JOÃO PAULO FERNANDES DE CERQUEIRA CÉSAR

ANÁLISE DOS ALGORITMOS DE PERFIL II DO PROJETO eSTREAM PARA CRIPTOGRAFIA DE IMAGENS

LAVRAS – MG

2018

JOÃO PAULO FERNANDES DE CERQUEIRA CÉSAR

ANÁLISE DOS ALGORITMOS DE PERFIL II DO PROJETO eSTREAM PARA CRIPTOGRAFIA DE IMAGENS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação para obtenção do título de Mestre.

Prof. Dr. Wilian Soares Lacerda (DCC/UFLA)
Orientador

Prof. Dr. Bruno de Abreu Silva (DCC/UFLA)
Coorientador

LAVRAS – MG
2018

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Cerqueira César, João Paulo Fernandes de.
Análise dos Algoritmos de Perfil II do Projeto eSTREAM para Criptografia de Imagens / João Paulo Fernandes de Cerqueira César. - 2017.
106 p. : il.

Orientador(a): Wilian Soares Lacerda.
Coorientador(a): Bruno de Abreu Silva.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2017.
Bibliografia.

1. FPGA. 2. Criptografia de Imagens. 3. eSTREAM. I. Lacerda, Wilian Soares. II. Silva, Bruno de Abreu. III. Título.

JOÃO PAULO FERNANDES DE CERQUEIRA CÉSAR

ANÁLISE DOS ALGORITMOS DE PERFIL II DO PROJETO eSTREAM PARA CRIPTOGRAFIA DE IMAGENS
ANALYSIS OF eSTREAM PROFILE II CIPHERS FOR IMAGE ENCRYPTION

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação para obtenção do título de Mestre.

APROVADA em 21 de Dezembro de 2017.

Prof. Dr. Wilian Soares Lacerda	DCC/UFLA
Prof. Dr. Bruno de Abreu Silva	DCC/UFLA
Prof. Dr. Bruno Henrique Groenner Barbosa	DEG/UFLA
Prof. Dr. Otávio de Souza Martins Gomes	IFMG

Prof. Dr. Wilian Soares Lacerda (DCC/UFLA)
Orientador

Prof. Dr. Bruno de Abreu Silva (DCC/UFLA)
Co-Orientador

LAVRAS – MG
2018

RESUMO

Segurança e privacidade sempre foram alvo de pesquisas. Atualmente com a popularização dos meios de comunicação em massa, como a internet, esse assunto se torna ainda mais fundamental. A comunicação nos dias de hoje não é realizada apenas pela troca de arquivos de texto ou áudio, mas, também, pela troca de imagens digitais. Sistemas de criptografia vêm sendo constantemente aprimorados e padronizados para prover segurança e privacidade, assim como algoritmos especialistas em cifra de imagens. Apesar de serem traduzidas em dados binários assim como os textos, as imagens possuem características particulares que impedem o uso de sistemas de criptografia populares, como o RSA e AES. A crescente utilização de câmeras em sistemas computacionais, como, por exemplo, em carros autônomos e sistemas de vigilância, fomenta a necessidade de técnicas para a transmissão segura de imagens. Além disso, aplicações que trabalham com tomadas de decisão em tempo real demandam baixa latência, tornando a implementação dessas em *hardware* desejável. Sistemas de criptografia por fluxo são compactos e de simples implementação, para promover seu desenvolvimento o ECRYPT *(European Network of Excellence for Cryptology)* organizou o projeto eSTREAM, resultando em um portfólio de cifras de fluxo validadas para implementações em *software* e *hardware*. O desenvolvimento de sistemas criptográficos em *hardware* é vantajoso devido à alta vazão provida por esse paradigma, além disso, plataformas reconfiguráveis como os *Field Programmable Gate Arrays (FPGAs)* oferecem flexibilidade e baixo custo no desenvolvimento de novas soluções. O presente projeto tem como objetivo a implementação em FPGA dos sistemas de criptografia de Perfil II do projeto eSTREAM e análise da qualidade de suas cifras quando aplicadas na segurança de imagens digitais.

Palavras-chave: FPGA. Criptografia de Imagens. eSTREAM.

ABSTRACT

Security and privacy have always been the subject of research. Nowadays with the popularization of the mass media, like the Internet, this subject becomes even more fundamental. Communication today is not only accomplished by exchanging text or audio files, but also by exchanging digital images. Encryption systems have been constantly enhanced and standardized to provide security and privacy, as well as algorithms for ciphering images. Although they are translated into binary data as well as texts, the images have particular characteristics that prevent the use of popular cryptographic systems, such as RSA and AES. The increasing use of cameras in computer systems, such as in autonomous cars and surveillance systems, fosters the need for techniques for secure image transmission. In addition, applications that work with real-time decision-making require low latency, making such hardware implementation desirable. Stream ciphers systems are compact and simple to implement, to promote their development ECRYPT (European Network of Excellence for Cryptology) has organized the eSTREAM project, resulting in a portfolio of validated stream ciphers for software and hardware implementations. The development of cryptographic systems in hardware is advantageous because of the high throughput provided by this paradigm, in addition, reconfigurable platforms such as Field Programmable Gate Arrays (FPGAs) offer flexibility and low cost in the development of new solutions. The present project aims at the implementation in FPGA of the cryptography systems of Profile II of the project eSTREAM and analysis of the quality of its figures when applied in the security of digital images.

Keywords: FPGA. Image Encryption. eSTREAM.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Atualmente, a criptografia é usada como técnica de transformação de dados segundo um código ou algoritmo para que eles se tornem ininteligíveis a não ser para quem possui a chave do código (TERADA, 2008). Existem várias técnicas e algoritmos para implementação da transferência segura de dados, bem como órgãos governamentais e instituições para controle e normatização de padrões na segurança de dados (STALLINGS, 2016). Com a popularização da internet, tornou-se comum enviar arquivos para outros usuários dessa rede, seja por meio de correio eletrônico, redes sociais ou mensageiros instantâneos. Entre esses tipos de arquivos compartilhados estão as imagens, que podem conter informações privadas e sigilosas. Outro cenário onde a transmissão de imagens é frequente e sua segurança e privacidade são de extrema importância são nos sistemas de vigilância e monitoramento. Mais frequentemente, também, observa-se a utilização de câmeras em sistemas automotivos autônomos com tomadas de decisão em tempo real. Para que essas imagens possam trafegar por um meio não seguro, como é a internet, de forma segura, utiliza-se dos conceitos da criptografia.

1.1 Motivação

O desenvolvimento de um sistema para criptografia de imagens em tempo real tem grande potencial de aplicação atualmente, como exemplo pode-se citar a comunicação segura por meio de vídeo conferências e troca segura de arquivos de imagens em comunicações com restrições de tempo (operações militares, aplicações médicas, monitoração de florestas, sistemas automotivos autônomos e sistemas de vigilância). Para que esse sistema seja funcional deve-se obter um baixo tempo de processamento, de modo a contribuir o mínimo possível na latência da comunicação. Nesse cenário torna-se interessante a implementação de um sistema para criptografia de imagens em tempo real utilizando plataformas de *hardware* reconfiguráveis, como os *Field Programmable Gate Arrays* (FPGAs). A implementação de algoritmos diretamente em *hardware* proporciona uma maior vazão de dados processados se comparados a *softwares* executados em um computador.

Com o crescimento dos investimentos nacionais na área de microeletrônica e na fabricação de circuitos integrados, é importante a realização de projetos de pesquisa e, principalmente, o seu desenvolvimento na área de prototipação de dispositivos reconfiguráveis, tendo em vista que eles participam do ciclo de projeto de um circuito integrado de aplicação específica *(Application Specific Integrated Circuits - ASIC)* na fase validação da ideia proposta. O FPGA é

uma alternativa no desenvolvimento ágil do processo de prototipagem devido ao baixo custo e tempo de desenvolvimento demandado (KUON; TESSIER; ROSE, 2007). Portanto, o FPGA é flexível ao ponto de permitir alterações de projeto em pouco tempo em relação ao tempo de planejar e construir um novo circuito ASIC aplicando novas mudanças.

O projeto eSTREAM foi anunciado em 2004 pelo ECRYPT *(European Network of Excellence for Cryptology)* com objetivo de identificar novos algoritmos de criptografia simétricos para futura utilização (RIJMEN, 2010). O projeto é dividido em dois perfis de implementação, o primeiro (Perfil I) trata de algoritmos com aplicação em *software* capazes de suprir demandas de alta vazão de dados, já o segundo (Perfil II) é aplicado em *hardware* com limitação de recursos. Ao final do ano de 2008 foram escolhidos os finalistas de cada perfil, que passaram então a compor o portfólio do projeto. Até o momento não foram encontrados trabalhos publicados que realizaram a análise da qualidade desses algoritmos quando aplicados em imagens digitais.

1.2 Objetivos

O objetivo desse projeto é a implementação em *software* e em *hardware* reconfigurável, mais especificamente em Matlab e FPGA respectivamente, dos algoritmos de Perfil II do portfólio atual do projeto eSTREAM (ECRYPT II, 2012b), são eles: Grain v1, MICKEY 2.0 e Trivium. Após a implementação, os algoritmos serão analisados de acordo com os métodos descritos na Seção 4.4 com objetivo de avaliar a viabilidade de utilização destes no processo de criptografia de imagens para futuro desenvolvimento de um *hardware* para transmissão segura de imagens estáticas (fotos) e/ou dinâmicas (vídeo), em tempo real ou não.

Não faz parte do objetivo do projeto a análise da qualidade da implementação em FPGA destes sistemas, mas sim a análise da qualidade oferecida por eles quando aplicados no processo de criptografia de imagens. Apesar disso, como forma de validar os sistemas para operação em um cenário com alta vazão de dados e em tempo real, como é o caso de aplicações de vídeo em tempo real, no Capítulo 5 são apresentadas algumas considerações sobre o desempenho das implementações em FPGA utilizadas neste trabalho.

1.3 Organização

Este documento é dividido da seguinte maneira: o referencial teórico, apresentado no Capítulo 2, aborda os principais conceitos imprescindíveis ao projeto, como elementos de ima-

gens digitais, conceitos de criptografia e números aleatórios, o projeto eSTREAM e conceitos de FPGA. As conclusões da revisão sistemática do estado da arte em algoritmos de criptografia de imagens desenvolvidos em *hardware* são apresentadas no Capítulo 3 enquanto o Capítulo 4 apresenta a metodologia de pesquisa utilizada no projeto. No Capítulo 5 são apresentados os resultados obtidos no projeto, enquanto que no Capítulo 6 estes sãos discutidos. Finalmente é apresentada a bibliografia consultada.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados conceitos fundamentais para o desenvolvimento do projeto, este material foi organizado durante a fase de pesquisa bibliográfica. O capítulo apresenta, fundamentos de imagens digitais na Seção 2.1, conceitos de criptografia na Seção 2.2, conceitos de aleatoriedade e geradores de números pseudoaleatórios na Seção 2.3, o projeto eSTREAM com ênfase nos algoritmos da Fase II é mostrado na Seção 2.4, e por fim, na Seção 2.5 são demonstrados os FPGAs e o processo de descrição de um *hardware* reconfigurável.

2.1 Elementos de Imagens Digitais

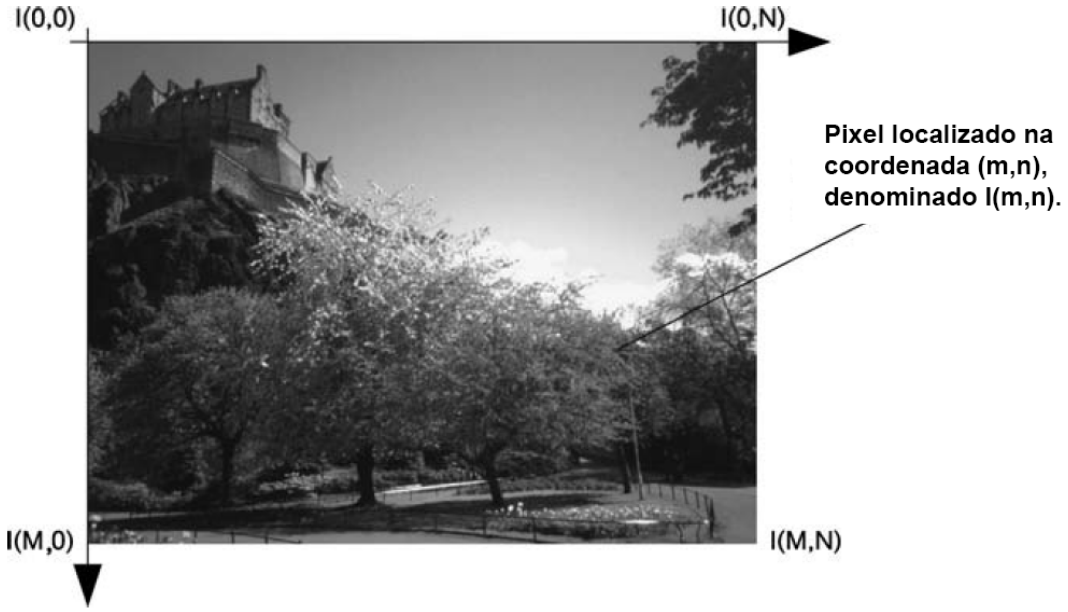
Imagens digitais estão atualmente presentes no cotidiano de inúmeras pessoas, por meio da exibição de fotografias, filmes ou documentos em um computador, *smartphone* ou televisões digitais. Essa seção têm como objetivo familiarizar o leitor com conceitos básicos de imagens em sistemas computacionais.

2.1.1 Representação

Segundo Solomon (2011, p. 1, tradução nossa) "uma imagem digital pode ser considerada como uma representação de dados discretos contendo informações tanto espaciais *(layout)*, quanto de intensidade (cor) [1]". Uma imagem digital I é derivada de um processo chamado discretização, onde os sinais contínuos provenientes da cena capturada são amostrados para sinais discretos no sensor da câmera. As informações espaciais de uma imagem representam, no caso de uma imagem de duas dimensões, a resposta de um sensor exposto a uma fonte luminosa, onde cada célula fotossensível é exibida por meio de representações em um plano cartesiano 2D. Cada elemento individual de uma imagem, i.e. $I(m,n)$, onde m e n são as coordenadas do plano cartesiano, é denominado *pixel*. A Figura 2.1 mostra uma imagem de dimensões *MxN* representada em um plano cartesiano, com destaque para um *pixel* localizado na posição (m,n) (SOLOMON, 2011).

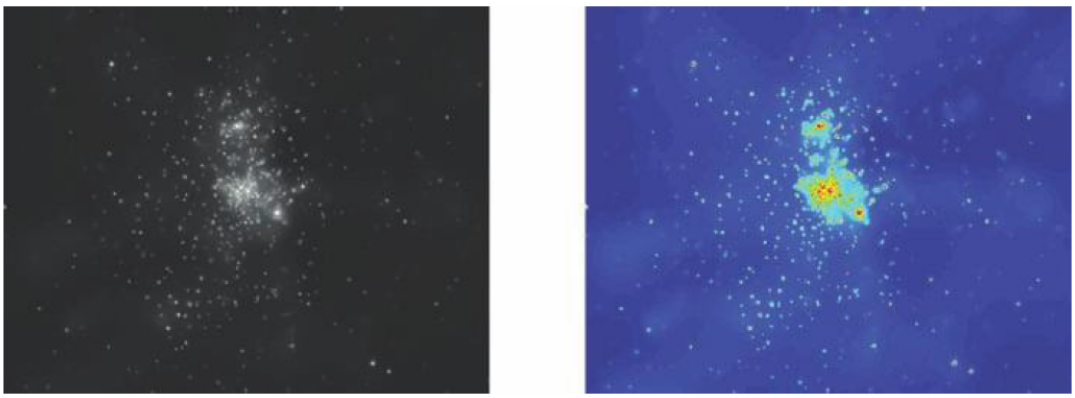
[1] *A digital image can be considered as a discrete representation of data possessing both spatial (layout) and intensity (colour) information*

Figura 2.1 – Plano cartesiano representando uma imagem 2D de dimensões *MxN*.



Fonte: Adaptado de (SOLOMON, 2011)

Figura 2.2 – Representação de cores de uma imagem.



Legenda: Imagem representada em escala de cinza (esquerda) e a mesma imagem representada por meio de mapeamentos de cor falsos (direita).
Fonte: (SOLOMON, 2011)

Uma imagem contém um ou mais canais que representam a intensidade ou a cor para cada um dos *pixels*. De uma maneira simplificada, uma imagem pode ser representada por um mapeamento de um único valor numérico para cada *pixel*, como, por exemplo, a intensidade, para uma representação de valores em uma escala de cinza, ou seja, entre valores presentes no intervalo de zero (preto) a um valor máximo (branco). Pode-se também representar uma imagem por meio de mapeamentos falsos de cor. Neste tipo de operação, uma imagem não colorida é exibida utilizando um intervalo de valores que representem *pixels* coloridos. Este tipo de representação é útil pois o ser humano somente consegue distinguir entre aproximadamente

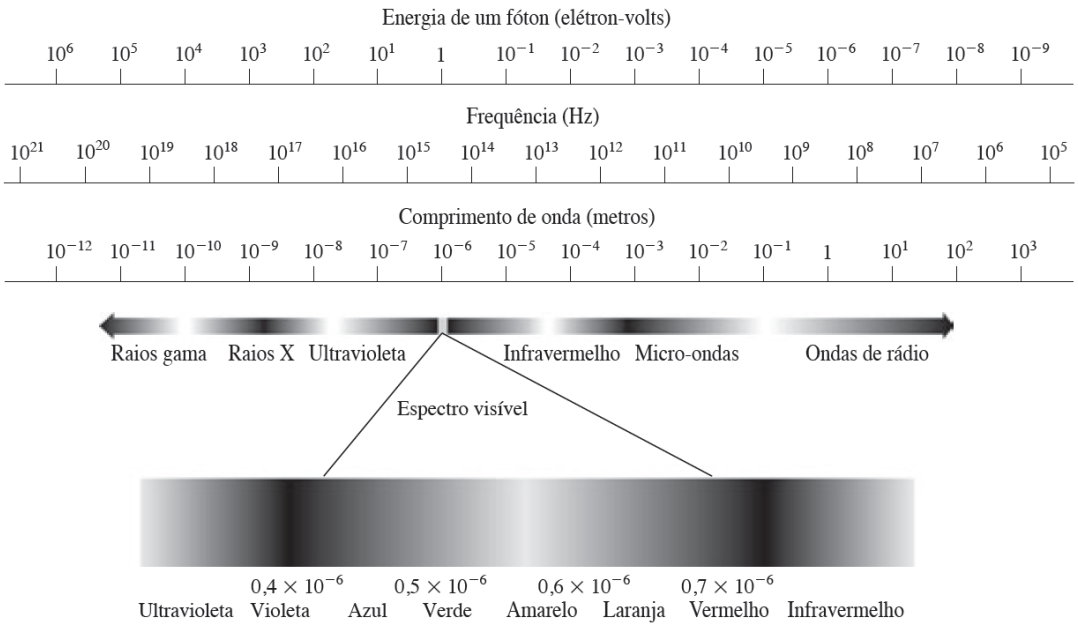
quarenta tons de cinza, dessa forma, o mapeamento poderia ser definido como um intervalo entre a cor azul escura, representando valores baixos, e a cor vermelho escuro para valores altos. A Figura 2.2 mostra uma imagem exemplificando ambos métodos de mapeamento.

Por fim, uma imagem também pode ser representada por cores verdadeiras, onde por meio de uma 3-upla o espectro de cores pode ser representado. Essa 3-upla trata-se da codificação RGB, em que cada *pixel* é codificado por meio da combinação de valores para cada um dos componentes: Vermelho (***R**ed*), Verde (***G**reen*) e Azul (***B**lue*). A cor do *pixel*, portanto, é resultado de uma combinação linear dos valores da 3-upla. Outra forma comum de representação de imagens coloridas é a *Hue, Saturation and Value* (HSV). Nessa forma de representação, o valor da intensidade V de uma cor é decomposto da informação cromática contida nos componentes H e S (SOLOMON, 2011).

2.1.1.1 Espectro Eletromagnético

Em 1666, Isaac Newton observou que quando um feixe de luz atravessa um prisma, o feixe resultante não consiste da mesma luz branca, mas sim de um espectro de cores variando da cor violeta até a cor vermelha. A Figura 2.3 mostra um esquemático do espectro eletromagnético, nela é possível observar que a área correspondente ao espectro visível é muito pequeno quando comparado com a totalidade do espectro. O espectro eletromagnético pode ser expressado por meio do tamanho de onda, frequência ou energia (GONZALEZ; WOODS, 2002).

Figura 2.3 – O espectro eletromagnético.



Fonte: (GONZALEZ; WOODS, 2009)

A análise de imagens explorando diversas fontes de iluminação provenientes do espectro eletromagnético é utilizada para explorar áreas de aplicação distintas. Abaixo são listadas algumas aplicações para cada faixa do espectro (GONZALEZ; WOODS, 2002):

- **Raios Gama:** aplicações de medicina nuclear e observações astronômicas;
- **Raios X:** diagnósticos médicos, indústria e outras áreas, como astronomia;
- **Ultravioleta:** inspeção industrial, microscopia, *lasers*, imagens biológicas e astronomia.
- **Espectro Visível e Infravermelho:** sensoreamento remoto, previsão e observação do clima, inspeção e automação industrial e agrícola. O Quadro 2.1 mostra as chamadas "bandas temáticas" no satélite LANDSAT da NASA, que é utilizado para obter e transmitir imagens da terra a partir do espaço, nele é possível observar que cada comprimento de onda tem uma função específica ao capturar uma imagem.
- **Micro-ondas:** majoritariamente aplicações em radares.
- **Ondas de Rádio:** diagnósticos médicos e observações astronômicas.

Quadro 2.1 – Bandas temáticas no satélite LANDSAT.

Número da banda	**Nome**	**Comprimento de onda** (μm)	**Características e utilizações**
1	Azul visível	0,45-0,52	Máxima penetração na água
2	Verde visível	0,52-0,60	Bom para mensuração do vigor de plantas
3	Vermelho visível	0,63-0,69	Discriminação de vegetação
4	Infravermelho próximo	0,76-0,90	Mapeamento de biomassa e linha costeira
5	Infravermelho médio	1,55-1,75	Conteúdo de umidade do solo e vegetação
6	Infravermelho termal	10,4-12,5	Umidade do solo, mapeamento térmico
7	Infravermelho médio	2,08-2,35	Mapeamento mineral

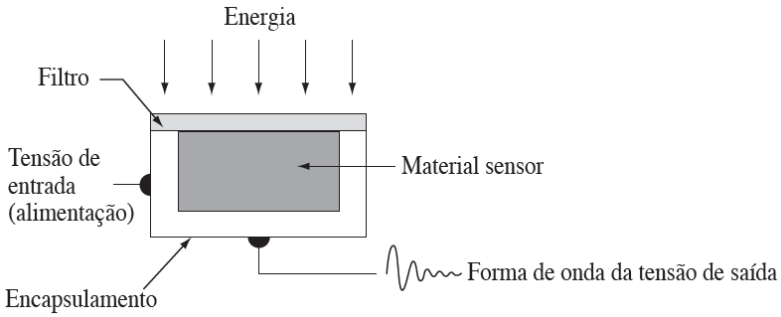
Fonte: (GONZALEZ; WOODS, 2009)

2.1.1.2 Sensores e Aquisição de Imagens

Para que uma cena do mundo real seja representada digitalmente, é necessária a utilização de algum tipo de sensor que converta a energia produzida pelos objetos da cena em pulsos elétricos, que posteriormente serão interpretados de alguma maneira e representarão digitalmente a imagem. Diversos tipos de sensores estão disponíveis, desde modelos simples com

um único sensor, como mostrado na Figura 2.4, por exemplo um fotodiodo, até modelos mais complexos como aqueles dispostos em um arranjo matricial (GONZALEZ; WOODS, 2002).

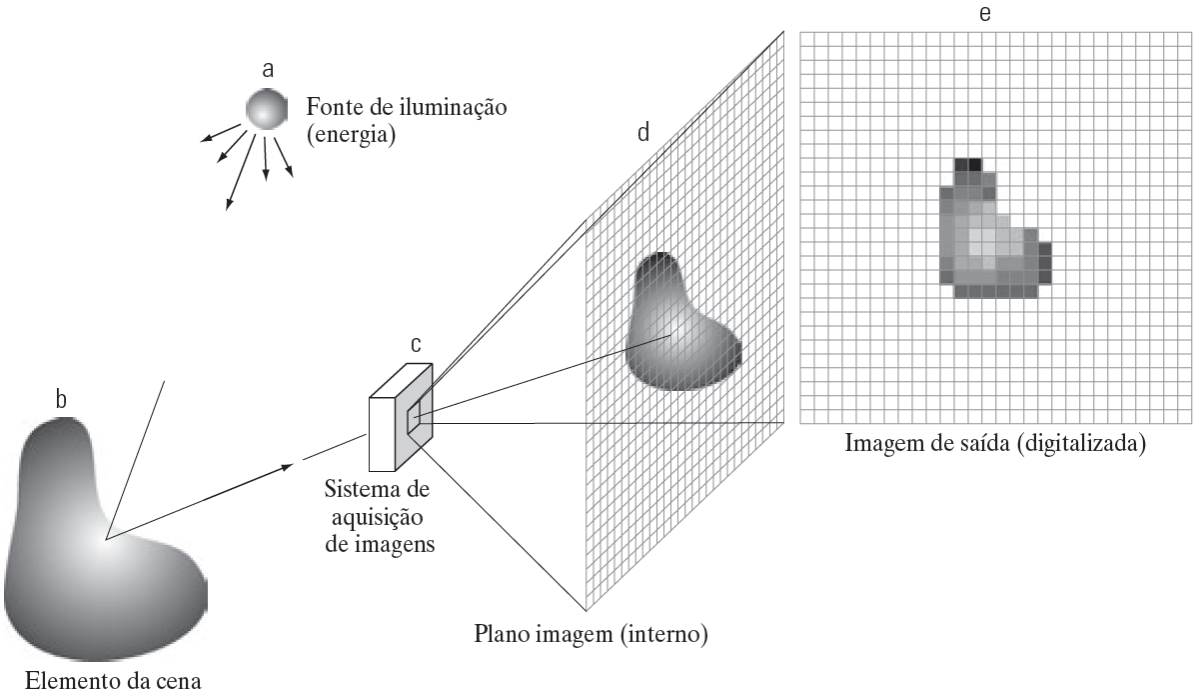
Figura 2.4 – Sensor único para captura de imagem.



Fonte: (GONZALEZ; WOODS, 2009)

Câmeras digitais em sua maioria possuem sensores em arranjo de matriz. Em um sistema com esse tipo de sensor, a resposta de cada um dos sensores que compõe a matriz é proporcional à integral da energia luminosa projetada em sua superfície.

Figura 2.5 – Exemplo do processo de aquisição de imagens por matriz de sensores.



