.329109	50265	48039	0.511322	0.942493	0.942816	0.942656
190	44	0.322303	50103	48201	0.509674	0.941954	0.941959	0.941299
190	45	0.310588	51164	47140	0.520467	0.936960	0.935877	0.936820
190	46	0.293273	51162	47142	0.520447	0.937357	0.936970	0.936337
190	47	0.268966	52510	45794	0.534159	0.929325	0.930387	0.929318
190	48	0.234249	59991	38313	0.610260	0.865629	0.866577	0.864026
190	49	0.171941	85356	12948	0.868286	0.441044	0.439206	0.442251
198	0	0.567607	57339	40965	0.583282	0.869835	0.859226	0.864333
198	1	0.567607	57287	41017	0.582753	0.868358	0.858470	0.863466
198	2	0.436940	51479	46825	0.523671	0.940787	0.936513	0.939332
198	3	0.436940	51470	46834	0.523580	0.940153	0.937119	0.938186
198	4	0.436769	50493	47811	0.513641	0.942443	0.942673	0.942710
198	5	0.436769	49764	48540	0.506226	0.943912	0.944813	0.944347
198	6	0.234249	53065	45239	0.539805	0.933451	0.933969	0.931983
198	7	0.293273	51002	47302	0.518819	0.939674	0.938406	0.939111
198	8	0.322303	49909	48395	0.507701	0.943063	0.942946	0.942479
198	9	0.331344	49940	48364	0.508016	0.941528	0.941539	0.942075
198	10	0.322303	50766	47538	0.516418	0.938856	0.938024	0.938305
198	11	0.293273	56471	41833	0.574453	0.906122	0.908229	0.904759
198	12	0.234248	90292	8012	0.918498	0.326066	0.323567	0.327769
198	13	0.234248	72782	25522	0.740377	0.754761	0.760405	0.759630
198	14	0.293273	54642	43662	0.555847	0.919967	0.922247	0.923543
198	15	0.322303	50315	47989	0.511831	0.941652	0.941390	0.941667
198	16	0.331344	49954	48350	0.508158	0.943944	0.943863	0.943603
198	17	0.322303	50521	47783	0.513926	0.939604	0.940173	0.939574
198	18	0.293273	50937	47367	0.518158	0.939412	0.938268	0.938112
198	19	0.234249	60315	37989	0.613556	0.863674	0.868107	0.866059
198	20	0.171941	96086	2218	0.977437	0.123989	0.117943	0.115887
198	21	0.234249	66289	32015	0.674327	0.783229	0.782128	0.782105
198	22	0.268966	53678	44626	0.546041	0.921693	0.920720	0.922117
198	23	0.293273	51668	46636	0.525594	0.934308	0.934545	0.934035
198	24	0.310588	51322	46982	0.522074	0.932366	0.932240	0.932402
198	25	0.322303	50281	48023	0.511485	0.938541	0.939157	0.939486
198	26	0.329109	50471	47833	0.513418	0.937671	0.938848	0.937842
198	27	0.331205	49754	48550	0.506124	0.942700	0.942338	0.941935
198	28	0.329109	50295	48009	0.511627	0.941622	0.941951	0.941399
198	29	0.322303	50730	47574	0.516052	0.939126	0.938746	0.939040
198	30	0.310588	51100	47204	0.519816	0.939470	0.939701	0.940167
198	31	0.293273	52979	45325	0.538930	0.931508	0.933721	0.931858

198	32	0.268966	58455	39849	0.594635	0.898338	0.898988	0.898959
198	33	0.234248	74889	23415	0.761810	0.675299	0.683203	0.681555
198	34	0.171941	95783	2521	0.974355	0.140310	0.132890	0.138622
198	35	0.171941	95219	3085	0.968618	0.147254	0.149883	0.152060
198	36	0.234248	85382	12922	0.868551	0.454936	0.453619	0.452605
198	37	0.268966	68038	30266	0.692118	0.785385	0.789911	0.783877
198	38	0.293273	53544	44760	0.544678	0.917413	0.917123	0.916793
198	39	0.310588	51172	47132	0.520549	0.933807	0.934444	0.934528
198	40	0.322303	50532	47772	0.514038	0.937948	0.937788	0.938889
198	41	0.329109	50702	47602	0.515767	0.934775	0.935421	0.935203
198	42	0.331205	49875	48429	0.507355	0.942818	0.941935	0.942057
198	43	0.329109	49849	48455	0.507090	0.941804	0.942102	0.942501
198	44	0.322303	50396	47908	0.512655	0.939072	0.940247	0.939528
198	45	0.310588	50783	47521	0.516591	0.939707	0.940294	0.939766
198	46	0.293273	51089	47215	0.519704	0.937935	0.937549	0.937219
198	47	0.268966	52118	46186	0.530172	0.929159	0.930266	0.929465
198	48	0.234249	57857	40447	0.588552	0.884805	0.885921	0.886491
198	49	0.171941	87791	10513	0.893056	0.353712	0.352562	0.354568