Fonte: (GONZALEZ; WOODS, 2009)

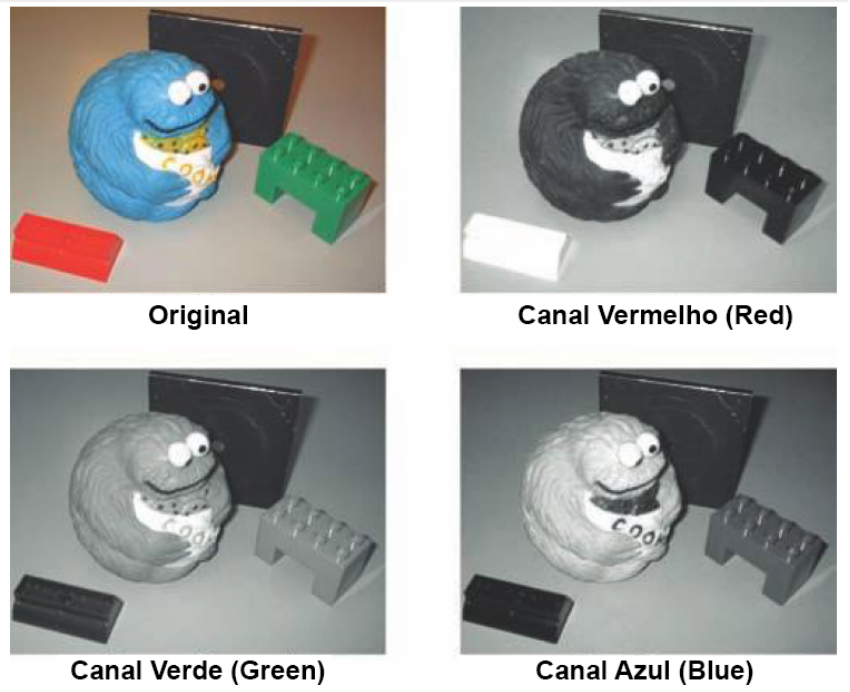
A Figura 2.5 mostra um esquemático do processo de aquisição de uma imagem utilizando sensores matriciais. O elemento da cena é iluminado por uma fonte de iluminação (deve-se lembrar que o próprio objeto poderia transmitir a iluminação), em seguida a energia

de entrada é coletada e projetada em um plano de imagem, que coincide com o arranjo de sensores, que por sua vez produz saídas proporcionais a integral da luz recebida em cada sensor. Essas saídas então são convertidas em sinais analógicos e, por último, em sinais digitais que representam a imagem digitalizada.

2.1.1.3 RGB

O sistema de codificação RGB é o espaço de cores mais utilizado para representação de imagens digitais. Uma imagem codificada em RGB pode ser descrita como um conjunto de vetores de três dimensões, pois, essa pode ser considerada como um conjunto de três planos bidimensionais, sendo um plano para cada valor de cor Vermelha *(**R**ed)*, Verde *(**G**reen)* ou Azul *(**B**lue)*. A cor resultante exibida no monitor ou em dispositivo semelhante é uma combinação de valores para cada um desses canais em cada *pixel*. A Figura 2.6 exibe uma imagem codificada em RGB e a representação visual de cada um de seus canais quando isolados. Nessa imagem nota-se que o objeto azul possui maior brilho na representação do canal azul (B), pois esse objeto possui mais luz azul, porém, ele também é representado nos outros canais de forma menos brilhante (SOLOMON, 2011).

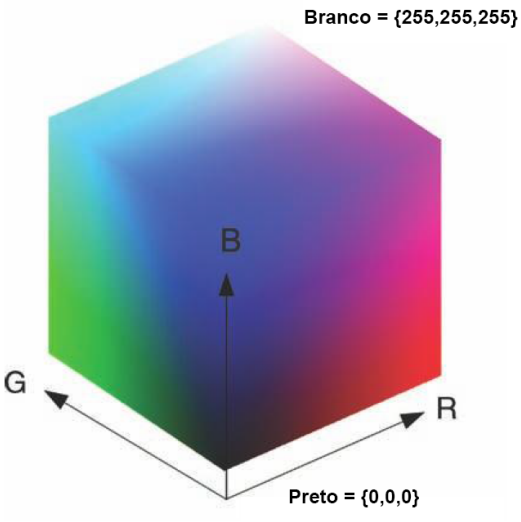
Figura 2.6 – Imagem em codificação RGB separada por canais.



Fonte: Adaptado de (SOLOMON, 2011)

O espaço de cores RGB pode ser representado com um cubo de cores com eixos R, G e B, onde todas as possíveis combinações estão presentes. Cada eixo possui a mesma faixa de valores de 0 a 1, que pode ser mapeada em valores de 0 a 255 quando trabalha-se com 1 *byte* por canal, representação conhecida como *true color* de 24 *bits*, conforme apresentado na Figura 2.7.

Figura 2.7 – Espaço de cores RGB.



Fonte: Adaptado de (SOLOMON, 2011)

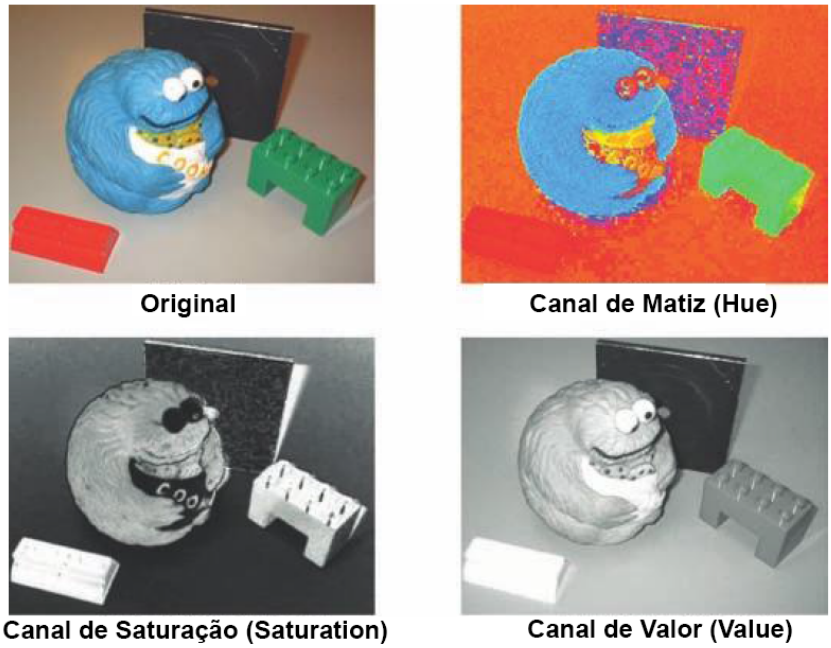
Devido a característica não linear desse espaço de cores, sua representação é de difícil interpretação pelos seres humanos pois não é relacionado com a forma como as cores são naturalmente percebidas. Uma alternativa é o uso de espaços de cores perceptuais, como por exemplo, o *Hue, Saturation and Value (HSV)*.

2.1.1.4 HSV

O espaço de cores HSV é o exemplo mais conhecido de espaços de cores perceptuais. O valor da cor resultante é dado por cada um de seus componentes (*hue*, *saturation* e *value*). A Figura 2.8 exibe uma imagem codificada em HSV e a decomposição de cada componente. Cada um dos componentes pode ser interpretado da seguinte forma:

- **H *(hue)*:** comprimento de onda dominante da cor, ou seja, vermelho, azul ou verde.
- **S *(saturation)*:** é a quantidade de luz branca presente na cor (seu nível de pureza).
- **V *(value)*:** representa a luminância da cor, ou seja, seu brilho na imagem.

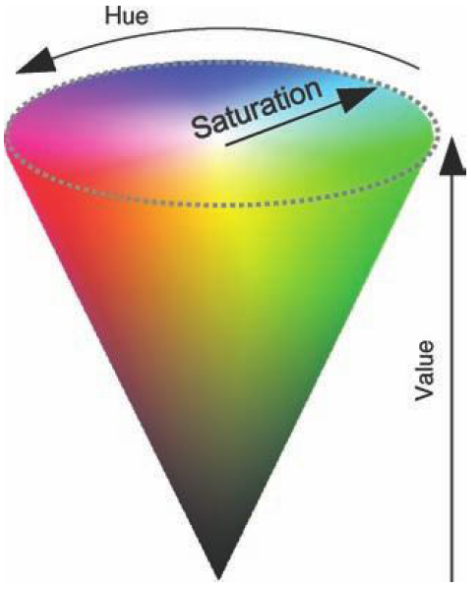
Figura 2.8 – Imagem em codificação HSV separada por componentes.



Fonte: Adaptado de (SOLOMON, 2011)

Assim como no espaço de cores RGB, uma imagem codificada em HSV pode ser representada por um conjunto de vetores de três dimensões, onde cada *pixel* contém um 3-upla (h, s, v) que pode ser convertida em RGB, caso necessário (SOLOMON, 2011). A Figura 2.9 exibe a representação gráfica do espaço de cores HSV.

Figura 2.9 – Espaço de cores HSV.



Fonte: (SOLOMON, 2011)

2.1.1.5 Tamanho de uma Imagem

O volume de dados é um dos maiores gargalos na representação, armazenamento e transmissão de imagens. Para calcular a quantidade de *bits* necessária para armazenar uma imagem digital em escala de cinza deve-se realizar o cálculo *MxNxB*, onde M representa a largura da imagem e N o comprimento, ambas medidas em *pixel*. A profundidade de tons utilizado para representação, em *bits*, é apresentada por B. Caso a imagem seja colorida, deve ser realizado o mesmo cálculo, porém o resultado deve ser multiplicado pela quantidade de canais de cores utilizada. O Quadro 2.2 exibe o volume de dados demandado (em *bytes*) para variadas configurações de imagens monocromáticas ou coloridas.

Quadro 2.2 – Espaço de dados para representação de diferentes imagens.

Dimensões Espaciais	**Resolução *Pixel (bits)***	**Tipo de Imagem**	**Volume de Dados *(bytes)***
128 x 128	1	Monocromática	2.048
256 x 256	1	Monocromática	8.192
512 x 512	1	Monocromática	32.768
1024 x 1024	1	Monocromática	131.072
128 x 128	8	Monocromática	16.384
256 x 256	8	Monocromática	65.536
512 x 512	8	Monocromática	262.144
1024 x 1024	8	Monocromática	1.048.576
128 x 128	3	Tricromática	6.144
256 x 256	3	Tricromática	24.576
512 x 512	3	Tricromática	98.304
1024 x 1024	3	Tricromática	393.216
128 x 128	24	Tricromática	49.152
256 x 256	24	Tricromática	196.608
512 x 512	24	Tricromática	786.432
1024 x 1024	24	Tricromática	3.145.728

Fonte: Adaptado de (BOVIK, 2005)

2.2 Criptografia

A comunicação sempre foi uma forma poderosa de troca de informações e de sustentabilidade para os governos. Reis e rainhas usavam a troca de mensagens para ordenar aos seus

exércitos ataques aos inimigos e para comunicação com outros países e membros do governo. Caso alguma dessas informações fosse interceptada pelos seus inimigos, as consequências poderiam ser devastadoras. Por exemplo, caso uma rainha desejasse atacar as terras de um país vizinho rival ao seu governo ela então enviaria para seus exércitos da fronteira uma mensagem informando que deveriam atacar ao amanhecer. Se a mensagem fosse entregue com sucesso, o ataque poderia ser realizado beneficiando-se do fator surpresa, ou seja, do fato do outro país não saber do ataque. Porém, caso a mensagem fosse interceptada de alguma forma no caminho, a pessoa que a interceptou poderia avisar ao exército do outro país sobre o ataque que ocorreria, e assim o elemento surpresa seria perdido (SINGH, 2011). Atualmente, a comunicação segura é imprescindível não só para assuntos de guerra, mas para a economia, para as empresas, e até mesmo para o cidadão comum que utiliza, por exemplo, os serviços de compras pela internet e também casas e cidades inteligentes. Sendo assim, é necessário um modo de proteger e garantir a integridade das informações, não só para objetivos militares, mas também para privacidade individual dos cidadãos em suas comunicações. Com esse objetivo, surge então a Criptografia, derivada da palavra grega *Kriptos* (oculto) e *graphein* (escrever). Seus estudos e a busca por modos cada vez mais seguros de comunicação se estendem até os dias de hoje.

2.2.1 Tipos de Criptografia

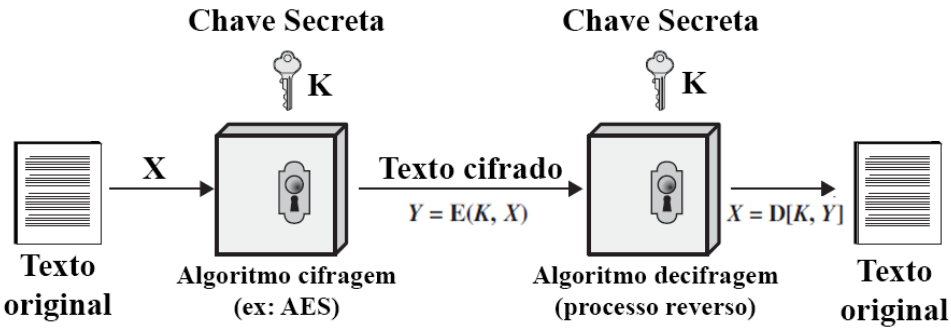
Os algoritmos de criptografia podem ser classificados quanto ao tratamento que é realizado nas informações que serão processadas e também quanto ao número de chaves utilizado nesse processo. Quanto ao número de chaves, pode-se identificar dois tipos de criptografia: simétricos ou assimétricos; já quanto ao tratamento, os algoritmos são separados por blocos ou por fluxo (STALLINGS, 2010).

2.2.1.1 Criptografia Simétrica

O conceito de criptografia simétrica é a unicidade da chave de cifragem e decifragem. Tanto o emissor de uma mensagem quanto o destinatário compartilham a mesma chave para comunicação. Esses, por sua vez, combinam previamente qual chave será usada, assim, o remetente cifra sua mensagem com essa chave e envia para o destinatário, que utiliza a mesma chave para decifrar a mensagem recebida e ler seu conteúdo. O ponto fraco desse tipo de criptografia é justamente sua única chave, pois, torna-se necessário o envio de maneira segura dessa ao destinatário para que assim ele possa ser capaz de decifrar a comunicação. Deve-se também

preocupar para que ninguém não autorizado intercepte tal chave, pois senão, toda a comunicação subsequente será decifrada. A Figura 2.10 mostra um esquemático para o modelo simétrico. São exemplos de algoritmos de criptografia simétricos: DES, 3DES, AES, Blowfish e GPG.

Figura 2.10 – Esquema de criptografia simétrica.

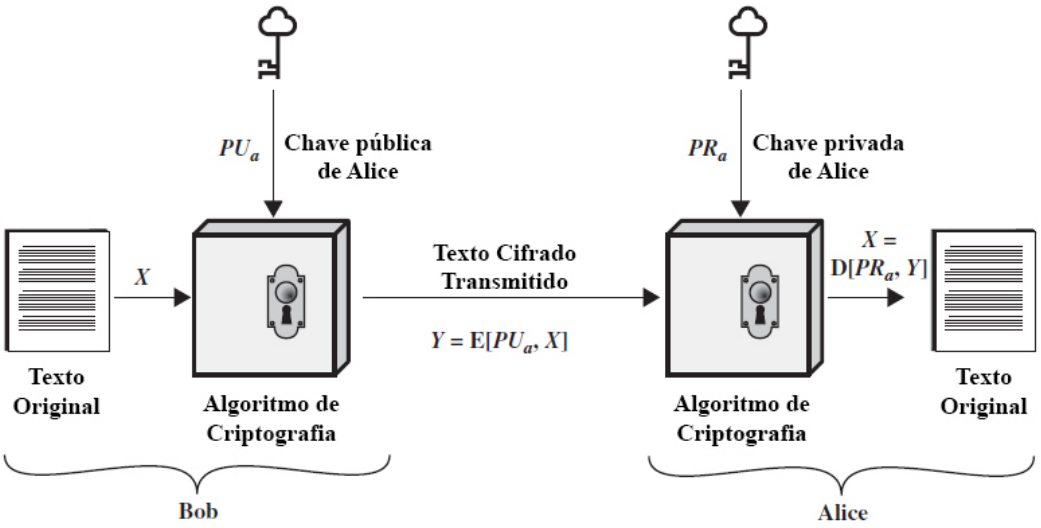


Fonte: Adaptado de (STALLINGS, 2010)

2.2.1.2 Criptografia Assimétrica

Como forma de solucionar a deficiência da distribuição de chaves no modelo simétrico, foi proposto o modelo assimétrico de criptografia. Nesse modelo, não é usada apenas uma chave, mas sim um par de chaves. O princípio básico desse modelo é a existência de uma chave pública, ou seja, que deve ser divulgada a todos, e uma chave privada, que deve ser guardada em segredo. Quando o remetente *Bob* deseja enviar uma mensagem de maneira segura para *Alice*, ele deve obter sua chave pública e utilizá-la no algoritmo de criptografia. O resultado será enviado para *Alice* de maneira segura por meio de um canal inseguro. Ao receber a mensagem de *Bob*, *Alice* utilizará sua chave privada para decifrar tal mensagem. A segurança do sistema reside no fato de que somente a chave privada de *Alice* é capaz de decifrar uma mensagem cifrada com sua chave pública. A Figura 2.11 mostra um esquemático para o modelo assimétrico. São exemplos de algoritmos de criptografia assimétricos: ElGamal, RSA e ECC.

Figura 2.11 – Esquema de criptografia assimétrica.

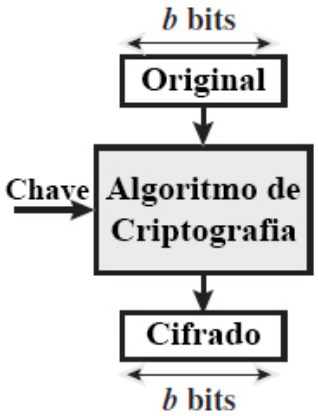


Fonte: Adaptado de (STALLINGS, 2010)

2.2.1.3 Criptografia por Blocos

A criptografia por blocos é aquela onde o texto original é dividido em blocos de tamanho n, que são tratados para produzir blocos cifrados também de tamanho n, onde n geralmente equivale a 64 ou 128 *bits*. A Figura 2.12 mostra um esquemático do processo de cifragem de uma mensagem utilizando esquema baseado em blocos, nela é possível observar que o texto cifrado possui mesmo tamanho do texto original.

Figura 2.12 – Esquema de criptografia por blocos.



Fonte: Adaptado de (STALLINGS, 2010)

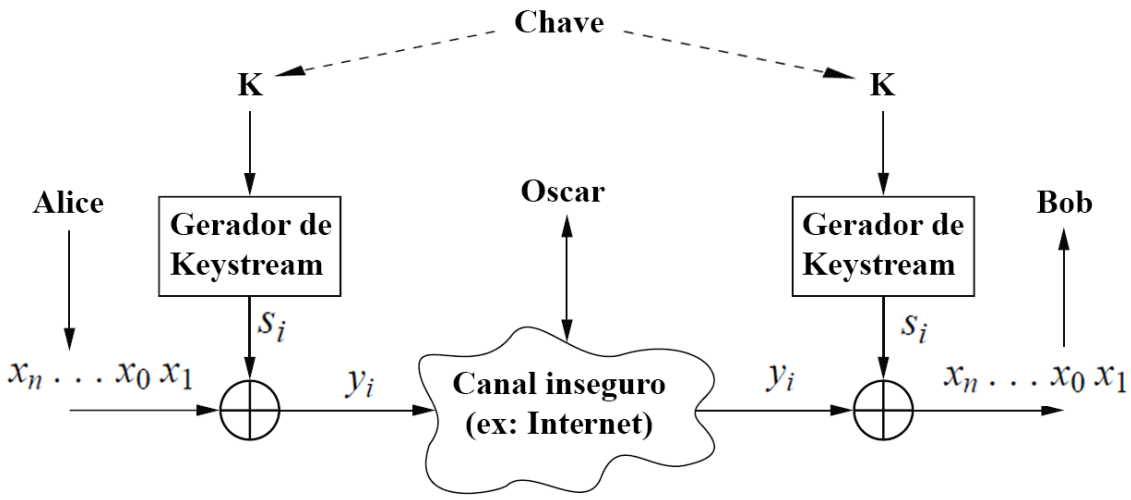
2.2.1.4 Criptografia por Fluxo

Na criptografia por fluxo os *bits* ou *bytes* do texto original são cifrados individualmente. Esse processo pode ser realizado por meio da aplicação da operação lógica ou-exclusivo (XOR - $\bigoplus$) entre os bits do texto original e os bits de uma cadeia binária gerada, por exemplo, por uma função pseudoaleatória. Ao projetar um criptossistema baseado em fluxo, deve-se atentar a qualidade do gerador de sequência pseudoaleatória, garantindo assim a dificuldade em quebrar a mensagem cifrada. Além disso, deve-se atentar para a necessidade da implementação desse gerador em ambos os lados comunicantes, já que dada uma semente, a mesma sequência deverá ser gerada tanto para o processo de cifragem quanto para a decifragem.

Nesse tipo de criptografia, a chave inicial deve ser conhecida por ambos os lados comunicantes. A troca de tal chave pode ser feita de maneira prévia por meio de um encontro pessoal, ou utilizando algum protocolo de troca de chaves, como por exemplo, o protocolo Diffie-Hellman. Outra possível solução seria a utilização do conceito de criptografia híbrida. Esse conceito aplica ambos os tipos de criptografia, tanto simétrica quanto assimétrica, para comunicação segura. No caso desse exemplo, a chave inicial seria enviada para o destinatário por meio de um outro algoritmo de criptografia assimétrico, e o restante da comunicação seria realizada normalmente através do algoritmo principal selecionado.

Na Figura 2.13, é mostrado um esquemático do processo de cifragem e decifragem baseado em fluxo. Em ambos os lados comunicantes está presente o mesmo gerador de sequências pseudoaleatórias, que é inicializado por meio de uma chave inicial *(initial key)*. No exemplo da Figura 2.13, uma pessoa chamada *Alice* deseja enviar uma mensagem para *Bob*. Dessa maneira, *Alice* cifra as mensagens em texto original (x_n) por meio da operação XOR com os *bits* aleatórios (s_i) e envia o resultado (y_i) para *Bob* através de um canal inseguro sem que um interceptador como *Oscar* possa ter acesso à mensagem. *Bob* por sua vez aplica a mesma operação XOR na mensagem cifrada, obtendo novamente a mensagem original enviada por *Alice*. Todo o processo ocorre de forma segura mesmo que para isso utilize-se de um canal inseguro de comunicação (PAAR; PELZL, 2010).

Figura 2.13 – Esquema de criptografia por fluxo.



Fonte: Adaptado de (PAAR; PELZL, 2010)

Utilizar a operação XOR, também chamada de adição módulo 2, entre os *bits* de texto original e *bits* do gerador é confiável devido a natureza dessa operação. Como é mostrado na Tabela 2.1, caso o interceptador de uma mensagem criptografada obtenha um *bit* "0" (coluna Y_i) ele não será capaz de determinar se o *bit* da mensagem original é "0" ou "1", pois existem duas situações previstas onde a combinação do *bit* original (coluna X_i) e do valor do gerador (coluna S_i) resultam no valor cifrado "0". Além disso a função XOR é inversível e também balanceada, ou seja, dada uma combinação de *bits* cifrados e a chave, é possível recuperar a mensagem original aplicando a mesma operação novamente. Já a característica de balanceamento ocorre pois, para qualquer combinação de *bits* de entrada existe 50% de chance do resultado ser "0" ou "1", caso o gerador aleatório seja uniforme (PAAR; PELZL, 2010).

Tabela 2.1 – Tabela verdade da operação XOR.

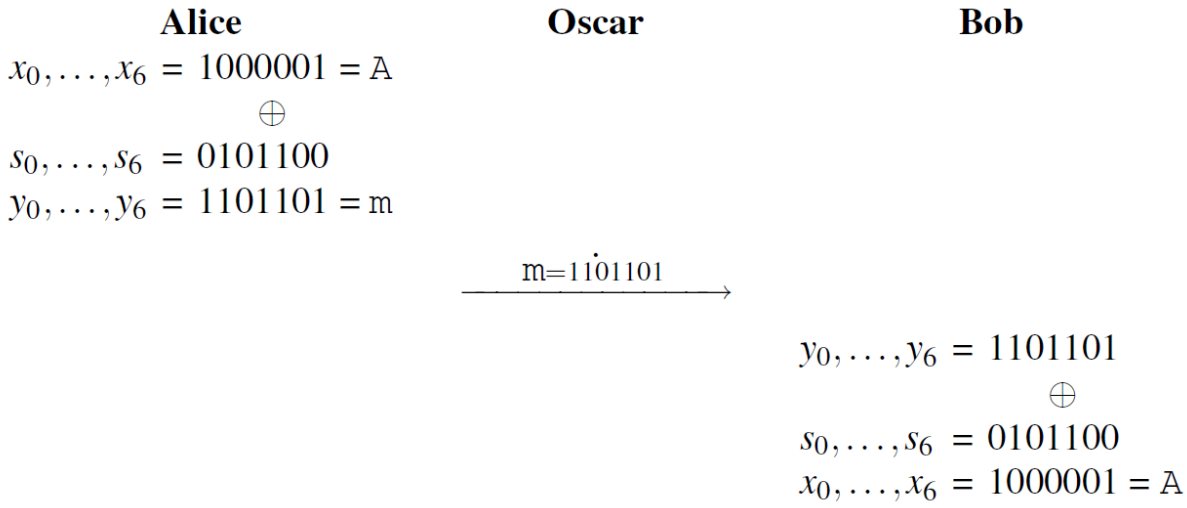
X_i	S_i	Y_i
0	**0**	**0**
0	**1**	**1**
1	0	1
1	1	0

Fonte: (PAAR; PELZL, 2010)

Paar e Pelzl (2010) demonstram na Figura 2.14 o processo de criptografia de uma mensagem trocada entre *Alice* e *Bob*. Para isso são aplicados os conceitos de cifra por fluxo e operação XOR. Em um primeiro momento *Alice* decide enviar uma mensagem para *Bob*, essa mensagem

contém apenas o caractere “A” que é convertido para representação ASCII, cujo valor decimal é 65_{10} e em representação binária equivale a 1000001_2, nesse exemplo, os sete primeiros *bits* da chave de criptografia são 0101100_2. A comunicação inicia com *Alice* executando a operação XOR entre a cadeia binária que representa o caractere em codificação ASCII e os *bits* da chave, obtendo assim a sequência cifrada 1101101_2 que equivale ao caractere “m”. *Alice* envia a mensagem através de um canal inseguro onde o interceptador *Oscar*, apesar de observar o fluxo da mensagem não é capaz de decifrá-la, pois esse não possui a chave de criptografia utilizada por *Alice*. A mensagem é recebida por *Bob* que é então capaz de decifrá-la utilizando a chave e aplicando a operação XOR, resultando na mensagem original "A".

Figura 2.14 – Exemplo de criptografia por fluxo fazendo uso de operação XOR.



Fonte: (PAAR; PELZL, 2010)

2.3 Números Aleatórios

Nos sistemas de criptografia por fluxo é de extrema importância a qualidade da sequência de *bits* aleatórios *(keystream)* que compõem a chave criptográfica utilizada. A geração dessa sequência é de responsabilidade do gerador aleatório do sistema, sendo assim, a segurança de todo o sistema é dependente desse componente. O *keystream* deve ter comportamento aleatório, de modo que um interceptador não seja capaz de advinhar o próximo *bit* da sequência, caso contrário, no exemplo mostrado na Figura 2.14 Oscar seria capaz de decifrar a mensagem enviada por Alice (PAAR; PELZL, 2010). Nesta seção, serão explicados os conceitos básicos de números aleatórios, dando ênfase nos LFSRs *(Linear Feedback Shift Registers)* e suas técnicas de construção, devido sua ampla utilização nos algoritmos do projeto eSTREAM.

2.3.1 Geradores de Números Aleatórios

Números aleatórios são de grande importância não só para a criptografia, mas também para simulação e estatística. Para obter tais números, é necessária a construção de sistemas com capacidade de gerá-los, de modo geral existem três tipos de geradores aleatórios, são eles (PAAR; PELZL, 2010):

- ***True Random Number Generators (TRNG):*** são caracterizados pela irreprodutibilidade de sua saída, ou seja, é virtualmente impossível serem geradas duas sequências de tamanho n idênticas. Esses geradores são baseados em processos físicos, como por exemplo o lançamento de uma moeda e o decaimento de elementos radioativos. Saídas desses geradores são comumente utilizadas em criptografia como chaves de sessão, que são distribuídas geograficamente entre os pares e alimentam geradores pseudoaleatórios (PAAR; PELZL, 2010). Para serem utilizados diretamente em propósitos criptográficos esses devem ter saídas imprevisíveis, e, como alguns processos físicos podem ser previsíveis torna-se interessante a combinação de saídas de mais de uma fonte para geração dos valores. Porém, para produzir grandes quantidades de números em TRNGs é demandado uma grande quantidade de tempo, sendo mais recomendável nesse caso a utilização de geradores pseudoaleatórios (RUKHIN et al., 2001).

- ***Pseudorandom Number Generators (PRNG):*** utilizados como alternativa à dificuldade de geração dos TRNGs, os geradores de números pseudoaleatórios são baseados em sequências calculadas a partir de um valor inicial chamado semente. A saída de um PRNG é comumente obtida por meio da aplicação de funções determinísticas nessa semente. É essa natureza determinística que leva a adoção do nome pseudoaleatório, uma vez que todos os valores gerados são obtidos por meio de uma semente inicial, e além do mais, para obter a mesma sequência novamente basta guardar a semente utilizada e aplicar o mesmo algoritmo de geração. Um requisito esperado desses geradores é a qualidade estatística de sua saída. Seu uso pode ser variado, incluindo diversas aplicações diferentes de sistemas de segurança e criptografia, um exemplo é na implementação do gerador LCG *(Linear Congruential Generator)* presente na função *rand()* da linguagem ANSI C. O comportamento desses geradores pode ser representado pela forma recursiva mostrada na Equação 2.1, onde a partir de um valor inicial *(seed)* os próximos números da sequência $S_{i+1}, \ldots, S_{i+n}$ são determinados pela aplicação de uma função f no valor atual

S_i (PAAR; PELZL, 2010). Geradores pseudoaleatórios possuem um tamanho predefinido de números que podem ser gerados, onde, a partir desse ponto a sequência gerada é repetida em ciclo. A esse tamanho é dado o nome de período (JAMES, 1990). Bons geradores pseudoaleatórios possuem longos períodos, tornando assim a aplicação onde são utilizados mais segura.

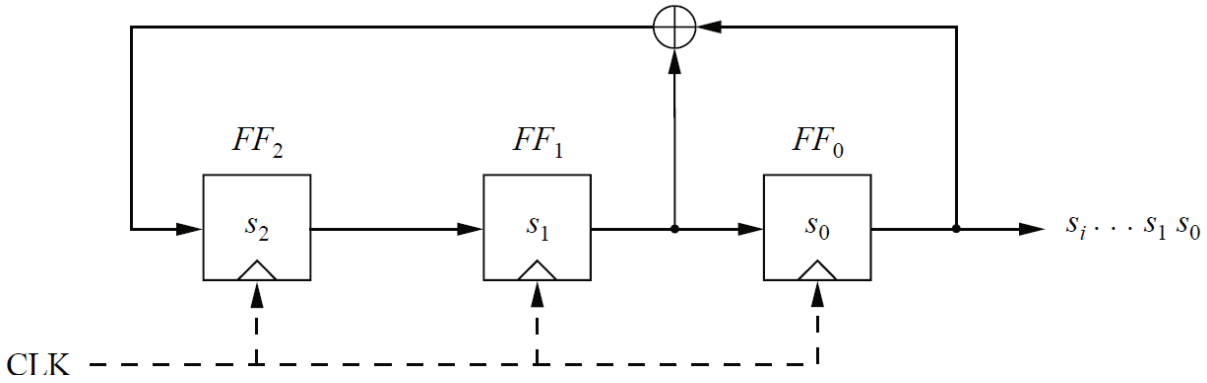
$$\begin{aligned} s_0 &= seed \\ s_{i+1} &= f(s_i), i = 0, 1, \ldots \end{aligned} \tag{2.1}$$

- ***Cryptographically Secure Pseudorandom Number Generators (CSPRNG):*** essa variante de gerador é um tipo especial de PRNG onde a saída desse não é previsível, ou seja, dado *n bits* consecutivos gerados, não existe algoritmo de tempo polinomial capaz de predizer o próximo *bit* S_{n+1} com chance de sucesso superior à 50%. De forma análoga, também não é possível calcular os valores para os *bits* antecessores à essa cadeia de *n bits*, ou seja, $S_{n-1}, S_{n-2}, \ldots$. Essa característica é necessária somente para aplicações com propósito criptográfico, para demais aplicações os geradores PRNG são suficientes. Quase a totalidade dos geradores do tipo PRNG que não são explicitamente desenvolvidos com propósito criptográfico não são CSPRNGs (PAAR; PELZL, 2010).

Um exemplo de gerador pseudoaleatório comumente utilizado em sistemas de criptografia implementados em *hardware* são aqueles baseados em registradores de deslocamento. Esses geradores estão presentes em todos os sistemas de Perfil II do portfólio do eSTREAM, que serão discutidos na Seção 2.4. Um exemplo dessa classe de gerador é o LFSR *(Linear Feedback Shif Register)*, que consiste de elementos de memória *(flip-flops)* sincronizados por um sinal de *clock* e um caminho de realimentação do circuito.

Uma visão geral de um LFSR de grau três com valores iniciais S_2, S_1 e S_0 é mostrado na Figura 2.15. A saída de um LFSR é produzida na extremidade direita do circuito, a cada ciclo de *clock* aplicado no circuito, os valores contidos nos *flip-flops* FF_2, FF_1 e FF_0 são deslocados para à direita até a saída do gerador, portanto, a cada ciclo um *bit* pseudoaleatório é produzido. O caminho de realimentação é montado pela combinação das saídas de alguns dos *flip-flops* por meio da operação lógica ou-exclusivo, o *bit* gerado mais recentemente pelo LFSR é utilizado como entrada para o elemento de memória mais a esquerda do circuito.

Figura 2.15 – Esquemático de um LFSR de grau 3.



Fonte: (PAAR; PELZL, 2010)

Assumindo no LFSR da Figura 2.15 que os valores de $S_2 = 1$, $S_0 = 0$ e $S_0 = 0$, é possível listar todos os valores de saída gerados, como mostrado na Tabela 2.2. Nesta tabela, observa-se que a partir do sexto ciclo de *clock*, a sequência 0010111 começa a ser repetida, esse fato ocorre pois o período do gerador alcança seu limite, ou seja, a partir daquele momento o gerador começa a repetir os *bits* gerados anteriormente. Um LFSR com grau n é capaz de gerar no máximo $2^n - 1$ diferentes *bits* e para alcançar período máximo deve ter as posições de realimentação definidas segundo um polinômio primitivo (PAAR; PELZL, 2010), uma tabela com os polinômios primitivos mais utilizados pode ser encontrada em Ward e Molteno (2007).

Tabela 2.2 – Sequência gerada pelo LFSR da Figura 2.15.

clk	FF_2	FF_1	$FF_0 = S_i$
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

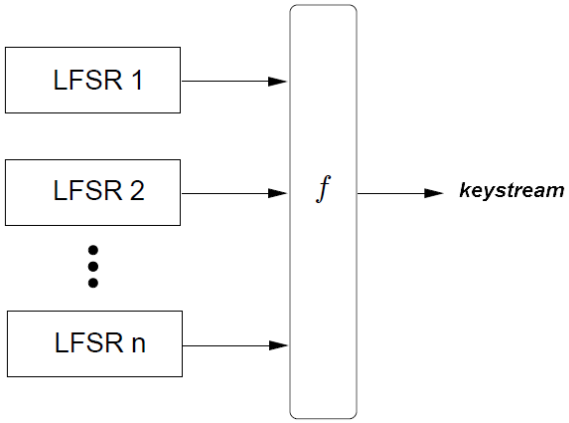
Fonte: (PAAR; PELZL, 2010)

A desvantagem dos LFSRs é que esses são lineares, ou seja, os valores de sua saída são compostos por uma combinação linear dos valores de entrada. Dessa forma, é possível prever o próximo *bit* gerado, recuperando as posições de realimentação do circuito. Para isso é neces-

sário apenas que seja conhecido $2m$ *bits* de saída de um LFSR de grau m e, a partir de então, é possível recuperar os coeficientes de realimentação simplesmente resolvendo um sistema linear. Ao fazer isso, o gerador pode ser reconstruído por um indivíduo invasor (PAAR; PELZL, 2010). Na obra de Menezes (1996), é afirmado que um LFSR nunca deve ser utilizado para aplicações de propósito criptográfico devido à linearidade inerente à sua construção. Porém, são destacadas três técnicas para remover a linearidade desses geradores sendo assim possível a sua utilização em tais aplicações, são elas:

- **Combinador não linear:** consiste na utilização de vários LFSRs em paralelo, onde a saída de cada um desses geradores é entrada de uma função de combinação não linear f, capaz de satisfazer uma série de propriedades que garantam sua segurança contra certos tipos de ataques. A Figura 2.16 mostra um esquemático dessa técnica.

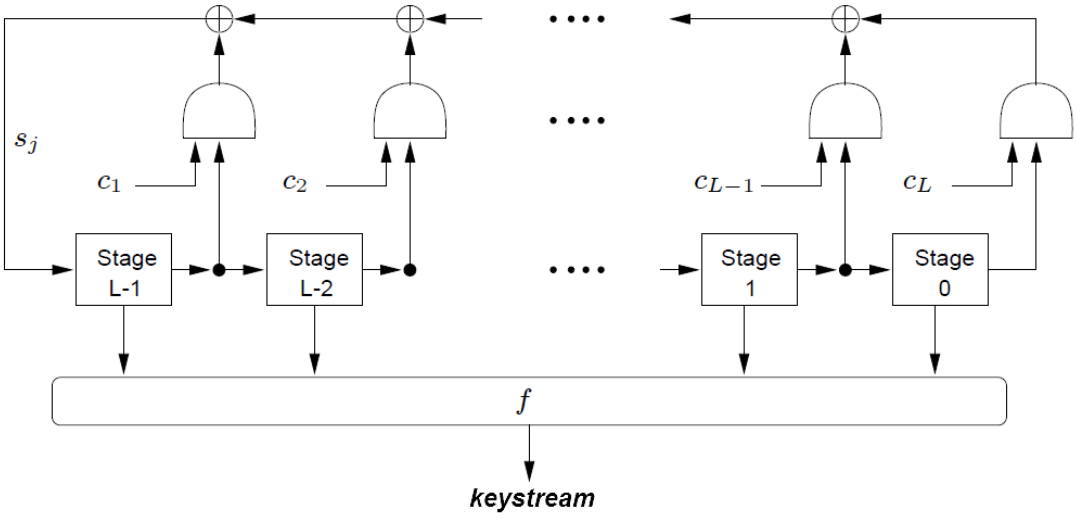
Figura 2.16 – Exemplo de combinador não linear.



Fonte: (MENEZES, 1996)

- **Filtro não linear:** esse tipo de gerador utiliza apenas um LFSR, porém para acabar com a linearidade, são utilizadas funções não lineares, chamadas de funções filtrantes, que são alimentadas com valores intermediários desse LFSR. A Figura 2.17 mostra um esquemático da construção desse gerador.

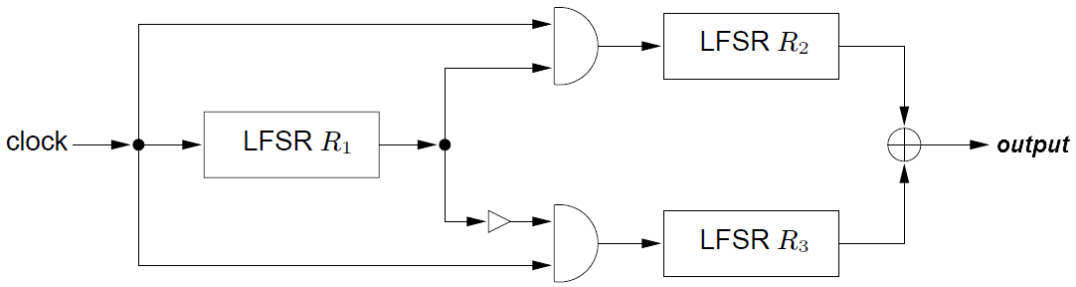
Figura 2.17 – Exemplo de filtro não linear.



Fonte: (MENEZES, 1996)

- **Controle de *clock*:** nos geradores construídos por meio de combinadores e filtros não lineares, os LFSRs que os compõe possuem *clock* regular, ou seja, o movimento de todos os dados ao longo dos LFSRs são comandados por um mesmo sinal de sincronismo. Nesta técnica, a remoção da linearidade é tratada por meio do controle do sinal de *clock* de um gerador LFSR por meio dos *bits* de saída *(keystream)* de um segundo gerador. Dessa maneira, ataques baseados na regularidade do funcionamento dos LFSRs são dificultados, pois, o sinal de *clock* gerado pelo LFSR é irregular. A Figura 2.18 mostra um exemplo de um sistema que aplica tal técnica, o gerador mostrado é chamado de gerador de sequências alternadas *(alternating step generator)*.

Figura 2.18 – Gerador de sequências alternadas.



Fonte: (MENEZES, 1996)

2.4 O Projeto eSTREAM

Anunciado no ano 2004 pelo ECRYPT *(European Network of Excellence for Cryptology)*, o projeto tem como objetivo aumentar a experiência de técnicas de desenvolvimento e análise de algoritmos de cifra por fluxo (ROBSHAW, 2008), além de identificar um portfólio de cifras de fluxo promissoras (RIJMEN, 2010). A principal motivação para o início desse projeto foi um discurso do criptógrafo Adi Shamir durante o evento *RSA Data Security Conference* em 2004. Nessa ocasião, Shamir questionou sobre o estado dos algoritmos de fluxo naquele momento e a popularização crescente do algoritmo simétrico AES, além de discutir a necessidade do desenvolvimento de algoritmos dedicados para criptografia por fluxo. Shamir concluiu mostrando duas áreas onde o uso de algoritmos de cifra por fluxo tendem a oferecer vantagens se comparados a cifra por blocos, são elas (ROBSHAW, 2008):

- Aplicações de *software* onde é requerida uma alta vazão de dados;
- Implementações em *hardware* com limitação de recursos.

A fase inicial do projeto eSTREAM, também chamada de fase de preparação teve início em 2004 e contou com o auxílio de especialistas em criptografia, tanto da indústria quanto da academia, contactados para discutir quais seriam os requisitos esperados dos algoritmos a serem submetidos. No total, ao final dessa fase, ocorrido em 2005, foram submetidos 34 algoritmos para análise, sendo estes apresentados no Quadro 2.3. Como forma de auxiliar os desenvolvedores e suprir as observações levantadas por Shamir, o projeto foi organizado de maneira a refletir dois perfis de cifras por fluxo a serem analisadas. Apesar dessa separação, muitos pesquisadores propuseram algoritmos adequados para ambos os perfis. Porém, de acordo com Robshaw (2008), nenhum desses conseguiram avançar até a fase final de avaliação do projeto. Portanto, o portfólio atual do eSTREAM não contém nenhum algoritmo adequado para ambos os perfis. Os perfis propostos e as características esperadas em cada um são (RIJMEN, 2010):

- **Perfil I:** cifras para aplicações de *software* com necessidade de alta vazão de dados.
- **Perfil II:** cifras voltadas para implementação em *hardware* com alta restrição de recursos, ou seja, restrição de área e baixo consumo de energia.

O projeto eSTREAM foi temporalmente dividido em três fases de avaliação. Em cada uma dessas fases, os algoritmos eram amplamente testados e técnicas de criptoanálise eram

Quadro 2.3 – Algoritmos inicialmente submetidos ao eSTREAM.

Perfil I	**Perfil I e II**	**Perfil II**
ABC	F-FCSR	Achterbahn
CryptMT/Fubuki	Hermes8	DECIM
DICING	LEX	Edon-80
DRAGON	MAG	Grain
Frogbit	NLS	MICKEY
HC-256	Phelix	MOSQUITO
Mir-1	Polar Bear	SFINKS
Py	POMARANCH	Trivium
SOSEMANUK	Rabbit	TSC-3
	SSS	VEST
	TRBDK3 YAEA	WG
	Yamb	ZK_Crypt
	Salsa20	

Fonte: (ROBSHAW, 2008)

realizadas tentando encontrar suas fraquezas. A cada fase era escolhido um subconjunto cada vez menor dos algoritmos inicialmente propostos na fase de submissão ocorrida em 2004. A fase final do projeto foi divulgada em 2008 e era composta por um portfólio de quatro algoritmos de Perfil I e outros quatro de Perfil II. Após a descoberta de fraquezas no algoritmo de Perfil II F-FCSR-H, a lista foi revisada, removendo tal algoritmo (RIJMEN, 2010). Portanto, até o momento o projeto é constituído de sete sistemas de cifra por fluxo, mostrados no Quadro 2.4.

Quadro 2.4 – Portfólio final de algoritmos após 1ª revisão (2008).

Perfil I	**Perfil II**
HC-128	Grain v1
Rabbit	Mickey v2
SALSA20/12	Trivium
SOSEMANUK	

Fonte: (RIJMEN, 2010)

Apesar do projeto não ser considerado como uma padronização, assim como ocorreu com AES pelo *National Institute of Standards and Technology (NIST)*, a análise dos algoritmos do eSTREAM contou com a participação de diversos pesquisadores especialistas em processos de padronização de importantes organizações (RIJMEN, 2010). A revisão do portfólio após a descoberta da quebra do algoritmo F-FCSR-H, torna o projeto confiável. Além disso, Robshaw (2008) afirma que o projeto pode ser atualizado caso sejam encontradas circunstâncias para tal. Em 2012, quatro anos após a divulgação do portfólio final, o ECRYPT divulgou outro docu-

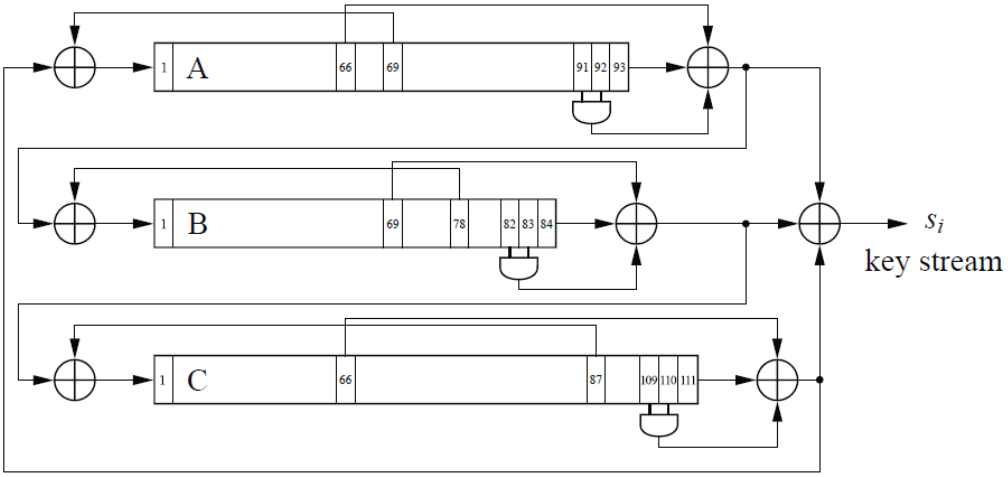
mento (CID; ROBSHAW, 2012), onde afirma que após quatro anos de testes na segurança e tentativas de criptoanálise nos algoritmos, nenhum desses teve sua segurança comprometida, e reafirmou que o projeto continua sendo mantido e que qualquer modificação no portfólio será anunciada no site oficial do projeto (ECRYPT II, 2012b). McKay et al. (2017) apresenta em um relatório técnico do NIST os três algoritmos de criptografia objetos de estudo deste trabalho, mostrando que no ano de 2017 tais algoritmos ainda são importantes. Vale ressaltar que ambos os sistemas de Perfil II foram implementados juntos em um único *hardware* de aplicação específica (ASIC) com tecnologia CMOS de 0.18 µm (eSCARGOT, 2008). A análise de performance dos sistemas em *hardware* pode ser conferida no artigo de Good e Benaissa (2007), já implementação e análise em FPGA é mostrada no artigo de Kitsos et al. (2013). Quanto à implementação em *software*, a análise dos sistemas de ambos os perfis pode ser encontrada em ECRYPT II (2012a). A seguir serão apresentados cada um dos finalistas de Perfil II do projeto.

2.4.1 TRIVIUM

O sistema de criptografia Trivium, desenvolvido por Christophe De Canniere e Bart Preneel (CANNIÈRE, 2006)(CANNIÈRE; PRENEEL, 2007), foi concebido com o propósito de explorar a simplicidade sem sacrificar a segurança, velocidade ou flexibilidade. Seu projeto é baseado na combinação de três registradores de deslocamento acrescido de componentes não lineares usados na saída de cada um desses registradores (PAAR; PELZL, 2010). O sistema foi projetado para gerar até 2^{64} *bits* de valores *(keystream)* utilizando para isso uma chave de 80 *bits* e vetor de inicialização *(Initialization Vector - I.V.)*[2] de 80 *bits*. Assim como a maioria das cifras de fluxo, o funcionamento do Trivium é divido em duas fases, a primeira consiste na inicialização dos estados internos do sistema usando o valor da chave e do I.V., em seguida o sistema é repetidamente atualizado a fim de gerar cada um dos *bits* do *keystream* (CANNIÈRE; PRENEEL, 2007). A estrutura do sistema, mostrada na Figura 2.19, é composta por três registradores de deslocamento, A, B e C, de tamanhos 93, 84 e 111 respectivamente. O fluxo de *bits* de saída é gerado por meio de uma função XOR dos valores de saída de cada registrador de deslocamento. Como a saída de cada registrador é conectada à entrada do outro, o sistema pode ser compreendido como um arranjo circular, ou seja, um único registrador de deslocamento de tamanho $93 + 84 + 111 = 288$, a Figura 2.20 mostra essa configuração (PAAR; PELZL, 2010).

[2] Sequência utilizada como fator de aleatorização no processo de cifragem, para que assim, duas sequências cifradas com a mesma chave *(Key)* mas com I.V's diferentes resultem em saídas diferentes (CHIN; BUER; LUO, 2011).

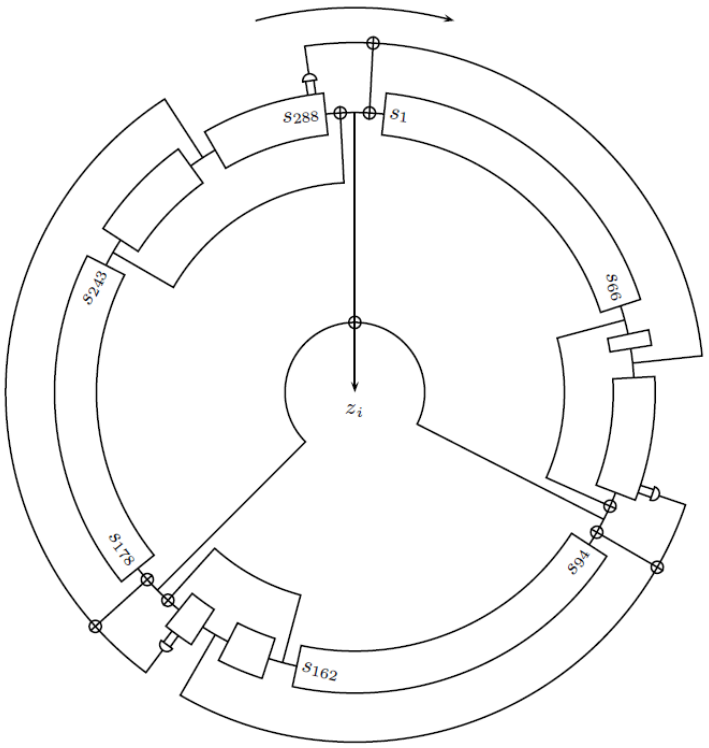
Figura 2.19 – Estrutura interna do Trivium.



Fonte: (PAAR; PELZL, 2010)

As operações AND presentes no sistema são de extrema importância para a segurança do sistema, pois previnem ataques que exploram a linearidade do sistema. A operação AND pode ser compreendida como uma multiplicação módulo 2, assim, caso sejam multiplicados dois valores desconhecidos, como são os conteúdos dos registradores, o resultado das equações não é mais linear, o que evita assim a utilização de técnicas para solução de sistemas lineares no processo de criptoanálise (PAAR; PELZL, 2010).

Figura 2.20 – Interpretação circular do Trivium.



Fonte: (CANNIÈRE; PRENEEL, 2007)

O Trivium é um sistema orientado à geração de apenas um *bit* por ciclo. Essa característica foi determinada devido ao tamanho compacto em *hardware* gerado por esse tipo de configuração, além do que, os autores esperam que com essa configuração o sistema seja menos suscetível ao fenômeno de correlação entre os *bits* gerados. Apesar dessa característica o Trivium permite operação em modo paralelizado o que fornece um modo de geração de até 64 *bits* por ciclo (CANNIÈRE, 2006). Considerando as proporções de 12 *gates* NAND para cada *flip-flop*, 2.5 *gates* por XOR e 1.5 *gates* por AND, é estimada a quantidade total de *gates* para cada variação de *bits*/ciclo, mostrado na Tabela 2.3. Com base na observação dessa tabela, verifica-se que a implementação da versão de 64 *bits* não chega a dobrar a quantidade de *gates* necessários quando comparada à implementação para geração de um único *bit*. Esse fato demonstra o alto poder de escalabilidade desse sistema, sem comprometer recursos de *hardware*.

Tabela 2.3 – Estimativa de número de *gates* para variação de 1 a 64 *bits* do Trivium

Components	1-bit	8-bit	16-bit	32-bit	64-bit
Flip-flops:	288	288	288	288	288
AND gates:	3	24	48	96	192
XOR gates:	11	88	176	352	704
NAND gate count:	3488	3712	3968	4480	5504

Fonte: (CANNIÈRE; PRENEEL, 2007)

Embora o sistema não tenha sido desenvolvido para aplicações de *software*, Cannière e Preneel (2007) mostra o desempenho da execução do sistema em um computador com processador Intel Xeon 1.5GHz para implementação em linguagem C, o resultado é exibido na Tabela 2.4. Outra análise de desempenho em *software*, dessa vez realizada pelo eSTREAM, pode ser encontrada em ECRYPT II (2012a).

Tabela 2.4 – Desempenho do sistema Trivium implementado em *software*.

Operation	
Stream generation:	12 cycles/byte
Key setup:	55 cycles
I.V. setup:	2050 cycles

Fonte: (CANNIÈRE; PRENEEL, 2007)

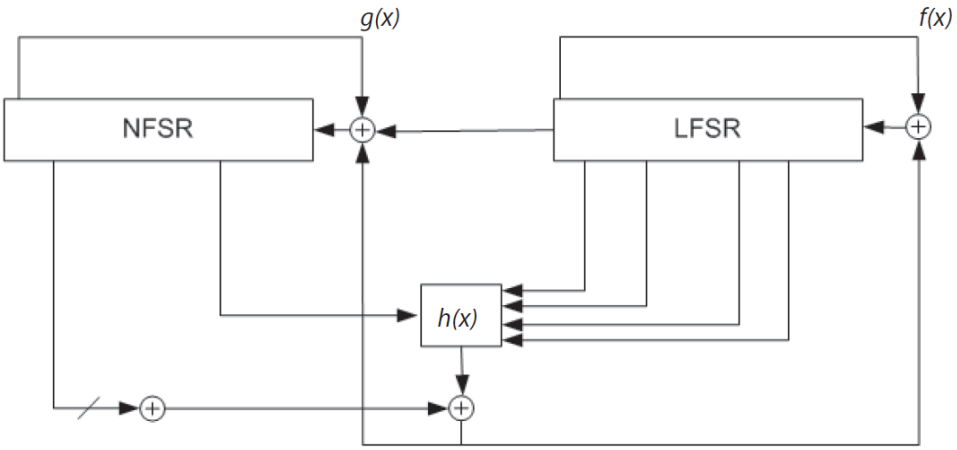
2.4.2 GRAIN

O sistema de criptografia Grain desenvolvido por Hell, Johansson e Meier (2007) tem como objetivo primário estabelecer um sistema de cifra por fluxo que agregue simplicidade, segurança e velocidade de processamento. Como exemplo de portabilidade os autores citam a importância de uma cifra capaz de ser acoplada no *hardware* de uma *tag* RFID, provendo dessa forma segurança no tráfego de informações, por exemplo em sistemas de pagamentos que utilizam tal tecnologia. Utilizando a cifra AES não é possível tal acoplamento, porém com o Grain esta situação torna-se viável.

A primeira versão funcional do algoritmo, denominada de Grain v1, foi proposta por Hell, Johansson e Meier (2007) e era baseado em cifra por fluxo síncrona, ou seja, o fluxo de bits aleatórios *(keystream)* é independente dos *bits* originais. A construção do Grain é baseada em dois registradores de deslocamento, sendo um linear i.e. *Linear Feedback Shift Register (LFSR)* e outro não linear i.e. *Nonlinear Feedback Shift Register (NFSR)*, ambos registradores possuem comprimento igual a 80 *bits*, chave de 80 *bits* e I.V. de 64 *bits*, sendo que o sistema foi desenvolvido para ser resistente a qualquer tipo de ataque exceto o de força bruta, que necessitaria de uma complexidade computacional não menor do que 2^{80} para quebrar a cifra. Um esquemático da construção do sistema é mostrado na Figura 2.21, de maneira sucinta o sistema é composto pela combinação de três funções, são elas:

- $f(x)$**:** representa o polinômio primitivo para construção do LFSR.
- $g(x)$**:** representa o polinômio para construção do NFSR.
- $h(x)$**:** função que atua como um filtro não linear entre saídas intermediárias tanto do LFSR quanto do NFSR.

Figura 2.21 – Estrutura interna do Grain.



Fonte: (HELL; JOHANSSON; MEIER, 2007)

Quanto ao desempenho o sistema é capaz de fornecer um *bit* a cada ciclo de *clock*, sendo possível incrementar essa vazão adicionando mais *hardware*, o processo é feito de maneira simples, podendo alcançar até 16 *bits*/*clock*. Devido ao fato de que o sistema foi projetado para ser desenvolvido em *hardware*, seu desempenho é degradado quando implementado em *software*. Os autores conduziram um teste de desempenho do sistema sintetizado em diferentes famílias de FPGAs da Altera, foi considerado nessa análise a vazão em *Mbit/s (throughput)* e o número de elementos lógicos *(gate count)* para diferentes organizações de *bit/ciclo* fornecidos, o resultado pode ser observado no Quadro 2.5.

Quadro 2.5 – Análise de *throughput* e *gate count* para o sistema Grain v1.

t	*Gate count*	*Throughput (Mbit/s)*		
		MAX 3000A	*MAX II*	*Cyclone*
1	1450	49	200	282
2	1637	98	422	576
4	2010	196	632	872
8	2756	–	1184	1736
16	4248	–	2128	3136

Fonte: (HELL; JOHANSSON; MEIER, 2007)

Uma versão do Grain v1 que incluía a esse suporte a chaves de 128 *bits* com I.V. de 80 *bits* foi desenvolvida por Hell et al. (2006). Porém, essa versão não é mais recomendada pelo eSTREAM devido a uma falha de segurança encontrada por Dinur e Shamir (2011). Após a publicação desse artigo, foi proposta uma versão atualizada do Grain, chamada Grain 128a

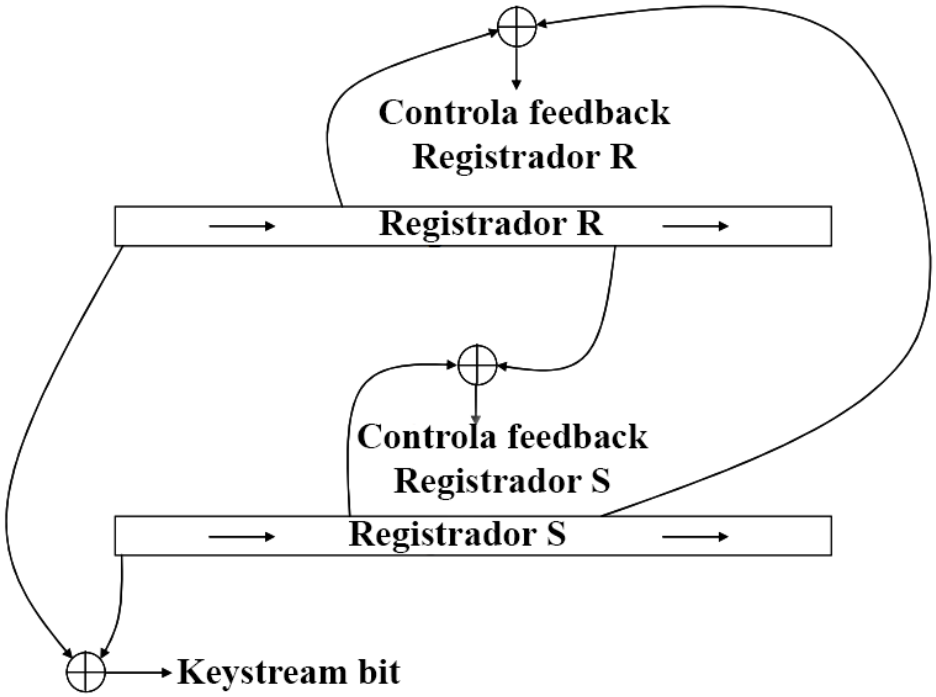
(ÅGREN et al., 2011), permitindo suporte a chave de 128 *bits* e melhorias de segurança, além de suporte a autenticação. O Grain 128a possui desempenho igual a metade do encontrado na versão Grain v1 e ocupa maior espaço de *hardware*, sendo necessários 2700 elementos lógicos *(gate count)* na sua construção. Atualmente, a variação Grain v1 é a recomendada pelo portfólio do eSTREAM (ECRYPT II, 2012c).

2.4.3 MICKEY

Desenvolvido por Steve Babbage e Matthew Dodd, o sistema de criptografia por fluxo MICKEY *(**M**utual **I**rregular **C**locking **KEY**stream generator)* foi projetado para estruturas de *hardware* com restrições de recursos, tendo assim baixa complexidade de construção, além de prover alto nível de segurança. Atualmente, o sistema recomendado no portfólio do projeto eSTREAM é a segunda versão do sistema, denominada MICKEY 2.0 (BABBAGE; DODD, 2006). A primeira versão foi retirada do portfólio após serem encontradas fraquezas em sua segurança no artigo de Gierlichs et al. (2008).

O sistema utiliza uma chave secreta (K) de 80 *bits* além de um vetor de inicialização (I.V.) também de 80 *bits*, sendo capaz de gerar uma cadeia com tamanho igual a 2^{40} *bits* com um único par (K, I.V.), podendo fixar o valor de K e alterar o valor do I.V. De modo geral, o sistema é composto por dois registradores de deslocamento, cada um com comprimento igual a 100 *bits*. O primeiro desses, chamado de **R** atua como registrador linear, já o segundo, chamado de **S** atua como registrador não linear. O MICKEY é dito ter *clock* irregular pois o sinal de *clock* do registrador R é influenciado pelo estado do registrador S, e vice-versa, essa característica contribui para a segurança do sistema contra variados tipos de ataque além de prover garantias no período dos registradores e a qualidade pseudoaleatória das sequências geradas (BABBAGE; DODD, 2006). A estrutura do sistema é mostrada na Figura 2.22.

Figura 2.22 – Estrutura interna do MICKEY 2.0.



Fonte: Adaptado de (BABBAGE; DODD, 2006)

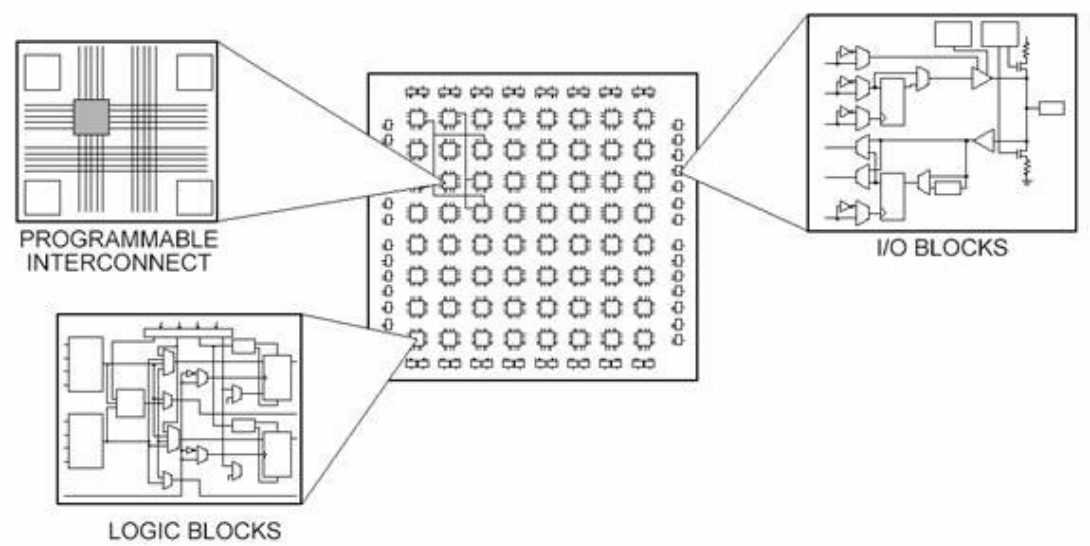
2.5 *Field Programmable Gate Arrays (FPGA)*

A Computação Reconfigurável é uma solução intermediária na resolução de problemas complexos, possibilitando combinar a velocidade do *hardware* com a flexibilidade do *software*. Uma arquitetura reconfigurável possui várias metas, entre elas o aumento de desempenho em relação ao *software*. Dentre os vários segmentos em relação às arquiteturas reconfiguráveis, destacam-se os Processadores Reconfiguráveis *soft-processors*, por exemplo o NIOS II da Altera. Estes processadores combinam as funções de um microprocessador com uma lógica reconfigurável e podem ser adaptados após o desenvolvimento (CASILLO, 2005).

O FPGA é composto estruturalmente por milhares de unidades lógicas idênticas. Neste aspecto, estas unidades lógicas podem ser vistas como componentes padrões que podem ser configurados independentemente e interconectados a partir de uma matriz de trilhas condutoras e chaves programáveis. A estrutura básica de um FPGA é formada pelo vetor de unidades lógicas, *Look-Up Tables (LUTs)*, blocos para entrada e saída de dados, e pela matriz de interconexão, que podem ser programados pelo usuário (KUON; TESSIER; ROSE, 2007), como mostrado na Figura 2.23. A configuração do FPGA é realizada por meio de um arquivo que descreve a estrutura e funcionalidade do componente a ser sintetizado. Por meio desse arquivo

é que a matriz de interconexão será fechada. Para gerar o arquivo binário, podem ser utilizadas ferramentas de *software* seguindo um determinado fluxo de projeto.

Figura 2.23 – Estrutura básica de um FPGA.

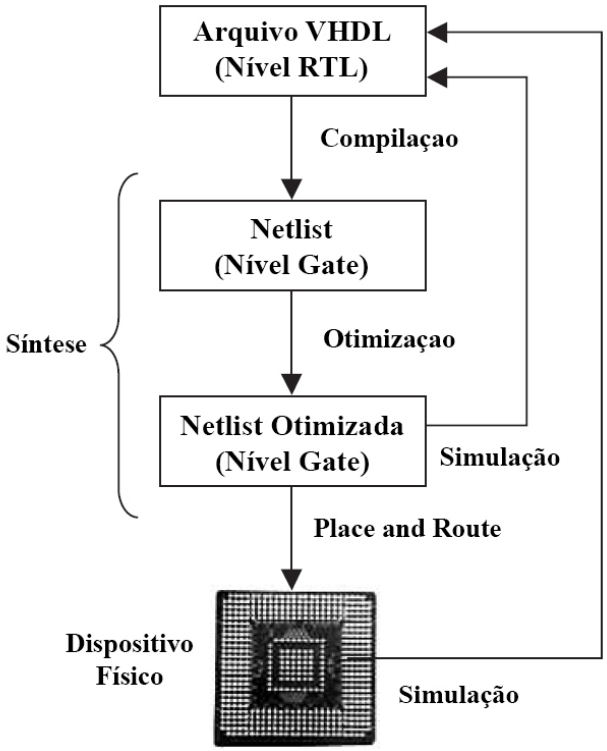


Fonte: (National Instruments, 2013)

Os recursos adicionais como *flip-flops*, multiplexadores, lógica de transporte, *carry* dedicado e portas lógicas, podem ser utilizados em conjunto com os LUTs para implementar diversas funções *booleanas*, multiplicadores e somadores, contadores, conversores serial-paralelo e paralelo-serial, além de memórias com grande variedade de tamanhos de palavra, fornecendo assim flexibilidade ao desenvolvimento (CASILLO, 2005). Para que um *hardware* seja desenvolvido em FPGA, podem ser utilizadas várias linguagens, chamadas de linguagens de descrição de *hardware* (HDL). Dentre as mais conhecidas pode-se destacar o VHDL. Criada pelo Departamento de Defesa Norte Americano *United States Department of Defense (DoD)* nos anos 80, VHDL é uma linguagem para descrição de *hardware* desenvolvida para descrever o comportamento eletrônico de um circuito ou sistema (PEDRONI, 2004). As aplicações básicas de VHDL são em FPGA's, *Complex Programmable Logic Device (CPLD's)* e ASIC's. O VHDL é uma linguagem que trabalha de forma concorrente, ao contrário de muitas linguagens de programação que são sequenciais (BOTROS, 2006). De modo geral, é uma linguagem adequada para se trabalhar com grandes projetos (PEDRONI, 2004). Em VHDL, assim como nas outras linguagens, os projetos são organizados em hierarquias (PEDRONI, 2004), isto é, interpretando componentes básicos como blocos que formam um bloco maior, que no caso é o projeto em si. Isso torna o código reutilizável, permitindo fazer alterações e reutilizar os componentes previamente desenvolvidos.

Após o processo de declaração dos códigos, o processo de desenvolvimento de uma aplicação em VHDL passa por um estágio de síntese dos circuitos e entidades declaradas, processo onde são utilizados algoritmos de otimização para que esses circuitos ocupem o menor espaço possível. É nessa fase também que ocorrem as simulações para validação do comportamento do componente desenvolvido. Caso tudo esteja correto, o componente é então entregue para a camada de controle físico, que será responsável então por carregar o componente na placa. Porém, se o resultado não for satisfatório nas simulações, o componente deve então ser reestruturado pelo desenvolvedor. O esquemático do fluxo de desenvolvimento é mostrado na Figura 2.24. Além de VHDL, existem outras linguagens tais como: Verilog, SystemVerilog e SystemC. Cada uma dessas tem seus pontos fortes e fracos, além do que, a utilização de qualquer uma delas no desenvolvimento dependerá do conhecimento por parte dos desenvolvedores e das ferramentas disponíveis. Um sistema pode ser modelado de maneira eficiente em qualquer uma destas linguagens (BOTROS, 2006).

Figura 2.24 – Fluxo de desenvolvimento de um projeto em VHDL.



Fonte: (PEDRONI, 2004)

3 TRABALHOS RELACIONADOS

Durante a fase de pesquisa bibliográfica executada no decorrer do projeto, conduziu-se uma revisão sistemática da literatura, onde por meio de uma *string* de busca montada a partir das principais palavras-chave relacionadas ao tema de pesquisa, buscou-se identificar o estado da arte em algoritmos de criptografia de imagens desenvolvidos em *hardware*, especificamente em FPGA. A busca foi realizada em repositórios eletrônicos científicos, considerando artigos que descrevem quais algoritmos de criptografia de imagens foram utilizados e qual a estratégia abordada nas suas implementações, além de descreverem o desenvolvimento desses algoritmos em HDLs. Partindo desse princípio, foram selecionadas as seguintes palavas-chave:

- Image Encryption;
- Picture Encryption;
- Image Cryptography;
- Picture Cryptography;
- Image Cipher;
- Picture Cipher;
- FPGA;
- Field Programmable Gate Array.

Após definição das palavras-chave, é necessária a combinação dessas de maneira lógica com o objetivo de realizar a busca nos repositórios eletrônicos. Assim, a seguinte expressão lógica foi construída:

```
("image encryption" OR "picture encryption" OR "image cryptography"
OR "picture cryptography" OR "image cipher" OR "picture cipher") AND (fpga
OR "field programmable gate array")
```

Durante a revisão foram considerados apenas artigos escritos em língua inglesa, devido essa ser o principal idioma utilizado no meio científico e nas comunicações mais importantes dessa área. Para execução da revisão foram utilizadas as seguintes bases de busca:

- IEEE Xplore <http://ieeexplore.ieee.org>

- Biblioteca digital da ACM <http://http://dl.acm.org>
- EI Compendex <http://www.engineeringvillage.com>
- Springer Link <http://www.springer.com>
- Science Direct <http://www.sciencedirect.com>
- Scopus <http://www.scopus.com>

Após finalização da fase de planejamento da Revisão Sistemática da Literatura (RSL), a *string* de busca foi pesquisada nas bases de dados selecionadas. Os resultados retornados foram salvos e em seguida filtrados em diferentes fases. A primeira fase da filtragem trata-se da remoção dos artigos/documentos irrelevantes, como por exemplo programação/agenda de conferências ou lista de artigos aprovados em eventos. Em seguida foram excluídas da união dos resultados das bases os artigos em duplicata. A próxima fase trata-se da exclusão dos artigos sem acesso integral por meio do *proxy* da Universidade Federal de Lavras. Apesar de não possuir acesso, esses artigos, por meio da análise de título e resumo, foram classificados como irrelevantes para o trabalho. O 4º passo é a remoção de artigos irrelevantes após leitura de seus títulos e *abstracts*. Por fim, a última fase de filtragem é a remoção dos artigos irrelevantes por meio da leitura de suas seções de introdução e conclusão. Após o processo de filtragem restaram 29 artigos a serem analisados na próxima fase da RSL. O Quadro 3.1 apresenta em detalhes a quantidade de artigos selecionados e descartados a cada fase do processo de filtragem.

Quadro 3.1 – Sumário de artigos filtrados por base científicas.

Base	**Inicial**	**Irrelevantes**	**Duplicatas**	**Sem Acesso**	**Seleção Primária**	**Seleção Secundária**	**Final**
IEEE Xplore	33	-3	0	0	-5	-7	**18**
ScienceDirect	37	-6	0	0	-17	-11	**3**
Scopus	50	-7	-30	-5	-3	-1	**4**
SpringerLink	47	-1	0	-19	-17	-6	**4**
EICompendex	42	-7	-34	-1	0	0	**0**
ACM	0	0	0	0	0	0	**0**
Total	209	-24	-64	-25	-42	-25	**29**

Fonte: Do autor (2017)

Após análise das publicações selecionadas na RSL notou-se uma tendência no crescimento atual no número de publicações relacionados ao tema e um número total de artigos

considerado baixo, esses dois fatores possivelmente indicam uma área com possibilidade de exploração e inovação. Quanto aos métodos atuais para criptografia e transmissão segura de imagens, observou-se que a maioria das publicações analisadas tratava tal problema com sistemas baseados em sistemas caóticos, como em (STOYANOV; KORDOV, 2014) e (HAI et al., 2015). Outras exploravam técnicas secundárias como autômatos celulares (RAJAGOPALA et al., 2014) e sistemas híbridos de algoritmos clássicos de criptografia de textos, como por exemplo o AES (SHAH; RAJA, 2015). No que diz respeito a plataformas de FPGA ambas as marcas Xilinx e Altera foram amplamente utilizadas. Outro aspecto importante encontrado em algumas implementações é uma fase inicial de desenvolvimento em *software* (majoritariamente Matlab), que serve também posteriormente como forma de validação da solução desenvolvida em *hardware*.

Diversos esquemas de criptografia foram propostos com o objetivo de transmitir dados de maneira segura, entretanto para a criptografia de imagens não é recomendada a utilização de algoritmos clássicos, como por exemplo, o AES e RSA. Essa restrição deve-se às características intrínsecas de uma imagem como por exemplo sua alta capacidade de armazenamento de dados e correlação entre *pixels*, esse problema é ressaltado ao tratar de imagens em meios de comunicação online (CHEN; MAO; CHUI, 2004). Uma das propostas atuais para criptografia de imagens, de um modo amplo, ou seja, não necessariamente em FPGA, é a utilização de sistemas caóticos e o aprimoramento de geradores para cifragem em fluxo, sendo utilizado inclusive combinação dessas duas técnicas, como em Shruthi, Sheela e Sathyanarayana (2014) e Rohith, Bhat e Sharma (2014). Sem dúvidas, o principal passo ao propor um novo método de criptografia ou comparar métodos já existentes é a fase de análise estatística. Essa fase é composta por diversos métodos, entre eles a análise da correlação entre *pixels* vizinhos, análise dos valores de entropia, a sensibilidade do sistema quanto à chave inicial e a análise de histograma.

O resultado da RSL mostrada não apresenta nenhum artigo que realiza a implementação de pelo menos um dos algoritmos de Perfil II do projeto eSTREAM aplicados a segurança de imagens digitais, assim, para garantir a qualidade da busca e do projeto foi montada uma nova *string* de busca mais refinada, sendo consideradas palavras-chave mais específicas. Dessa forma, a seguinte expressão lógica foi construída:

```
(trivium OR mickey OR grain OR estream OR ecrypt) AND (“image
encryption” OR “picture encryption” OR “image cryptography” OR “picture
cryptography” OR “image cipher” OR “picture cipher”)
```

Após processamento da *string* nas mesmas bases mostradas anteriormente, nenhum artigo relevante foi retornado. Dessa maneira é forte o indício de inovação desse projeto, e sua importância é evidente ao proporcionar à comunidade científica o resultado da qualidade desses algoritmos quando aplicados na criptografia de imagens. Com o sucesso desse trabalho, a simplicidade e eficiência das cifras simétricas poderão ser fundamentais na construção de sistemas de comunicação seguros para troca de imagens digitais.

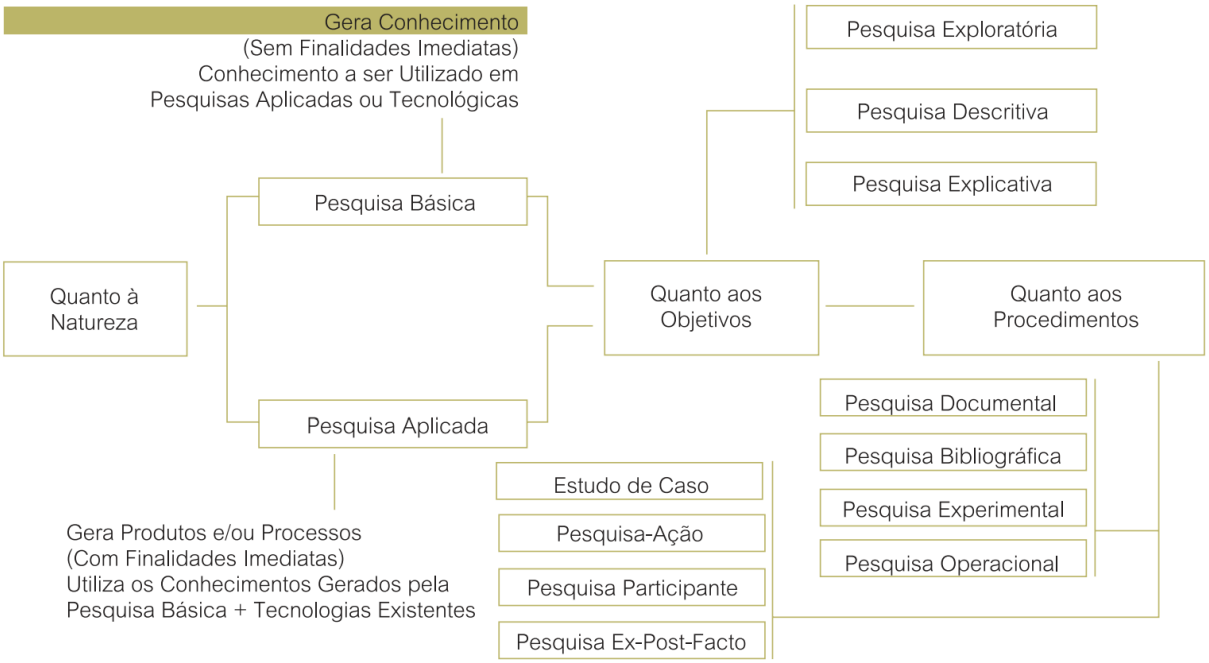
4 METODOLOGIA

Esse capítulo apresenta a classificação do presente projeto em relação ao tipo de pesquisa, em seguida são apresentados os métodos que guiaram o desenvolvimento do projeto, a base de dados e métricas para validação das implementações e as ferramentas de *hardware* e *software* necessárias ao projeto.

4.1 Tipo de Pesquisa

De acordo com Prodanov e Freitas (2013), uma pesquisa pode ser classificada quanto a sua natureza, seus objetivos, seus procedimentos e por último, quanto à forma de abordagem do problema. A Figura 4.1 mostra um esquemático de tais variações, em seguida cada classificação é sucintamente explicada.

Figura 4.1 – Tipos de pesquisa científica.



Fonte: (PRODANOV; FREITAS, 2013)

- **Quanto à natureza:** a pesquisa pode ser classificada como **pesquisa básica** se objetiva a geração de novos conhecimentos sem aplicação prática prevista ou como **pesquisa aplicada** quando objetiva geração de conhecimentos para aplicação prática voltada para solução de problemas específicos.

- **Quanto aos objetivos:** a pesquisa pode ser classificada como **pesquisa exploratória** se tem como finalidade proporcionar mais informações sobre o assunto investigado, as-

sumindo geralmente a forma de pesquisas bibliográficas ou estudos de caso. Também pode ser classificada como **pesquisa descritiva** se objetiva descrever as características de determinada população ou fenômeno ou o estabelecimento de relação entre variáveis, nesse tipo de pesquisa o pesquisador apenas observa, registra e descreve os dados, sem manipulá-los. Para coleta dos dados utiliza-se da entrevista, formulários, questionários, testes e a observação. Já em uma **pesquisa explicativa** o pesquisador procura explicar os porquês das coisas e suas causas, por meio do registro, análise, classificação e interpretação dos fenômenos observados.

- **Quanto aos procedimentos técnicos:** a pesquisa pode ser classificada como **pesquisa bibliográfica** quando a pesquisa é elaborada a partir de material já publicado, constituindo principalmente de livros, revistas, publicações em periódicos e artigos científicos, monografias, dissertações e teses, entre outros. Já na **pesquisa documental** diferentemente da pesquisa bibliográfica, que se utiliza a contribuição de documentos escritos por vários autores, essa baseia-se em documentos que ainda não receberam tratamento analítico. Na **pesquisa experimental** o pesquisador procura refazer as condições de um fato a ser estudado, selecionando as variáveis capazes de influenciá-lo, sendo então capaz de observar tal fato sob controle. É um tipo de pesquisa mais comum nas ciências tecnológicas e biológicas. As pesquisas do tipo **levantamento ou *survey*** ocorrem quando se envolve a interrogação direta das pessoas cujo comportamento deseja-se conhecer por meio de algum tipo de questionário. Já a **pesquisa de campo** é aquela com objetivo de conseguir informações e/ou conhecimentos sobre um determinado problema a ser solucionado ou hipótese a ser comprovada. Consiste na observação de fatos e fenômenos, na coleta de dados, e registro de variáveis presumidas relevantes para análise. No **estudo de caso** é realizada a coleta e análise de informações sobre um determinado indivíduo, família, um grupo ou comunidade, objetivando estudar aspectos variados de sua vida, de acordo com o assunto de pesquisa. Na **pesquisa *ex-post-facto***, o experimento ocorre após os fatos, nesse tipo de pesquisa são analisadas as situações que se desenvolvem naturalmente após algum acontecimento, buscando-se saber quais os possíveis relacionamentos entre as variáveis. A **pesquisa-ação** é aquela concebida e realizada em estreita associação com uma ação ou resolução de um problema coletivo, nesse tipo de pesquisa os pesquisadores e participantes se envolvem de maneira cooperativa no trabalho. Por último, a **pesquisa**

participante é aquela que se desenvolve a partir da interação entre pesquisadores e membros das situações investigadas.

- **Quanto a Abordagem do Problema**
  - **Pesquisa Quantitativa:** tipo de pesquisa que considera que tudo pode ser quantificável, ou seja, tudo pode ser traduzido em números, utiliza-se de recursos e técnicas estatísticas.
  - **Pesquisa Qualitativa:** considera que existe uma relação dinâmica entre o mundo real e o sujeito, que não pode ser traduzida em números, assim, este método não requer o uso de métodos e técnicas estatísticas.

Portanto, o presente trabalho pode ser classificado como aplicado quanto à natureza, pois objetiva a análise da qualidade de sistemas de criptografia para aplicação em troca segura de imagens entre pares comunicantes. Já quanto aos objetivos pode ser classificado como exploratório, pois proporciona mais familiaridade com o assunto de pesquisa. É experimental ao serem manipuladas variáveis para análise de suas influências no resultado obtido, e, por fim, pode ser classificado como quantitativo, pois os resultados obtidos serão avaliados segundo métricas que serão utilizados como indicativo de qualidade de cada sistema de criptografia.

4.2 Método de Pesquisa

O processo de desenvolvimento adotado na execução deste trabalho pode ser compreendido em quatro diferentes etapas, são elas: **1)** embasamento teórico, **2)** implementação em *software*, **3)** análise e validação I, **4)** implementação em *hardware*, **5)** análise e validação II e **6)** apresentação de resultados.

1) **Embasamento Teórico:** nessa fase é estudada a literatura relacionada à criptografia de imagens desenvolvidos em FPGA. Com base nesse estudo é então definido quais os algoritmos devem ser alvo de estudo deste trabalho, e quais são as métricas utilizadas na bibliografia para validação da qualidade de tais algoritmos. É também objetivo dessa fase a familiarização do autor com conceitos de *hardware* reconfigurável, fundamentos de imagem digital e técnicas de criptografia, sendo produto dessa fase o referencial teórico escrito no Capítulo 2 desse documento.

2) **Implementação em *Software*:** essa fase compreende os processos de estudo, adaptação e implementação dos algoritmos de Perfil II do Projeto eSTREAM em *software* utilizando a ferramenta de calculo numérico Matlab.

3) **Análise e Validação I:** após o desenvolvimento em *software* dos algoritmos de criptografia, inicia-se o processo de validação dessas implementações por meio da cifragem e decifragem de diversas imagens da base de dados a ser citada na Seção 4.3. Também nessa fase ocorre a análise da qualidade desses sistemas de criptografia quando aplicados a essas imagens, por meio das métricas que serão apresentadas na Seção 4.4.

4) **Implementação em *Hardware*:** após a validação e análise das implementações dos algoritmos em *software*, o próximo passo consiste no estudo e desenvolvimento de tais algoritmos em *hardware*, nesse caso em um FPGA e utilizando a linguagem de descrição de *hardware* VHDL.

5) **Análise e Validação II:** o último processo de análise envolve a validação dos algoritmos implementados em *hardware*, utilizando para isso a implementação física no FPGA e a simulação dos algoritmos em ferramentas de simulação específicas. Após validação, inicia-se o processo de análise do desempenho entre as implementações em *hardware*, avaliando a utilização de recursos do FPGA e vazão máxima *(bits/clock)* de cada implementação.

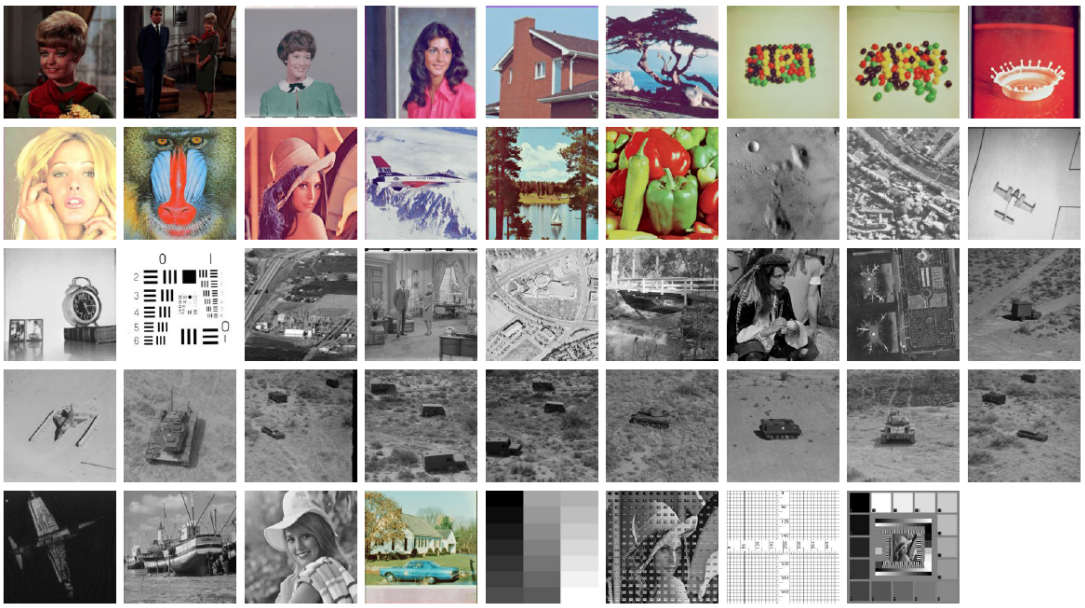
6) **Apresentação de Resultados:** os resultados obtidos ao final do projeto são expostos para a comunidade por meio da monografia e artigos submetidos em periódicos relacionados à área de pesquisa.

4.3 Bases de Dados e Experimentos

Para análise da qualidade dos algoritmos de criptografia, torna-se necessário a reprodutibilidade dos experimentos. Uma forma de garantir isso é utilizar um conjunto de dados de entrada padronizados comumente chamado de *dataset*. Nesse trabalho foi escolhido o *dataset* ***USC-SIPI image database, miscellaneous volume*** (University of Southern California, 2017). Esse é composto por um total de 44 imagens, sendo 16 coloridas e 28 em escala de cinza, 14 imagens possuem dimensão igual a 256 x 256 *pixels*, 26 iguais a 512 x 512 e 4 iguais a 1024 x 1024. A Figura 4.2 exibe a miniatura de todas as imagens desse *dataset*. Esse *dataset* foi

escolhido devido a sua utilização em trabalhos que utilizem técnicas de processamento de imagens, além de possuir imagens com características diversas, proporcionando com isso dados de entrada heterogêneos.

Figura 4.2 – Conjunto de imagens *miscellaneous*.



Fonte: Do autor (2017)

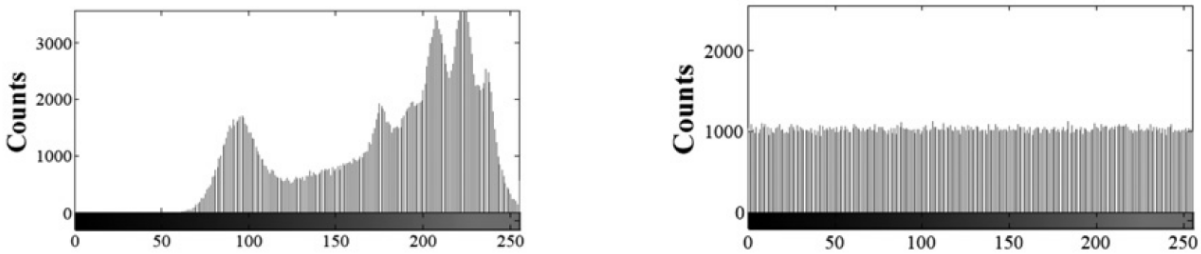
4.4 Avaliação dos Resultados

Os algoritmos de criptografia serão avaliados quanto ao desempenho em *hardware* e quanto à qualidade desses ao cifrar imagens digitais. Entre as implementações em *hardware* será analisada a eficiência da utilização de recursos do FPGA e a vazão máxima em *bits/clock* alcançada no processo de cifragem e decifragem. A qualidade dos algoritmos na aplicação de criptografia de imagens será mensurada por meio da união das mais significativas métricas utilizadas nos trabalhos referenciados no Capítulo 3, são elas:

- **Análise Visual:** após cifragem de qualquer imagem espera-se de um algoritmo de criptografia duas características. A primeira é que a imagem cifrada não contenha nenhum traço de informação da imagem original, isso é, se pareça o máximo possível com uma iamgem sem informação ou sentido aparente. A segunda característica é que após o processo de decifragem com a chave correspondente, a imagem decifrada seja exatamente igual à original. Este método de análise é subjetivo e sem a presença de outras métricas não é suficiente para garantir a qualidade dos algoritmos analisados.

- **Análise de Histograma:** um histograma descreve de forma gráfica a distribuição estatística da intensidade das cores de uma imagem. Esquemas de criptografia seguros são caracterizados por um histograma uniforme, como mostrado na Figura 4.3 (MANSINGKA et al., 2014). Além da análise visual, calcula-se o valor médio de intensidade dos *pixels* de cada imagem a fim de comprovar o comportamento uniforme dessa.

Figura 4.3 – Exemplo de histograma.



Legenda: Histograma de canal vermelho (RGB) de imagem sem criptografia (esquerda) e da mesma imagem criptografada (direita).
Fonte: (MANSINGKA et al., 2014)

- **Análise de Entropia:** entropia é a medida da previsibilidade de uma fonte aleatória, ela expressa o grau de incerteza em um sistema. Para uma fonte binária S que produz 2^8 símbolos cada um com iguais probabilidades, a entropia pode ser definida como na Equação 4.1, onde $P(S_i)$ é a probabilidade do símbolo S_i (RAMIREZ-TORRES; MURGUIA; MEJIA-CARLOS, 2014).

$$Entropy = -\sum_{i=1}^{2^8} P(S_i) \log 2P(S_i) \tag{4.1}$$

Em uma fonte realmente aleatória capaz de produzir 2^N símbolos diferentes, o valor da entropia é igual a N (RAMIREZ-TORRES; MURGUIA; MEJIA-CARLOS, 2014). Ou seja, para imagens cifradas com 8 *bits* de resolução por canal, isto é, imagens com 256 níveis (símbolos) diferentes, o valor de entropia desejado é igual a 8 pois $2^8 = 256$. No artigo de Merah, Ali-Pacha e Hadj-Said (2015), é calculada a entropia para cada canal RGB individualmente, comparando o resultado entre a imagem original e a cifrada.

- **Análise de Correlação:** uma baixa correlação entre *pixels* adjacentes indica um sistema de criptografia robusto. A correlação entre dois *pixels* adjacentes x e y pode ser dada pela Equação 4.2, onde $cov(x,y)$ representa a covariância entre os *pixels*, $D(x)$ e $D(y)$

representam respectivamente a variância de x e de y, e M e N correspondem a largura e à altura da imagem (MERAH; ALI-PACHA; HADJ-SAID, 2015).

$$\gamma_{xy} = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{4.2a}$$

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \frac{1}{N}\sum_{j=1}^{n} x_j\right)\left(y_i - \frac{1}{N}\sum_{j=1}^{n} y_j\right) \tag{4.2b}$$

$$D(x) = \frac{1}{n}\sum_{i=1}^{N}\left(x_i - \frac{1}{n}\sum_{j=1}^{N} x_j\right)^2 \tag{4.2c}$$

$$D(y) = \frac{1}{n}\sum_{i=1}^{N}\left(y_i - \frac{1}{n}\sum_{j=1}^{N} y_j\right)^2 \tag{4.2d}$$

Em uma imagem com alta correlação de *pixels*, os valores de *pixels* adjacentes são muito próximos uns dos outros, ou seja, a cor exibida é praticamente igual para as duas posições na imagem. A alta correlação em uma imagem cifrada é ruim para a segurança dessa, pois, proporciona a um eventual atacante uma silhueta do que era a imagem original. Na Figura 4.4, a imagem à esquerda exibe detalhe ampliado da imagem à direita, demonstrando assim a alta correlação em uma imagem sem criptografia, ou seja, *pixels* adjacentes possuem valores muito próximos.

Figura 4.4 – Alta correlação entre *pixels* adjacentes.

Fonte: Do autor (2017)

- **Análise de Sensibilidade** para que um algoritmo de criptografia possa ser seguro é desejável que ao realizar uma pequena mudança na chave de cifragem (alterar apenas um *bit*) a correlação entre a imagem cifrada com a chave original (K1) e a imagem cifrada com a chave alterada (K2), seja próximo de 0. O mesmo processo vale para o processo de decifragem. De modo geral pode-se dizer que é desejável que uma pequena mudança na chave gere uma enorme mudança na imagem cifrada.

- **Análise Diferencial:** pequenas alterações na imagem original devem resultar em alterações significativas na imagem cifrada para que seja evitada qualquer relação estatística entre a entrada e a saída do sistema. A sensibilidade de um sistema de criptografia quanto aos dados de entrada pode ser calculado por duas métricas: *Number of Pixels Change Rate (NPCR)* e *Unified Average Changing Intensity (UACI)*. De acordo com Mansingka et al. (2014) tais métricas podem ser definidas como:

  - **NPCR:** representa a taxa de *pixels* modificados na imagem cifrada enquanto apenas um *pixel* é alterado na imagem original. Assumindo $C1$ e $C2$ como imagens cifradas provenientes de duas imagens com apenas um *pixel* de diferença respectivamente e $D(i,j)$ como à imagem decifrada, o cálculo do NPCR pode ser realizado por meio da Equação 4.3.

$$D(i,j) = \begin{cases} 0, & C_1(i,j) = C_2(i,j) \\ 1, & C_1(i,j) \neq C_2(i,j) \end{cases} \tag{4.3a}$$

$$NPCR = \frac{1}{MxN} \sum_{i=1}^{M} \sum_{j=1}^{N} D(i,j) \tag{4.3b}$$

  - **UACI:** representa a diferença de intensidade média entre duas imagens cifradas, cujas imagens originais são diferentes por apenas um *pixel*, o calculo é realizado por meio da Equação 4.4.

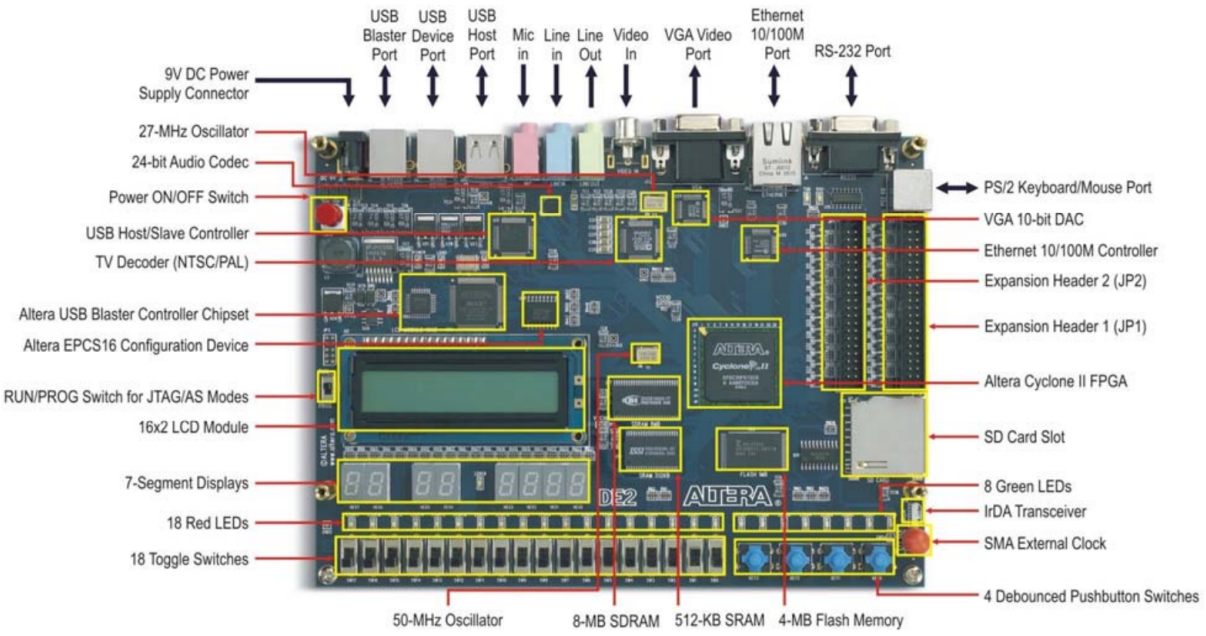
$$UACI = \frac{1}{MxN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\frac{|C_1(i,j) - C_2(i,j)|}{255} \right) x100 \tag{4.4}$$

4.5 *Hardware* e *Software* utilizados

No projeto foi utilizado um notebook HP Pavilion n070br com processador Intel Core I7-4500U, 8GB de RAM e Windows 7 Professional SP1 para desenvolvimento dos sistemas de criptografia em *software* e *hardware*. Como plataforma de desenvolvimento em *hardware* foi escolhida a placa de desenvolvimento Altera DE2, mostrada na Figura 4.5, equipada com o FPGA Altera Cyclone II EP2C35F672C6. Esse FPGA possui 33.216 elementos lógicos, e a placa de desenvolvimento conta com diversos elementos como *pushbuttons*, LEDs, *clocks* de 50MHz e 27MHZ, além de 80 pinos de expansão de uso geral (GPIO).

No processo de desenvolvimento em *software* dos sistemas de criptografia do projeto eSTREAM, foi utilizado o ambiente de cálculo numérico Matlab R2015a. Já no processo de desenvolvimento em *hardware* a ferramenta Altera Quartus II Web Edition 13.0 foi escolhida para desenvolvimento em linguagem VHDL e para simulação o *software* Altera ModelSim 10.1d. Ambos os recursos de *software* e *hardware* estiveram disponíveis para a realização do trabalho, não sendo assim necessário a aquisição de nenhum desses.

Figura 4.5 – Placa de desenvolvimento Altera DE2.



Fonte: (Altera Corporation, 2006)

5 RESULTADOS

Esse capítulo mostra os resultados alcançados no projeto, na Seção 5.1 é apresentada a plataforma de análise automatizada para qualidade dos algoritmos, já na Seção 5.2 são apresentadas as soluções desenvolvidas em FPGA, por último, na Seção 5.3, é apresentada a análise de qualidade de cada sistema de criptografia perante cada métrica determinada na Seção 4.4.

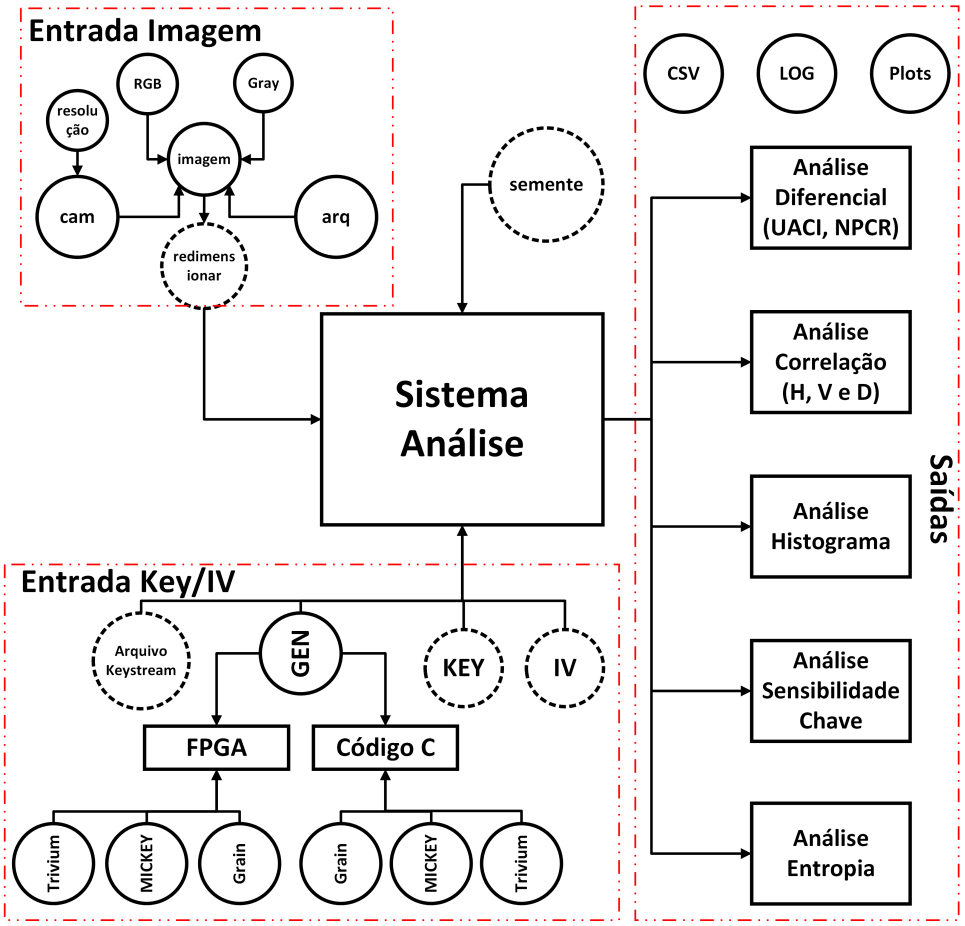
5.1 Plataforma de Análises

Após definição do *dataset* e das métricas utilizadas na literatura para determinar se os sistemas são ou não adequados para o processo de criptografia de imagens, foram criados *scripts* com auxílio da ferramenta Matlab R2015a (Mathworks, 2015) com o objetivo de tornar o processo de análise automatizado. Esses *scripts* são capazes de realizar o processo de cifragem/-decifragem, além de calcular as métricas estabelecidas. Após cada execução, os resultados numéricos são salvos em arquivos ".csv" para posterior síntese estatística. Para cada execução, são salvas as imagens cifradas, decifradas e gráficos (histogramas, correlação e sensibilidade), além das sequências de *bits* pseudoaleatórios gerados *(keystream)* e o gerador aleatório do Matlab. A implementação dos sistemas de criptografia analisados foi realizada em *software* a partir de adaptações nos códigos ".c" disponibilizados pelos seus respectivos autores (ECRYPT II, 2012b).

A plataforma de análise desenvolvida é capaz de processar imagens provenientes tanto de arquivos quanto realizar a captura por meio de uma *webcam*. O usuário têm à disposição parâmetros para redimensionar as imagens ou configurar qual a resolução de captura será utilizada, além de escolher se deseja processar a imagem colorida ou em escala de cinza. Para garantir a repetibilidade de um experimento, é possível importar para o sistema o gerador aleatório do Matlab (semente) a ser utilizado. A plataforma fornece ainda a capacidade de escolha de qual entre os três algoritmos deverá ser utilizado no processo de análise, seja em *software*, ou em *hardware* por meio da comunicação via porta serial com o sistema descrito em FPGA. Um esquemático do sistema de análise é mostrado na Figura 5.1.

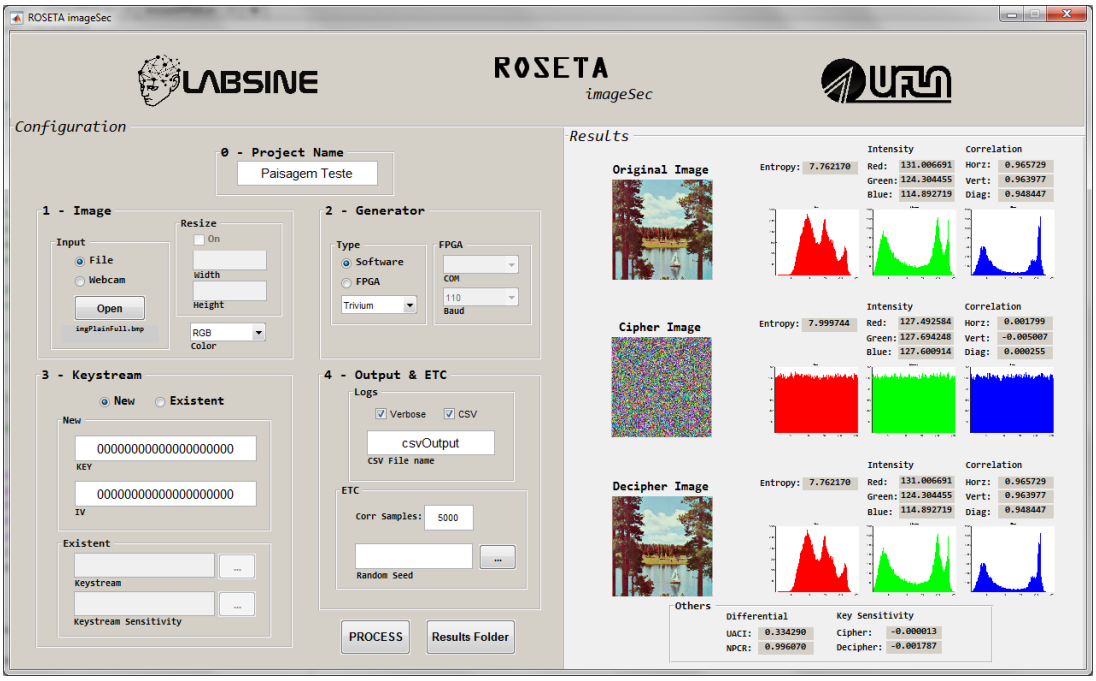
Essa plataforma de análise pode ser executada em linha de comando, porém, como forma de facilitar sua utilização foi criada a interface gráfica mostrada na Figura 5.2, que proporciona de maneira visual total controle da plataforma, permitindo executar as mesmas funcionalidades previstas em linha de comando.

Figura 5.1 – Esquemático do sistema de análise automatizada.



Fonte: Do autor (2017)

Figura 5.2 – Interface gráfica do sistema de análise automatizada.



Fonte: Do autor (2017)

5.2 Desenvolvimento em *Hardware*

Para validação dos sistemas de criptografia em *hardware*, ambos foram descritos em VHDL e sintetizados em FPGA. Cada um dos três sistemas é composto por um arquivo que descreve o gerador propriamente dito ("TRIVIUM_Generator", "GRAIN_Generator"e "MICKEY_Generator"), além de um *hardware* controlador, que é responsável por gerir o processo de *reset* e requisição de novos *bits* pseudoaleatórios para o respectivo gerador, são eles: ("TRIVIUM_TopLevel", "GRAIN_TopLevel"e "MICKEY_TopLevel"). Cada um desses sistemas possui uma particularidade no que diz respeito a número de ciclos necessários para *reset* e tamanho dos vetores de Key e I.V. A partir desta premissa foram criados um arquivo de configuração para cada um desses sistemas, são eles: ("TRIVIUM_Functions", "GRAIN_Functions"e "MICKEY_Functions"). Todos os componentes controladores dos sistemas são agrupados em um arquivo principal, chamado "ROSETA", que é quem controla as requisições recebidas via interface serial, seleciona qual sistema deve ser ativado e, por fim, envia via serial os *bytes* pseudoaleatórios gerados. A implementação de cada um desses módulos e o resultado das simulações de cada um serão discutidos a seguir.

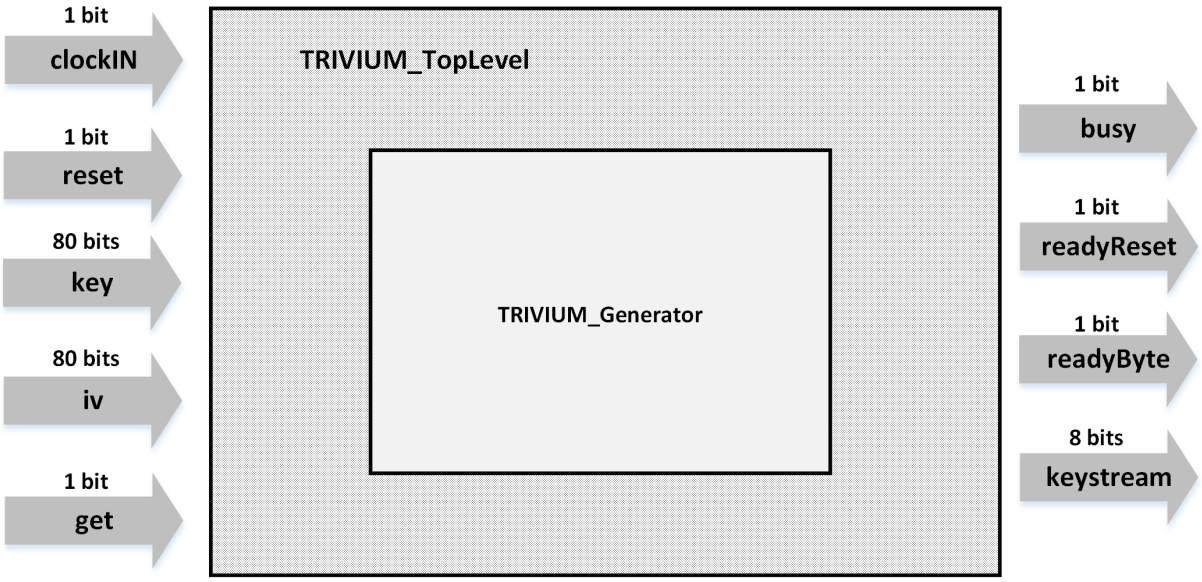
5.2.1 Trivium

O *hardware* que descreve o algoritmo Trivium é composto basicamente de dois blocos como mostrado na Figura 5.3, são eles:

- ***Trivium_TopLevel:*** módulo principal para controle do gerador, é responsável pelo processo de *reset* e inicialização desse. Além disso é por meio dele que ocorre a solicitação de geração de *bits* pseudoaleatórios ao módulo gerador “TRIVIUM_Generator”. Por meio de sua operação e controle são solicitados oito *bits* ao gerador. Assim, ao completar o recebimento de todos esses, envia em seu barramento de saída *(“keystream”)* um *byte* pseudoaleatório a cada requisição de geração feita no seu pino *“get”*.

- ***Trivium_Generator:*** esse módulo é o responsável de fato pela geração de *keystream*. A cada ciclo de *clock* um *bit* é gerado e enviado para o módulo *TopLevel*.

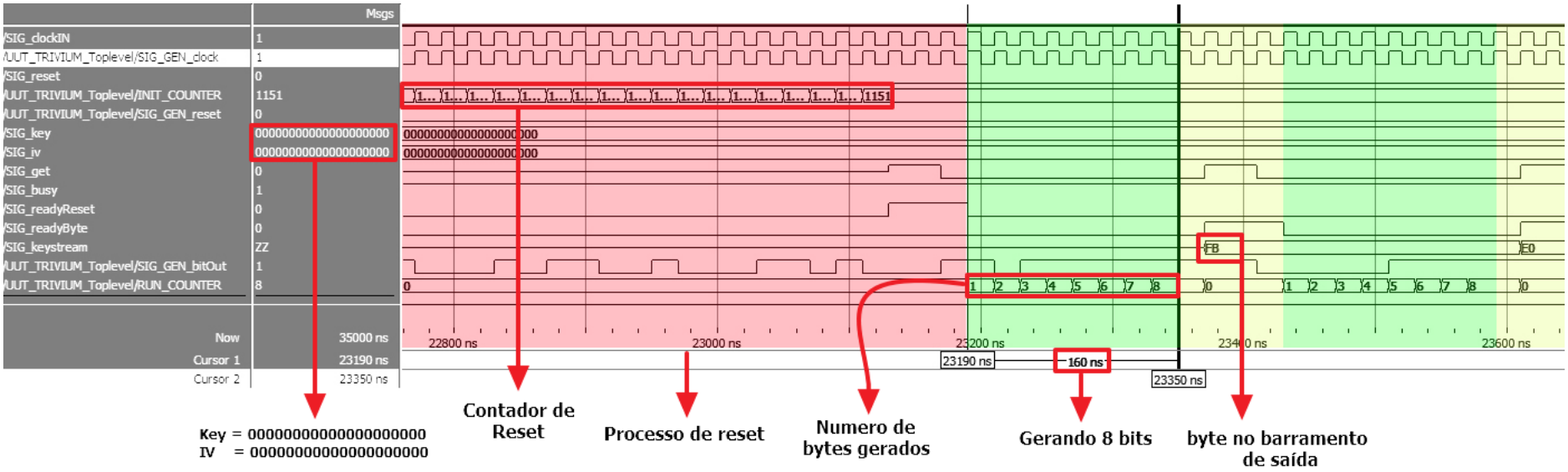
Figura 5.3 – Esquemático do sistema Trivium em FPGA.



Fonte: Do autor (2017)

O sistema Trivium foi validado fisicamente após síntese e também por meio de simulações na ferramenta ModelSim - Altera, um exemplo dessa última é mostrado na Figura 5.4. Nela é mostrado em detalhes os ciclos de operação do *hardware*, isto é, o ciclo de *reset* e de geração de *bytes*, além disso também é exibido o tempo necessário para a geração destes *bytes*.

Figura 5.4 – Ondas resultantes da simulação do *hardware* Trivium.



Fonte: Do autor (2017)

Após síntese, verificou-se também qual o consumo de elementos lógicos desses componentes, esses resultados são mostrados na Tabela 5.1. Verificando essa tabela observa-se que o sistema Trivium utiliza um número muito pequeno de recursos do FPGA onde foi sintetizado, comprovando de fato a eficiência para sistemas com restrição de espaço.

Tabela 5.1 – Recursos utilizados após síntese do sistema Trivium em FPGA.

Entidade	**Elem. Combinacionais**	**Registradores**	**Memória *(bits)***	**Pinos**
TRIVIUM_TopLevel	68	212	0	174
TRIVIUM_Generator	293	288	0	0
TOTAL	**361**	**500**	**0**	**174**

Fonte: Do autor (2017)

Para analisar qual o desempenho desse componente em relação à vazão de *bits* pseudo-aleatórios, a Tabela 5.2 foi montada. Após análise, foi observado, em simulação na ferramenta ModelSim - Altera, que o sistema Trivium implementado possui frequência de operação máxima igual a 306 MHz, vale ressaltar que a implementação escolhida é capaz de gerar apenas 1 *bit* por ciclo de *clock*. Uma maneira de ilustrar o desempenho encontrado é utilizando o ambiente de transmissão de vídeo em tempo real. Considerando imagens (quadros) com 24 *bits* de profundidade (cada *pixel* é representado por 3 *bytes* - RGB), observou-se que o sistema é capaz de estabelecer um fluxo de transmissão de imagens (vídeo) em resolução HD, i.e. 1280 x 720 *pixels*, a uma taxa de 14,5 quadros por segundo. Ou caso as restrições de aplicação permitam, o sistema entrega em resolução SD, i.e. 640 x 480 *pixels*, 43,5 quadros por segundo.

Tabela 5.2 – Análise de *throughput* do sistema Trivium após síntese em FPGA.

***bits*/ciclo**	***clock* max**	***throughput* max**	**vídeo HD**	**vídeo SD**
1	306 MHz	306 Gbps	14,5 FPS	43,5 FPS

Fonte: Do autor (2017)

5.2.2 Grain

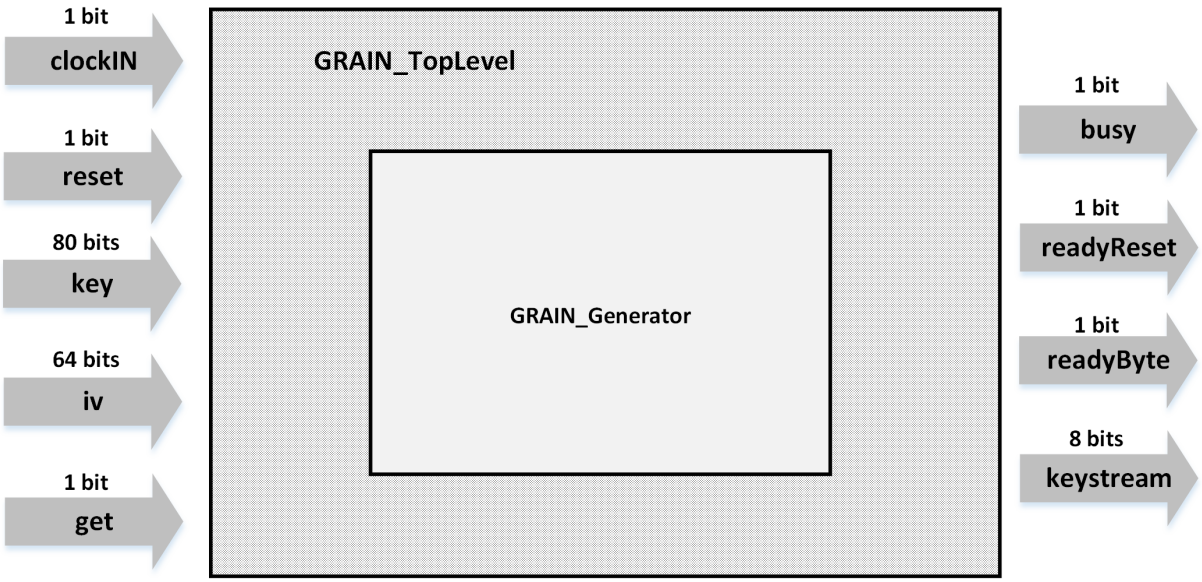
O *hardware* que descreve o algoritmo Grain possui organização da arquitetura igual a desenvolvida para o sistema Trivium, ou seja, é também composto de dois blocos como mostrado na Figura 5.5, são eles:

- ***Grain_TopLevel:*** módulo principal para controle do gerador, é responsável pelo processo de *reset* e inicialização desse. Além disso é por meio dele que ocorre a solicitação

de geração de *bits* pseudoaleatórios ao módulo gerador “GRAIN_Generator”. Por meio de sua operação e controle são solicitados oito *bits* ao gerador. Assim, ao completar o recebimento de todos esses, envia em seu barramento de saída *(“keystream”)* um *byte* pseudoaleatório a cada requisição de geração feita no seu pino *“get”*.

- ***Grain_Generator:*** esse módulo é o responsável de fato pela geração de *keystream*. A cada ciclo de *clock* um *bit* é gerado e enviado para o módulo *TopLevel*.

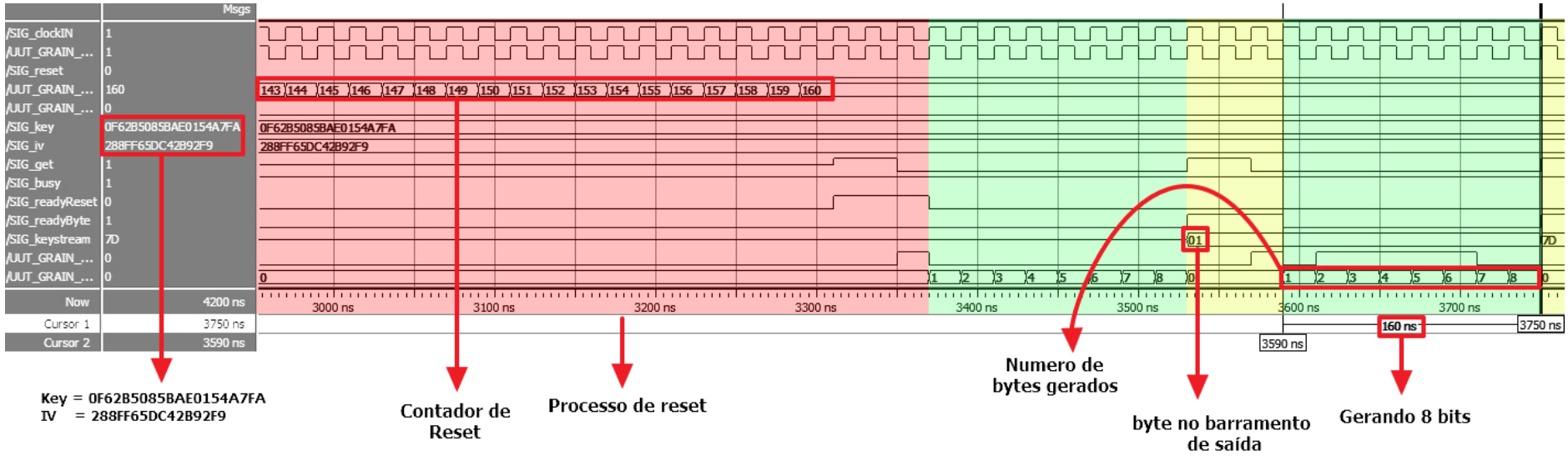
Figura 5.5 – Esquemático do sistema Grain em FPGA.



Fonte: Do autor (2017)

O sistema Grain foi validado fisicamente após síntese e também por meio de simulações na ferramenta ModelSim - Altera, um exemplo dessa última é mostrado na Figura 5.6.

Figura 5.6 – Ondas resultantes da simulação do *hardware* Grain.



Fonte: Do autor (2017)

Após síntese, verificou-se, assim como no componente Trivium, um baixo consumo de elementos lógicos, esses resultados são mostrados na Tabela 5.3.

Tabela 5.3 – Recursos utilizados após síntese do sistema Grain em FPGA.

Entidade	**Elem. Combinacionais**	**Registradores**	**Memória *(bits)***	**Pinos**
GRAIN_TopLevel	68	194	0	158
GRAIN_Generator	183	161	0	0
TOTAL	**251**	**355**	**0**	**158**

Fonte: Do autor (2017)

Na análise de vazão de *bits* pseudoaleatório desse componente a Tabela 5.4 foi montada. Foi observado, em simulação na ferramenta ModelSim - Altera, que o sistema Grain implementado possui frequência de operação máxima igual a 251 MHz. Aplicando no cenário de transmissão de vídeo em tempo real observou-se que o sistema é capaz de estabelecer um fluxo de transmissão de imagens (vídeo) em resolução HD, i.e. 1280 x 720 *pixels*, a uma taxa de 11,8 quadros por segundo, ou então 35,6 quadros por segundo em resolução SD, i.e. 640 x 480 *pixels*.

Tabela 5.4 – Análise de *throughput* do sistema Grain após síntese em FPGA.

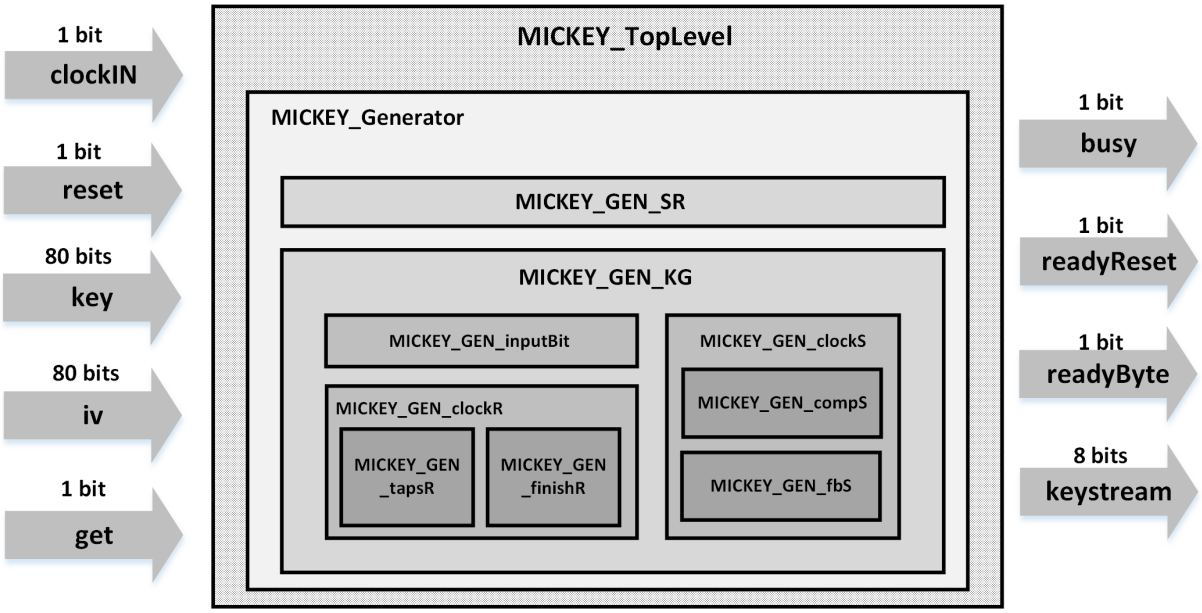
***bits*/ciclo**	***clock* max**	***throughput* max**	**vídeo HD**	**vídeo SD**
1	251 MHz	251 Gbps	11,8 FPS	35,6 FPS

Fonte: Do autor (2017)

5.2.3 MICKEY

O sistema MICKEY possui configuração semelhante aos dois mostrados anteriormente, porém aqui o módulo “Generator” é subdividido em outros módulos devido a complexidade do código que descreve o sistema. A implementação utilizada é baseada no trabalho de Nydia (2008). A Figura 5.7 exibe a arquitetura desse sistema.

Figura 5.7 – Esquemático do sistema MICKEY em FPGA.



Fonte: Do autor (2017)

São módulos implementados no sistema:

- ***MICKEY_TopLevel:*** módulo principal para controle do gerador, é responsável pelo processo de *reset* e inicialização desse. Além disso é por meio dele que ocorre a solicitação de geração de *bits* pseudoaleatórios ao módulo gerador "MICKEY_Generator". Por meio de sua operação e controle são solicitados oito *bits* ao gerador. Assim, ao completar o recebimento de todos esses, envia em seu barramento de saída *("keystream")* um *byte* pseudoaleatório a cada requisição de geração feita no seu pino *"get"*.

- ***MICKEY_Generator:*** esse módulo é o responsável de fato pela geração de *keystream*. A cada ciclo de *clock* um *bit* é gerado e enviado para o módulo *TopLevel*. Esse *bit* gerado é proveniente da operação XOR entre os *bits* da posição "0" dos registradores "R" e "S".

- ***MICKEY_GEN_SR:*** módulo responsável por atualizar os vetores "S" e "R" apresentados na descrição do sistema MICKEY proposto por Babbage e Dodd (2006).

- ***MICKEY_GEN_clocksKG:*** responsável por calcular o valor dos sinais de controle "CONTROL_BIT_R" e "CONTROL_BIT_S", além de gerenciar os componentes "InputBit", "clockR" e "clockS".

- ***MICKEY_GEN_inputBit:*** responsável por definir os valores dos sinais “INPUT_BIT_R” e “INPUT_BIT_S”.

- ***MICKEY_GEN_clockR:*** calcula o valor do sinal “FEEDBACK_BIT” para o registrador “R”, além de coordenar os componentes “tapsR” e “finishR”.

- ***MICKEY_GEN_tapsR:*** define o registrador “R” já com os *taps* preestabelecidos.

- ***MICKEY_GEN_finishR:*** gera o valor de saída definitivo do registrador “R”.

- ***MICKEY_GEN_clockS:*** calcula o valor do sinal “FEEDBACK_BIT” para o registrador “S”, além de coordenar os componentes “compS” e “fbS”.

- ***MICKEY_GEN_compS:*** armazena os valores dos vetores “COMP0” e “COMP1”, para então a partir deles calcular o valor intermediário de saída do registrador “S”.

- ***MICKEY_GEN_fbS:*** armazena os valores dos vetores “FB0” e “FB1”, para que juntamente com o valor intermediário calculado no componente “GEN_compS”, gere valor de saída definitivo do registrador “S”.

Na análise de performance de vazão e *bits* pseudoaleatório desse componente a Tabela 5.5 foi montada. Foi observado, em simulação na ferramenta ModelSim - Altera, que o sistema MICKEY implementado possui frequência de operação máxima igual a 166 MHz. Aplicando no cenário de transmissão de vídeo em tempo real observou-se que o sistema é capaz de estabelecer um fluxo de transmissão de imagens (vídeo) em resolução HD, i.e. 1280 x 720 *pixels*, a uma taxa de 7,8 quadros por segundo, ou então 23,6 quadros por segundo em resolução SD, i.e. 640 x 480 *pixels*.

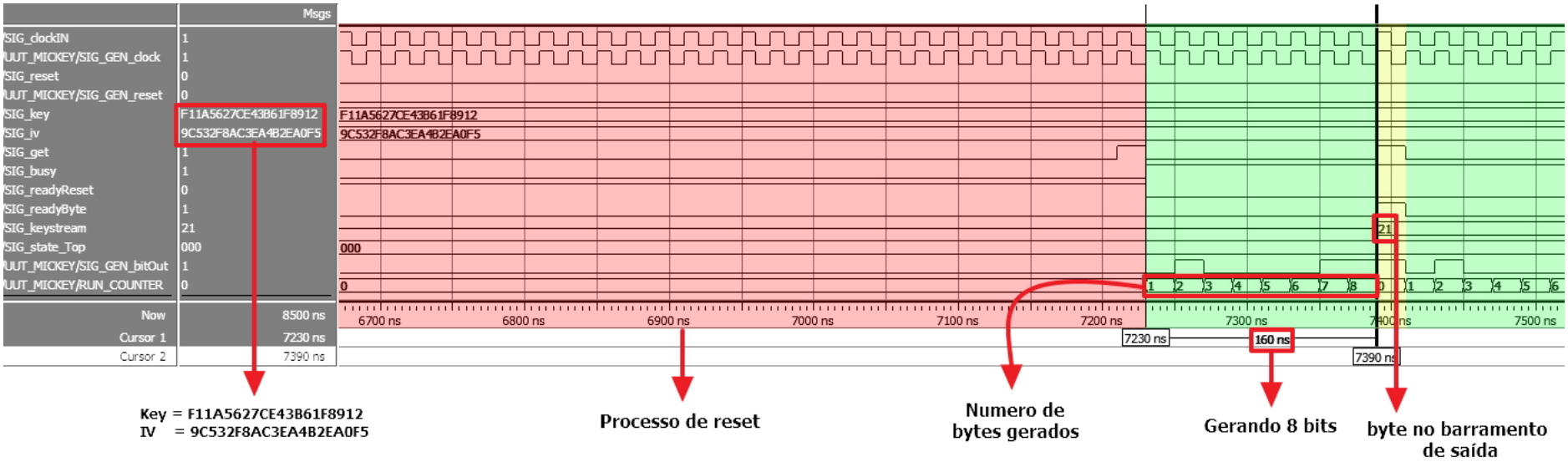
Tabela 5.5 – Análise de *throughput* do sistema MICKEY após síntese em FPGA.

***bits*/ciclo**	***clock* max**	***throughput* max**	**vídeo HD**	**vídeo SD**
1	166 MHz	166 Gbps	7,8 FPS	23,6 FPS

Fonte: Do autor (2017)

O sistema MICKEY foi validado fisicamente após síntese e também por meio de simulações na ferramenta ModelSim - Altera, um exemplo dessa última é mostrado na Figura 5.8.

Figura 5.8 – Ondas resultantes da simulação do *hardware* MICKEY.



Fonte: Do autor (2017)

Após síntese, verificou-se, um maior consumo de elementos lógicos quando comparado aos dois sistemas apresentados anteriormente, esses resultados são mostrados na Tabela 5.6.

Tabela 5.6 – Recursos utilizados após síntese do sistema MICKEY em FPGA.

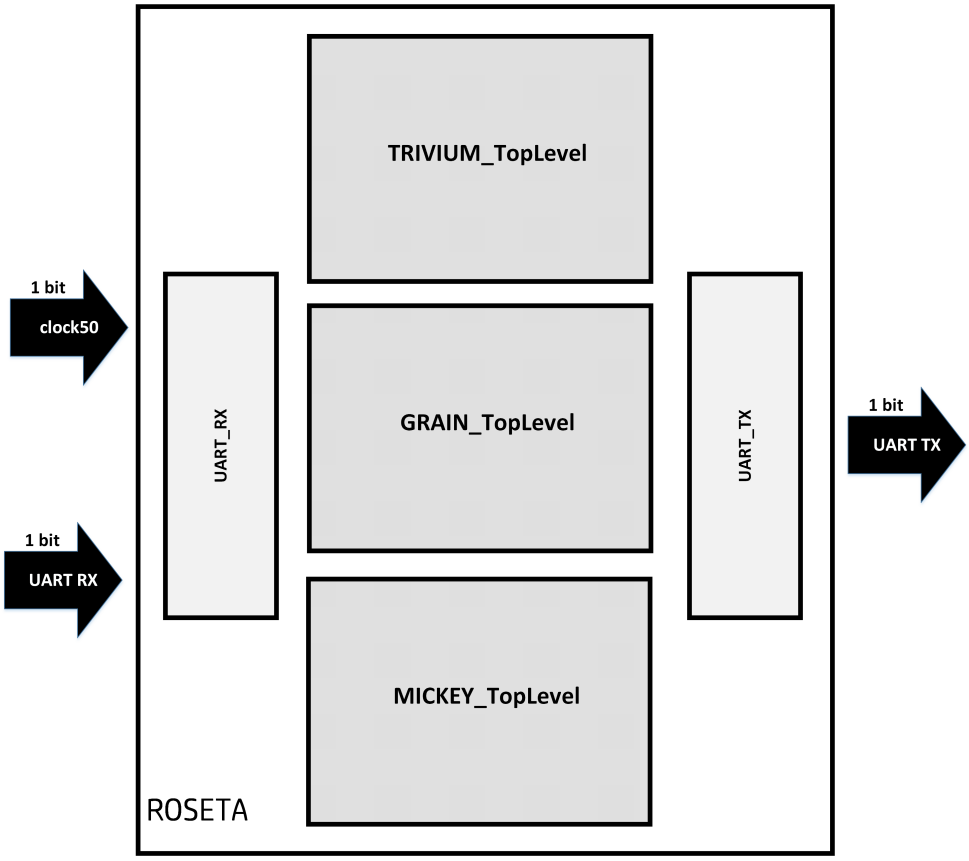
Entidade	**Elem. Combinacionais**	**Registradores**	**Memória *(bits)***	**Pinos**
TopLevel	201	37	0	177
Generator	301	212	0	0
GEN_clockKG	2	0	0	0
GEN_clockR	1	0	0	0
GEN_finishR	100	0	0	0
GEN_clockS	1	0	0	0
GEN_compS	77	0	0	0
GEN_fbS	76	0	0	0
GEN_SR	0	0	0	0
TOTAL	**759**	**249**	**0**	**177**

Fonte: Do autor (2017)

5.2.4 ROSETA

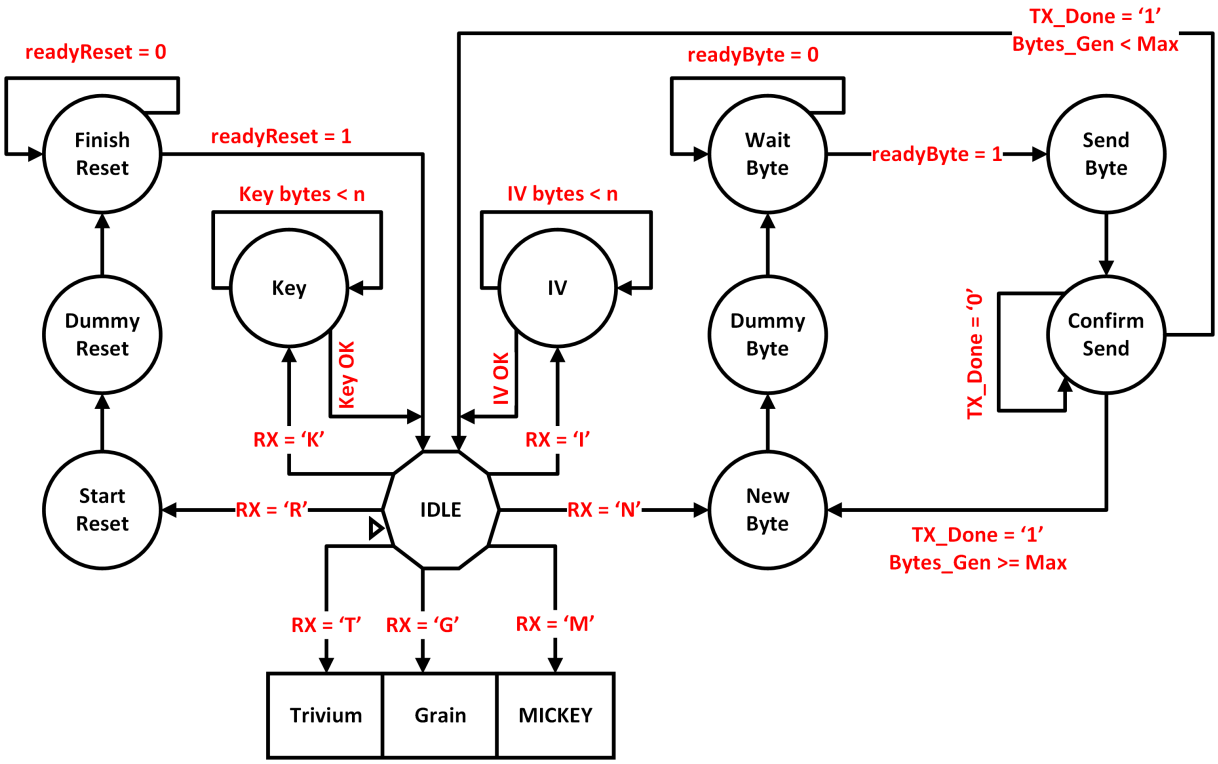
A função do sistema ROSETA é implementar uma interface entre todos os três geradores (Trivium, Grain e MICKEY) e o ambiente externo através de comunicação serial com a plataforma de análise. Para isso ele fornece uma lógica de controle para carregamento de valores da chave *(Key)* e I.V. para os componente de controle do gerador selecionado, além de comandos para *reset* e geração de *keystream*. O esquemático que ilustra sua integração com os demais módulos do sistema é mostrado na Figura 5.9. Internamente, as regras de controle desse sistema são baseadas na máquina de estados *(finite state machine - FSM)* exibida na Figura 5.10.

Figura 5.9 – Esquemático do sistema ROSETA em FPGA.



Fonte: Do autor (2017)

Figura 5.10 – Máquina de estados para controle do *hardware* ROSETA.



Fonte: Do autor (2017)

A definição das funções que devem ser executadas pelo *hardware* são especificadas por meio do envio de caracteres ASCII para o FPGA via interface serial utilizando o *script* implementado em Matlab. Os comandos são:

- **T:** escolhe o sistema Trivium como gerador a ser utilizado.
- **G:** escolhe o sistema Grain como gerador a ser utilizado.
- **M:** escolhe o sistema MICKEY como gerador a ser utilizado.
- **K:** inicia processo de envio da chave de criptografia, após essa solicitação o sistema espera o envio de dez *bytes*, totalizando assim os 80 *bits* da chave para o gerador Trivium.
- **I:** inicia processo de envio do I.V., após essa solicitação o sistema espera o envio de dez *bytes*, totalizando assim os 80 *bits* do I.V. para o gerador Trivium.

- **R:** solicita operação de *reset* do circuito, ou seja, ao enviar esse caractere, o gerador selecionado será reiniciado com o valor da chave (Key) e I.V. enviados previamente.

- **N:** solicita a operação de *"get"* para geração e envio de *bits* pseudoaleatórios, via interface serial. Para cada solicitação são gerados e enviados um número predeterminado de *bits*, por exemplo, nessa implementação foi determinada a geração de 76.800 *bits*, o que equivale a uma imagem de dimensões iguais a 320 x 240 pixels em escala de cinza.

Cada estado é responsável por controlar a execução de uma tarefa, a função de cada um é definida como:

- ***IDLE:*** estado em execução quando o *hardware* não está em operação. Nele, a cada ciclo de *clock* é verificado se foi recebido algum dado via serial. Caso tenha recebido, compara o dado com os caracteres predefinidos para execução de cada operação, caso o casamento ocorra encaminha a FSM para o estado responsável pela execução daquela operação. Se não foi recebido nenhum dado, ou esse é inválido, a máquina de estados continua nesse mesmo estado no próximo ciclo.

- ***Start Reset:*** esse estado é executado quando o *hardware* entra em modo de *reset*, ou seja, quando é enviado o caractere "R". Quando em execução, desabilita transmissões via serial, os sinais de *Key* e I.V. são atualizados para os recebidos anteriormente, além disso, é enviado um sinal de *reset* para o controle do gerador selecionado. Ao final desse processo a FSM é direcionada para o estado *"Dummy Reset"*.

- ***Dummy Reset:*** esse estado é necessário para que o sinal de *reset* do sistema selecionado permaneça em nível alto por mais um ciclo. Isso é necessário pois os sistemas possuem mecanismo de *reset* síncronos, ou seja, a identificação de sinal em nível alto só ocorrerá quando a borda de subida do *clock* for alcançada. Após esse ciclo de *clock* á FSM é direcionada para o estado *"Finish Reset"*.

- ***Finish Reset:*** estado executado quando o *hardware* termina o estado *"Start Reset"*. Nele é enviado o valor lógico 0 para o sinal de *reset* do controle do gerador selecionado, para que esse então inicie o processo de reinicialização de seu circuito. A FSM permanece

nesse estado enquanto a operação de *reset* do gerador não for finalizada, ou seja, enquanto o sinal *"readyReset"* desse circuito não retornar valor 1. Quando esse processo termina a FSM é encaminhada para o estado *"IDLE"*.

- ***New Byte:*** aqui o sinal *"get"* do gerador selecionado é ativado, mostrando assim intenção de solicitação de *byte*. Em seguida a FSM é encaminhada para o estado ‘*'Dummy Byte"*.

- ***Dummy Byte:*** possui propósito igual ao estado *"Dummy Reset"*, nesse caso manter o sinal *"get"* em nível alto por mais um ciclo de *clock*. Ao final de sua execução a FSM é redirecionada para o estado *"Wait Byte"*.

- ***Wait Byte:*** o sinal de *"get"* do gerador, antes colocados em nível alto, é atualizado para nível lógico 0. Ao fazer isso o controlador do gerador inicia o processo de geração do *byte* pseudoaleatório, que será completado assim que o sinal *"readyByte"* correspondente for alterado para nível alto. Quando o *byte* é gerado a FSM é encaminhada para o estado *"Send Byte"*, caso contrário aguarda nesse mesmo estado até a geração estar completa.

- ***Send Byte:*** responsável por ativar o módulo transmissor serial e, além disso, colocar o *byte* gerado no estado anterior *("Wait Byte")* no barramento serial para envio. Ao fazer isso é que o contador de *bytes* gerados é incrementado. Ao final da execução a FSM é encaminhada para o estado *"Confirm Send"*.

- ***Confirm Send:*** o sinal de ativação da transmissão via serial é desativado, caso o módulo transmissor serial retorne com o sinal de envio finalizado (TX_Done) em nível alto, significa que o *byte* foi enviado com sucesso. Caso isso ocorra verifica-se então se o número de *bytes* gerados até o momento é menor que a constante com o valor máximo a ser gerado, caso seja encaminha a FSM para o estado *"New Byte"* para mais um ciclo de solicitação/envio, caso contrário a FSM é encaminhada para o estado *"IDLE"*. Por outro lado, se o a transmissão serial ainda não foi concluída, a FSM permanecerá nesse mesmo estado no próximo ciclo.

- ***Key:*** esse estado é responsável por receber a chave de criptografia *(Key)* e armazená-la em um sinal temporário, para que então, quando o *hardware* for reinicializado esse valor seja

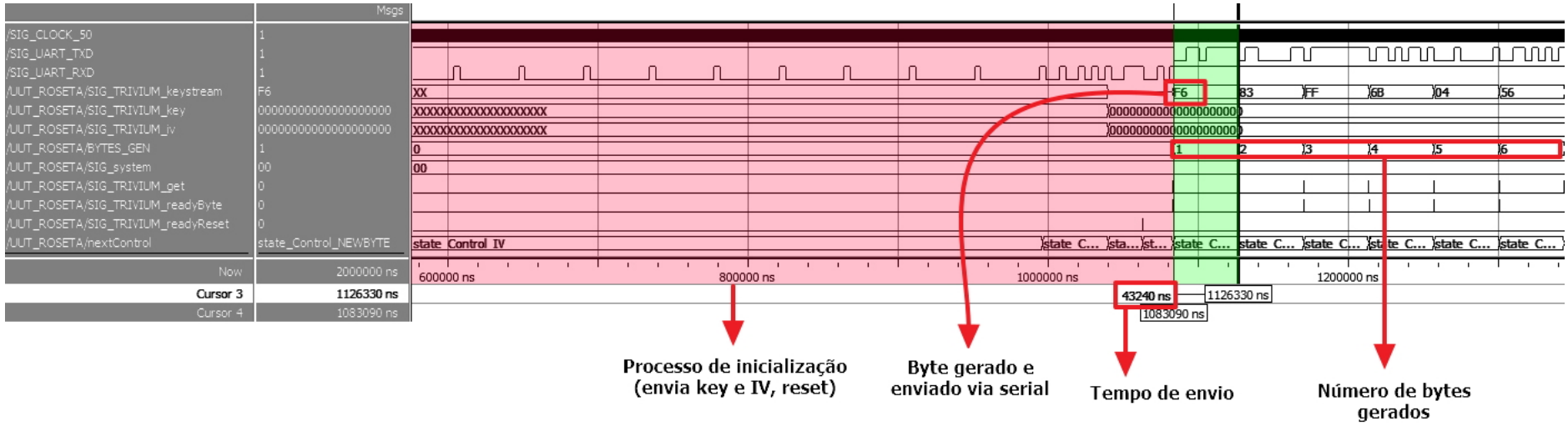
encaminhado para o controlador do gerador selecionado. A chave é enviada em n ciclos, onde $n = length(key)/8$, pois a comunicação serial só transporta 8 *bits* por requisição. Após executar os n ciclos o valor do sinal temporário é encaminhado para o barramento do gerador e o próximo estado da FSM será o estado *"IDLE"*. Caso o contador seja menor que n, a FSM continua nesse mesmo estado esperando o próximo *byte* da chave. O valor recebido é então concatenado no sinal temporário e o contador é incrementado.

- ***I.V:*** estado com operação análoga ao estado *"Key"*, porém responsável por receber o vetor de inicialização (I.V.).

O *hardware* foi testado de duas maneiras diferentes, sendo a primeira por meio de simulação utilizando a ferramenta Altera ModelSim 10.1d. O segundo teste foi realizado sintetizando e carregando o projeto na placa de desenvolvimento Altera DE2, para que em seguida essa fosse conectada ao computador e à plataforma de análise desenvolvida em Matlab via cabo serial. Ambos os testes foram bem sucedidos, na Figura 5.11 é mostrado o comportamento de ondas da simulação do sistema ROSETA. Nela observa-se o tempo de envio *byte*, o resultado da simulação mostrou que o tempo de envio do *byte* é muito superior ao tempo de geração desse pelo seu respectivo gerador.

Como mostrado anteriormente, cada gerador era capaz de produzir um *bit* novo a cada ciclo de *clock*, ou seja, para a geração de 8 *bits* a uma frequência aplicada de 50MHz, o tempo necessário para processamento é igual a 160 ns, porém, como visto agora, o tempo de envio é igual a 43.240 ns. Dessa forma o gargalo dessa implementação encontra-se na interface de comunicação entre o FPGA e o computador, sendo assim necessário novos estudos para encontrar uma interface mais adequada. Vale ressaltar ainda que estes valores foram encontrados utilizando como base para o desenvolvimento a placa de desenvolvimento Altera DE2, ou seja, os resultados aqui apresentados estão sujeitos às limitações deste *hardware* como por exemplo no processo de síntese e roteamento.

Figura 5.11 – Ondas resultantes da simulação do *hardware* ROSETA.



Fonte: Do autor (2017)

O teste com o Matlab ocorreu como esperado, sendo esse capaz de realizar a análise automatizada de uma determinada imagem por meio da requisição e leitura de *bits* pseudoaleatórios gerados no FPGA. A comunicação entre FPGA e Matlab foi configurada com *baud rate* igual à 460.800, sendo assim capaz de transmitir $460.800 * 8 = 3.686.400$ *bits* por segundo, esse valor de *baud* foi escolhido por ser o maior empiricamente encontrado capaz de realizar a comunicação sem erros na transmissão. Após síntese do *hardware*, a utilização de recursos na placa Altera DE2 pode ser observado na Tabela 5.7. Como era esperado o sistema teve uma taxa de utilização de recursos muito baixa, o que comprova o objetivo do Perfil II do projeto eSTREAM: aplicações em *hardware* com restrições de recursos.

Tabela 5.7 – Recursos utilizados após síntese do sistema ROSETA em FPGA.

Entidade	**Elem. Combinacionais**	**Registradores**	**Memória *(bits)***	**Pinos**
ROSETA	199	981	0	3
Grain	251	355	0	0
MICKEY	759	249	0	0
Trivium	361	500	0	0
UART RX	51	27	0	0
UART TX	36	26	0	0
TOTAL	**1657**	**2138**	**0**	**3**

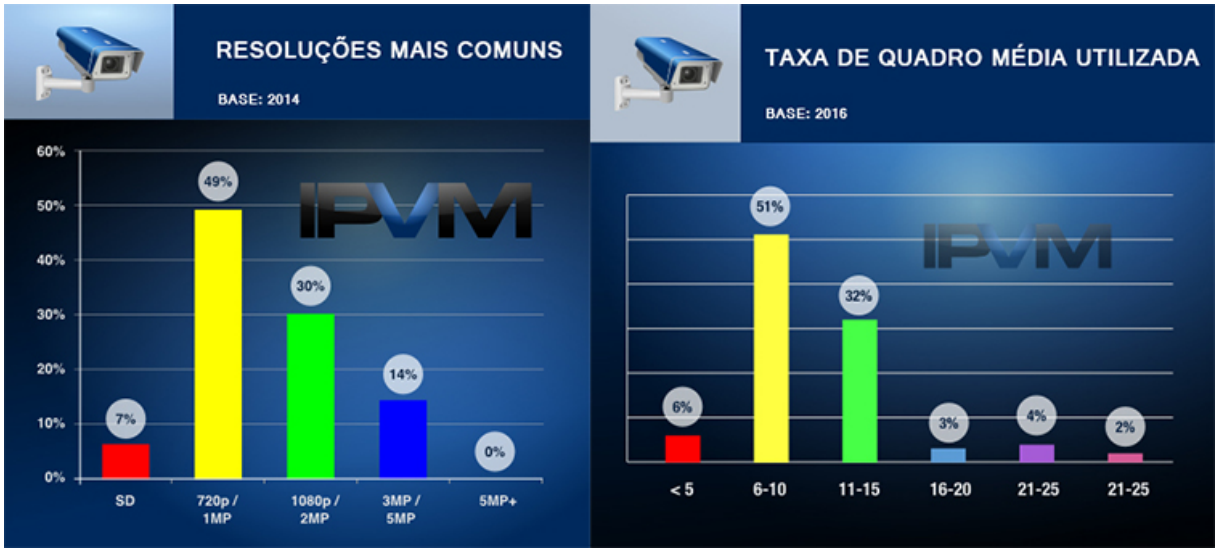
Fonte: Do autor (2017)

O objetivo principal deste trabalho não é a análise da qualidade da implementação em FPGA dos sistemas de Perfil II do projeto eSTREAM, mas sim da qualidade oferecida por eles quando aplicados no processo de criptografia de imagens, que será mostrada na próxima seção. Diversos outros trabalhos já exploraram a qualidade do desenvolvimento em *hardware*, como por exemplo, nos trabalhos de Good e Benaissa (2007) e Kitsos et al. (2013). Porém, como forma de validar as implementações em *software* disponibilizadas pelos autores dos sistemas, decidiu-se realizar mesmo assim a implementação em *hardware*. Essa decisão foi importante pois com ela foi possível desenvolver parte do sistema de análise automatizada e, principalmente garantir que de fato o sistema implementado é viável para um cenário com alta vazão de dados e em tempo real, como é o caso de aplicações de vídeo em tempo real.

Na Tabela 5.2, Tabela 5.4 e Tabela 5.5, foi mostrada a quantidade de imagens (quadros) que cada sistema desenvolvido é capaz de enviar por segundo, considerando duas resoluções de imagem, a *Standard Definition - SD* (640 x 480) e a *High Definition - HD* (1280 x 720).

Para validar a escolha dessas resoluções de imagem e verificar se os valores encontrados de quadros por segundo *(Frame per Second - FPS)* são de fato significativos em um cenário de sistemas de vigilância por câmeras, foram utilizados estatísticas que reportam o valor médio de resolução das câmeras de vigilância vendidas mundialmente no ano de 2014 e a taxa de quadro média encontrada nesse tipo de câmera em 2016. Estes dados são provenientes de pesquisas conduzidas pelo *IPVM Video Surveillance Information* (IPVM, 2017), portal especializado em câmeras de vigilância, que conta com mais de 10.000 membros espalhados por mais de 100 países. Os gráficos representados na Figura 5.12 exibem estas informações.

Figura 5.12 – Câmeras de vigilância - Resoluções e Taxas de quadro.



Fonte: Adaptado de (IPVM, 2017)

Tabela 5.8 – Valor de *throughput* dos sistemas de Perfil II após síntese em FPGA.

Sistema	***bits*/ciclo**	***clock* max**	***throughput* max**	**vídeo HD**	**vídeo SD**
Trivium	1	306 MHz	306 Gbps	14,5 FPS	43,5 FPS
Grain	1	251 MHz	251 Gbps	11,8 FPS	35,6 FPS
MICKEY	1	166 MHz	166 Gbps	7,8 FPS	23,6 FPS

Fonte: Do autor (2017)

A Tabela 5.8 resume os dados de *throughput* encontrados para cada um dos sistemas. Analisando os valores da tabela com as estatísticas da Figura 5.12 referentes aos valores mais comuns de resolução no ano de 2014, observa-se que o modo de operação em resolução SD permite aos sistemas atenderem a uma parcela igual a 7% das câmeras de vigilância consideradas na pesquisa. Por sua vez, quando os sistemas operam em modo HD estes abrangem uma

parcela maior da amostra com valor igual a 49%. Além disso pode-se destacar alguns outros pontos referentes aos valores de taxa de quadros encontrados:

- **Trivium:** comparando a taxa de quadros obtida, este sistema operando em modo SD atende a 100% das câmeras analisadas em 2016. Já no modo de operação HD o sistema é capaz de operar com valores de FPS compatíveis com 89% da amostra da pesquisa. Além disso, é o sistema que proporciona maior vazão de *bits* entre os três analisados.

- **Grain:** comparando a taxa de quadros obtida, este sistema operando em modo SD atende a 100% das câmeras analisadas em 2016. Já no modo de operação HD o sistema é capaz de operar com valores de FPS compatíveis com 59% da amostra da pesquisa. Por outro lado, considerando a vazão de *bits* como parâmetro de comparação o sistema Grain fica em 2º lugar entre os três sistemas.

- **MICKEY:** analisando a taxa de quadros obtida, este sistema operando em modo SD atende a 100% das câmeras analisadas em 2016. Já no modo de operação HD a mesma taxa equivale a aproximadamente 66% do valor encontrado na mesma situação utilizando o sistema Grain, este valor encontra-se próximo do limite inferior da faixa que engloba as câmeras que operam com FPS variando de 6 a 10. É o sistema que possui menor taxa de vazão entre os três analisados.

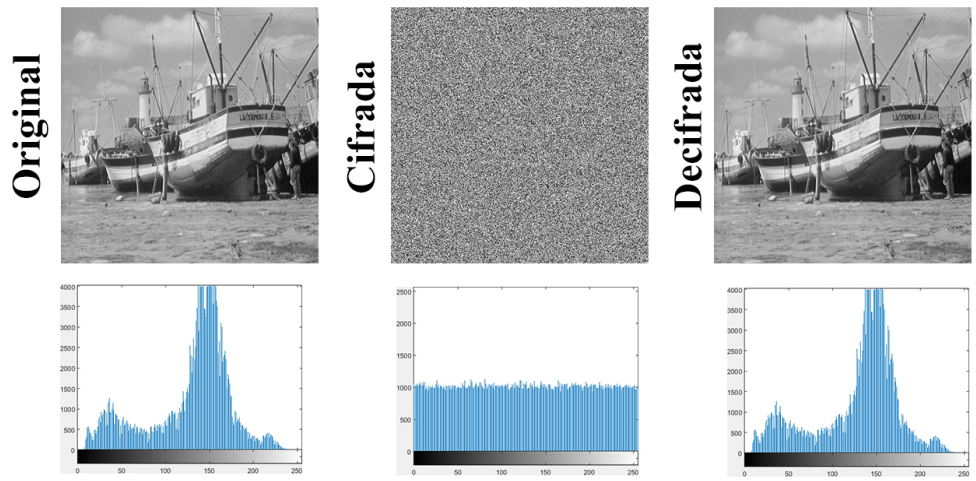
5.3 Qualidade dos Sistema de Criptografia

Para reprodutibilidade dos experimentos, foram utilizados 50 geradores aleatórios no Matlab, assim, cada uma das 44 imagens foi executada 50 vezes, cada i-ésima execução utilizando o i-ésimo gerador, sem repetir nenhum. Esse processo foi realizado para cada um dos três algoritmos analisados, totalizando assim $44 * 50 * 3 = 6.600$ execuções.

5.3.1 Análise Visual

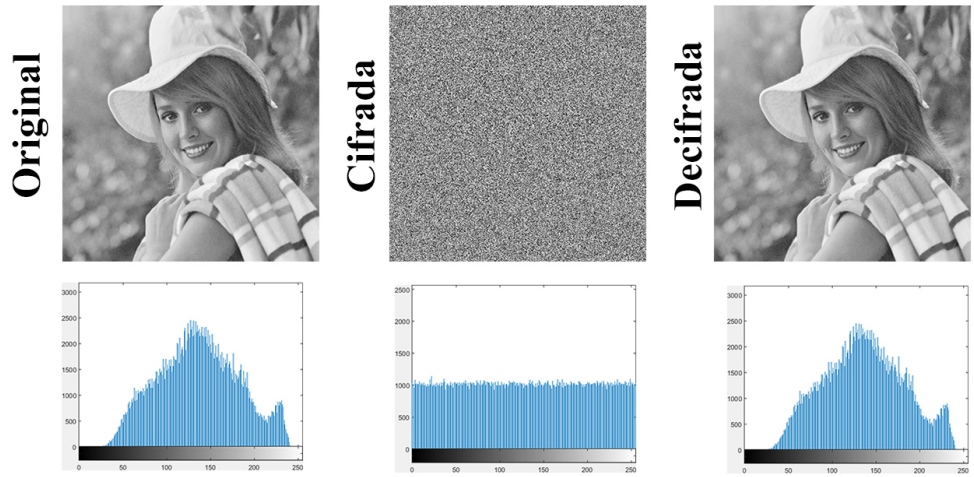
Na Figura 5.13 e na Figura 5.14 é mostrada a eficiência dos algoritmos quanto à analise visual quando aplicados sobre imagens em tons de cinza, enquanto que na Figura 5.15 essa análise é mostrada para imagens coloridas (RGB). Em ambas as situações os sistemas se comportaram como esperado, mostrando assim que a informação original foi, de fato, ocultada após o processo de criptografia e revelada após a decifragem utilizando a mesma chave de criptografia e vetor de inicialização.

Figura 5.13 – Imagem “boat.512” - sistema Trivium.



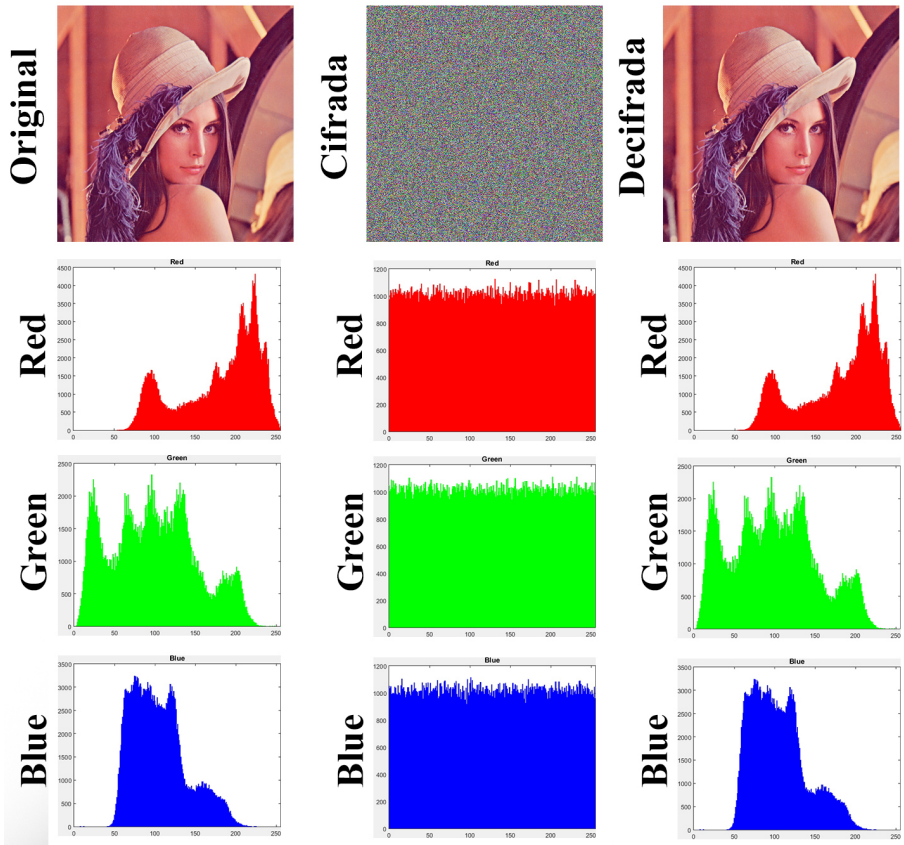
Fonte: Do autor (2017)

Figura 5.14 – Imagem “elaine.512” - sistema MICKEY.



Fonte: Do autor (2017)

Figura 5.15 – Imagem “4.2.04” - sistema Grain.



Fonte: Do autor (2017)

5.3.2 Análise de Histograma

As Figuras 5.15, 5.13 e 5.14 exibem, graficamente, os histogramas das imagens “4.2.04”, “boat.512” e “elaine.512”, respectivamente, na sua forma original e também após cifradas e decifradas. Além da análise gráfica, calcula-se também o valor médio de intensidade dos *pixels* de cada imagem a fim de comprovar o comportamento uniforme dessa. Assim, é esperado em uma imagem com valores de *pixel* variando de $0 \ldots 255$, um valor médio o mais próximo possível de $\frac{a+b}{2} = \frac{0+255}{2} = \frac{255}{2} = 127,5$. A Tabela 5.9 exibe a intensidade média e desvio padrão para todas as imagens do *dataset* em tons de cinza, originais e cifradas, enquanto a Tabela 5.10 e a Tabela 5.11 exibem essas mesmas informações para todas as imagens do *dataset* coloridas (RGB). Ambas as tabelas comprovam a uniformidade das imagens cifradas pelos três sistemas. Em destaque nas tabelas estão os valores encontrados para as imagens exibidas no processo de análise visual na Subseção 5.3.1.

Tabela 5.9 – Intensidade dos *pixels* nas imagem originais e cifradas - *grayscale*.

Arquivo	Original	Trivium	Grain	MICKEY
5.1.09	127,760025	127,527996 ± 0,2525	127,484123 ± 0,3206	127,500289 ± 0,2833
5.1.10	140,507065	127,441748 ± 0,2921	127,512474 ± 0,2947	127,520769 ± 0,2995
5.1.11	193,553833	127,544098 ± 0,2856	127,554648 ± 0,2354	127,507110 ± 0,2670
5.1.12	185,980270	127,517059 ± 0,2587	127,545041 ± 0,2794	127,550716 ± 0,2267
5.1.13	225,914978	127,485142 ± 0,3103	127,574224 ± 0,2440	127,533406 ± 0,2964
5.1.14	104,469925	127,563426 ± 0,3056	127,487378 ± 0,2924	127,503052 ± 0,3271
5.2.08	123,177197	127,480063 ± 0,1445	127,510601 ± 0,1464	127,499035 ± 0,1332
5.2.09	180,571754	127,439453 ± 0,1338	127,498343 ± 0,1652	127,506569 ± 0,1243
5.2.10	113,801506	127,493777 ± 0,1375	127,484973 ± 0,1561	127,514039 ± 0,1308
5.3.01	89,007527	127,508452 ± 0,0670	127,499943 ± 0,0769	127,498161 ± 0,0746
5.3.02	82,994536	127,514930 ± 0,0775	127,511907 ± 0,0628	127,511489 ± 0,0736
7.1.01	107,114017	127,541193 ± 0,1351	127,495372 ± 0,1297	127,472611 ± 0,1486
7.1.02	175,334938	127,454014 ± 0,1480	127,498130 ± 0,1226	127,522193 ± 0,1343
7.1.03	132,384727	127,478093 ± 0,1398	127,505950 ± 0,1311	127,449757 ± 0,1500
7.1.04	116,118458	127,549649 ± 0,1374	127,508154 ± 0,1437	127,505006 ± 0,1168
7.1.05	106,393360	127,536252 ± 0,1245	127,483934 ± 0,1358	127,472816 ± 0,1451
7.1.06	90,482506	127,538657 ± 0,1194	127,506095 ± 0,1220	127,481181 ± 0,1339
7.1.07	108,238628	127,506636 ± 0,1200	127,483910 ± 0,1338	127,486891 ± 0,1261
7.1.08	127,188091	127,502891 ± 0,1538	127,528178 ± 0,1604	127,542047 ± 0,1217
7.1.09	125,603642	127,534205 ± 0,1263	127,492313 ± 0,1649	127,522853 ± 0,1621
7.1.10	119,272945	127,509718 ± 0,1195	127,522173 ± 0,1299	127,523926 ± 0,1492
7.2.01	32,513995	127,515405 ± 0,0768	127,502996 ± 0,0871	127,502423 ± 0,0660
boat.512	129,707966	127,452536 ± 0,1469	127,509175 ± 0,1220	127,525035 ± 0,1304
elaine.512	136,356514	127,534981 ± 0,1456	127,488781 ± 0,1640	127,498975 ± 0,1439
gray21.512	127,038322	127,471030 ± 0,1492	127,513045 ± 0,1184	127,486098 ± 0,1547
numbers.512	103,519405	127,523202 ± 0,1596	127,482759 ± 0,1556	127,486180 ± 0,1231
ruler.512	226,940117	127,489680 ± 0,1378	127,513721 ± 0,1600	127,494625 ± 0,1463
testpat.1k	124,616351	127,502161 ± 0,0859	127,499622 ± 0,0625	127,508269 ± 0,0870

Fonte: Do autor (2017)

Tabela 5.10 – Intensidade dos *pixels* nas imagem originais - RGB.

Arquivo	Red	Green	Blue
4.1.01	75,826752	52,559174	46,304993
4.1.02	42,074921	30,086411	27,539536
4.1.03	137,602783	139,957764	144,017792
4.1.04	129,218079	99,267456	125,199265
4.1.05	146,564026	132,999695	142,022537
4.1.06	132,202057	124,901642	143,263229
4.1.07	179,204071	180,649933	142,347534
4.1.08	174,897430	170,865936	128,346130

Arquivo	Red	Green	Blue
4.2.01	176,270004	70,494366	79,928276
4.2.02	234,194530	208,644001	163,551582
4.2.03	137,391335	128,858776	113,117107
4.2.04	180,223660	99,051216	105,410252
4.2.05	177,576923	177,852436	190,213970
4.2.06	131,006691	124,304455	114,892719
4.2.07	149,821335	115,568256	66,533596
house	155,435745	168,225960	142,209373

Fonte: Do autor (2017)

Tabela 5.11 – Intensidade dos *pixels* nas imagem cifradas - RGB.

	Trivium			Grain			MICKEY		
Arquivo	**Red**	**Green**	**Blue**	**Red**	**Green**	**Blue**	**Red**	**Green**	**Blue**
4.1.01	127,521636 ± 0,2677	127,561753 ± 0,3021	127,529765 ± 0,2933	127,521195 ± 0,3009	127,469154 ± 0,2735	127,529150 ± 0,3138	127,516372 ± 0,2466	127,504313 ± 0,2244	127,513237 ± 0,2637
4.1.02	127,508528 ± 0,2344	127,570996 ± 0,2917	127,558523 ± 0,2857	127,490977 ± 0,2571	127,542251 ± 0,2551	127,492128 ± 0,3114	127,479119 ± 0,2586	127,505716 ± 0,2965	127,494509 ± 0,3186
4.1.03	127,539200 ± 0,2486	127,506234 ± 0,3091	127,427942 ± 0,2857	127,514123 ± 0,2774	127,392363 ± 0,3005	127,529581 ± 0,3293	127,517062 ± 0,2472	127,513598 ± 0,2873	127,521242 ± 0,2621
4.1.04	127,505773 ± 0,2606	127,592027 ± 0,2560	127,518044 ± 0,2965	127,518820 ± 0,3284	127,529403 ± 0,3070	127,525427 ± 0,2933	127,506949 ± 0,2545	127,537065 ± 0,2962	127,421623 ± 0,3215
4.1.05	127,555608 ± 0,2836	127,420214 ± 0,2993	127,505060 ± 0,2825	127,494496 ± 0,2456	127,466472 ± 0,2930	127,564506 ± 0,3185	127,595583 ± 0,2823	127,519508 ± 0,2750	127,526602 ± 0,2684
4.1.06	127,496622 ± 0,2393	127,573097 ± 0,2901	127,445532 ± 0,2672	127,564506 ± 0,3144	127,494703 ± 0,3041	127,491137 ± 0,2668	127,474357 ± 0,2971	127,542092 ± 0,2913	127,487901 ± 0,2456
4.1.07	127,499918 ± 0,2813	127,421074 ± 0,2499	127,476418 ± 0,2673	127,615375 ± 0,2797	127,428922 ± 0,2768	127,528088 ± 0,3271	127,518751 ± 0,2461	127,415252 ± 0,3086	127,478479 ± 0,2890
4.1.08	127,546267 ± 0,2628	127,427188 ± 0,3033	127,498379 ± 0,2713	127,584009 ± 0,3124	127,433999 ± 0,2667	127,582190 ± 0,2671	127,495917 ± 0,2945	127,465472 ± 0,3321	127,471376 ± 0,2824
4.2.01	127,491768 ± 0,1268	127,471782 ± 0,1526	127,521149 ± 0,1648	127,471609 ± 0,1544	127,516212 ± 0,1263	127,484685 ± 0,1431	127,479397 ± 0,1428	127,518244 ± 0,1359	127,538376 ± 0,1369
4.2.02	127,465245 ± 0,1372	127,496067 ± 0,1523	127,477066 ± 0,1172	127,500116 ± 0,1444	127,502489 ± 0,1537	127,485383 ± 0,1433	127,511187 ± 0,1393	127,498003 ± 0,1262	127,478222 ± 0,1251
4.2.03	127,548495 ± 0,1481	127,520798 ± 0,1374	127,485255 ± 0,1290	127,508843 ± 0,1341	127,503225 ± 0,1565	127,484812 ± 0,1443	127,490810 ± 0,1406	127,549031 ± 0,1493	127,493784 ± 0,1715
4.2.04	127,489926 ± 0,1379	127,488449 ± 0,1551	127,518575 ± 0,1544	127,470704 ± 0,1194	127,525226 ± 0,1500	127,502625 ± 0,1174	127,500020 ± 0,1462	127,503870 ± 0,1376	127,519766 ± 0,1567
4.2.05	127,451920 ± 0,1380	127,522794 ± 0,1736	127,479739 ± 0,1294	127,478402 ± 0,1514	127,514780 ± 0,1273	127,498622 ± 0,1544	127,499514 ± 0,1370	127,512162 ± 0,1514	127,488302 ± 0,1488
4.2.06	127,516124 ± 0,1401	127,489392 ± 0,1669	127,464392 ± 0,1570	127,511761 ± 0,1492	127,513096 ± 0,1560	127,477699 ± 0,1430	127,494748 ± 0,1163	127,517370 ± 0,1293	127,520118 ± 0,1389
4.2.07	127,458826 ± 0,1183	127,512458 ± 0,1515	127,547018 ± 0,1584	127,507897 ± 0,1334	127,512587 ± 0,1321	127,492875 ± 0,1530	127,471256 ± 0,1482	127,489200 ± 0,1547	127,499842 ± 0,1533
house	127,501067 ± 0,1320	127,513427 ± 0,1445	127,522466 ± 0,1392	127,515183 ± 0,1447	127,523518 ± 0,1263	127,494096 ± 0,1230	127,493349 ± 0,1513	127,487968 ± 0,1451	127,461433 ± 0,1436

Fonte: Do autor (2017)

5.3.3 Entropia

A Tabela 5.12 exibe o valor de entropia média de todos os canais RGB juntos para as imagens originais e cifradas do *dataset*. Já a Tabela 5.13 e a Tabela 5.14, mostram os resultados das mesmas análises, porém calculando o valor de entropia para cada canal de cor individualmente. Por fim, a Tabela 5.15 mostra o valor de entropia para as imagens em tons de cinza do *dataset*. Os valores encontrados nas imagens cifradas são extremamente próximos do valor ótimo $E = 8$, mostrando assim que as imagens analisadas são de fato parecidas com uma informação aleatória. Em destaque nas tabelas estão os valores encontrados para as imagens exibidas no processo de análise visual na Subseção 5.3.1.

Tabela 5.12 – Entropia de todos os canais das imagens originais e cifradas - RGB.

Arquivo	Original	Trivium	Grain	MICKEY
4.1.01	6,898139	7,999062 ± 0,0001	7,999054 ± 0,0001	7,999068 ± 0,0001
4.1.02	6,294498	7,999072 ± 0,0001	7,999075 ± 0,0001	7,999062 ± 0,0001
4.1.03	5,970916	7,999057 ± 0,0001	7,999078 ± 0,0001	7,999058 ± 0,0001
4.1.04	7,426957	7,999053 ± 0,0001	7,999063 ± 0,0001	7,999060 ± 0,0001
4.1.05	7,068625	7,999069 ± 0,0001	7,999058 ± 0,0001	7,999078 ± 0,0001
4.1.06	7,537089	7,999052 ± 0,0001	7,999059 ± 0,0001	7,999063 ± 0,0001
4.1.07	6,583485	7,999045 ± 0,0001	7,999066 ± 0,0001	7,999074 ± 0,0001
4.1.08	6,852716	7,999054 ± 0,0001	7,999077 ± 0,0001	7,999060 ± 0,0001

Arquivo	Original	Trivium	Grain	MICKEY
4.2.01	7,242831	7,999767 ± 0,0000	7,999769 ± 0,0000	7,999762 ± 0,0000
4.2.02	6,416494	7,999768 ± 0,0000	7,999771 ± 0,0000	7,999763 ± 0,0000
4.2.03	7,762436	7,999769 ± 0,0000	7,999768 ± 0,0000	7,999767 ± 0,0000
4.2.04	7,750197	7,999769 ± 0,0000	7,999769 ± 0,0000	7,999765 ± 0,0000
4.2.05	6,663908	7,999763 ± 0,0000	7,999766 ± 0,0000	7,999768 ± 0,0000
4.2.06	7,762170	7,999769 ± 0,0000	7,999767 ± 0,0000	7,999767 ± 0,0000
4.2.07	7,669826	7,999765 ± 0,0000	7,999763 ± 0,0000	7,999766 ± 0,0000
house	7.485787	7.999768 ± 0,0000	7,999764 ± 0,0000	7,999766 ± 0,0000

Fonte: Do autor (2017)

Tabela 5.13 – Entropia de cada canal RGB das imagens originais,

Arquivo	Red	Green	Blue
4.1.01	6,420046	6,445678	6,380712
4.1.02	6,249884	5,964151	5,930919
4.1.03	5,715009	5,373845	5,711657
4.1.04	7,254873	7,270383	6,782501
4.1.05	6,431052	6,538931	6,232038
4.1.06	7,210437	7,413611	6,920742
4.1.07	5,262618	5,694728	6,546413
4.1.08	5,791993	6,219509	6,798641

Arquivo	Red	Green	Blue
4.2.01	6,948061	6,884455	6,126452
4.2.02	4,337192	6,664328	6,428813
4.2.03	7,706672	7,474432	7,752217
4.2.04	7,253102	7,594038	6,968427
4.2.05	6,717765	6,798979	6,213774
4.2.06	7,312388	7,642854	7,213642
4.2.07	7,338827	7,496253	7,058306
house	7,415627	7,229479	7,435384

Fonte: Do autor (2017)

Tabela 5.14 – Entropia de cada canal RGB das imagens cifradas.

	Trivium			Grain			MICKEY		
Arquivo	**Red**	**Green**	**Blue**	**Red**	**Green**	**Blue**	**Red**	**Green**	**Blue**
4.1.01	7,997221 ± 0,0002	7,997219 ± 0,0003	7,997231 ± 0,0003	7,997215 ± 0,0002	7,997162 ± 0,0002	7,997165 ± 0,0002	7,997211 ± 0,0003	7,997236 ± 0,0002	7,997216 ± 0,0003
4.1.02	7,997242 ± 0,0002	7,997220 ± 0,0003	7,997187 ± 0,0002	7,997187 ± 0,0002	7,997176 ± 0,0002	7,997188 ± 0,0002	7,997205 ± 0,0002	7,997205 ± 0,0003	7,997157 ± 0,0002
4.1.03	7,997215 ± 0,0002	7,997191 ± 0,0003	7,997202 ± 0,0003	7,997244 ± 0,0002	7,997206 ± 0,0003	7,997205 ± 0,0003	7,997201 ± 0,0002	7,997209 ± 0,0002	7,997145 ± 0,0003
4.1.04	7,997192 ± 0,0002	7,997171 ± 0,0002	7,997142 ± 0,0003	7,997188 ± 0,0003	7,997181 ± 0,0002	7,997242 ± 0,0003	7,997197 ± 0,0003	7,997182 ± 0,0003	7,997195 ± 0,0002
4.1.05	7,997163 ± 0,0003	7,997190 ± 0,0002	7,997095 ± 0,0003	7,997154 ± 0,0003	7,997202 ± 0,0002	7,997140 ± 0,0003	7,997257 ± 0,0003	7,997205 ± 0,0002	7,997203 ± 0,0003
4.1.06	7,997170 ± 0,0002	7,997229 ± 0,0003	7,997177 ± 0,0002	7,997176 ± 0,0003	7,997201 ± 0,0002	7,997212 ± 0,0002	7,997220 ± 0,0002	7,997194 ± 0,0002	7,997200 ± 0,0002
4.1.07	7,997180 ± 0,0002	7,997112 ± 0,0003	7,997172 ± 0,0003	7,997197 ± 0,0002	7,997179 ± 0,0002	7,997223 ± 0,0003	7,997215 ± 0,0003	7,997174 ± 0,0003	7,997207 ± 0,0003
4.1.08	7,997177 ± 0,0002	7,997233 ± 0,0002	7,997111 ± 0,0002	7,997222 ± 0,0002	7,997170 ± 0,0002	7,997148 ± 0,0002	7,997221 ± 0,0003	7,997234 ± 0,0002	7,997233 ± 0,0003
4.2.01	7,999288 ± 0,0001	7,999297 ± 0,0001	7,999304 ± 0,0001	7,999307 ± 0,0001	7,999297 ± 0,0001	7,999308 ± 0,0001	7,999308 ± 0,0001	7,999309 ± 0,0001	7,999289 ± 0,0001
4.2.02	7,999296 ± 0,0001	7,999296 ± 0,0001	7,999297 ± 0,0001	7,999305 ± 0,0001	7,999305 ± 0,0001	7,999290 ± 0,0001	7,999301 ± 0,0001	7,999288 ± 0,0001	7,999293 ± 0,0001
4.2.03	7,999297 ± 0,0001	7,999295 ± 0,0001	7,999307 ± 0,0001	7,999304 ± 0,0001	7,999301 ± 0,0001	7,999307 ± 0,0001	7,999291 ± 0,0001	7,999289 ± 0,0001	7,999285 ± 0,0001
4.2.04	7,999291 ± 0,0001	7,999294 ± 0,0001	7,999304 ± 0,0001	7,999295 ± 0,0001	7,999287 ± 0,0001	7,999313 ± 0,0001	7,999291 ± 0,0001	7,999295 ± 0,0001	7,999304 ± 0,0001
4.2.05	7,999297 ± 0,0001	7,999287 ± 0,0001	7,999298 ± 0,0001	7,999299 ± 0,0001	7,999291 ± 0,0001	7,999290 ± 0,0001	7,999303 ± 0,0001	7,999300 ± 0,0001	7,999306 ± 0,0001
4.2.06	7,999300 ± 0,0001	7,999284 ± 0,0000	7,999315 ± 0,0001	7,999310 ± 0,0001	7,999289 ± 0,0001	7,999297 ± 0,0001	7,999274 ± 0,0001	7,999300 ± 0,0000	7,999291 ± 0,0001
4.2.07	7,999319 ± 0,0001	7,999305 ± 0,0001	7,999306 ± 0,0001	7,999301 ± 0,0001	7,999290 ± 0,0001	7,999283 ± 0,0001	7,999299 ± 0,0001	7,999302 ± 0,0001	7,999305 ± 0,0000
house	7,999313 ± 0,0001	7,999291 ± 0,0001	7,999294 ± 0,0001	7,999305 ± 0,0000	7,999299 ± 0,0001	7,999293 ± 0,0001	7,999292 ± 0,0001	7,999305 ± 0,0001	7,999294 ± 0,0001

Fonte: Do autor (2017)

Tabela 5.15 – Entropia das imagens originais e cifradas - *grayscale*.

Arquivo	Original	Trivium	Grain	MICKEY
5.1.09	6,709312	7,997233 ± 0,0002	7,997235 ± 0,0003	7,997155 ± 0,0002
5.1.10	7,311807	7,997264 ± 0,0002	7,997161 ± 0,0002	7,997227 ± 0,0003
5.1.11	6,452275	7,997170 ± 0,0002	7,997191 ± 0,0002	7,997158 ± 0,0002
5.1.12	6,705667	7,997201 ± 0,0002	7,997190 ± 0,0002	7,997175 ± 0,0003
5.1.13	1,548314	7,997200 ± 0,0003	7,997190 ± 0,0003	7,997131 ± 0,0003
5.1.14	7,342433	7,997201 ± 0,0003	7,997209 ± 0,0002	7,997173 ± 0,0002
5.2.08	7,201008	7,999302 ± 0,0001	7,999298 ± 0,0001	7,999306 ± 0,0001
5.2.09	6,993994	7,999306 ± 0,0001	7,999301 ± 0,0001	7,999300 ± 0,0001
5.2.10	5,70556	7,999292 ± 0,0001	7,999305 ± 0,0001	7,999300 ± 0,0001
5.3.01	7,523737	7,999821 ± 0,0000	7,999826 ± 0,0000	7,999823 ± 0,0000
5.3.02	6,83033	7,999823 ± 0,0000	7,999825 ± 0,0000	7,999830 ± 0,0000
7.1.01	6,027415	7,999310 ± 0,0001	7,999316 ± 0,0001	7,999278 ± 0,0001
7.1.02	4,004499	7,999293 ± 0,0000	7,999302 ± 0,0001	7,999307 ± 0,0001
7.1.03	5,49574	7,999310 ± 0,0001	7,999309 ± 0,0001	7,999293 ± 0,0001

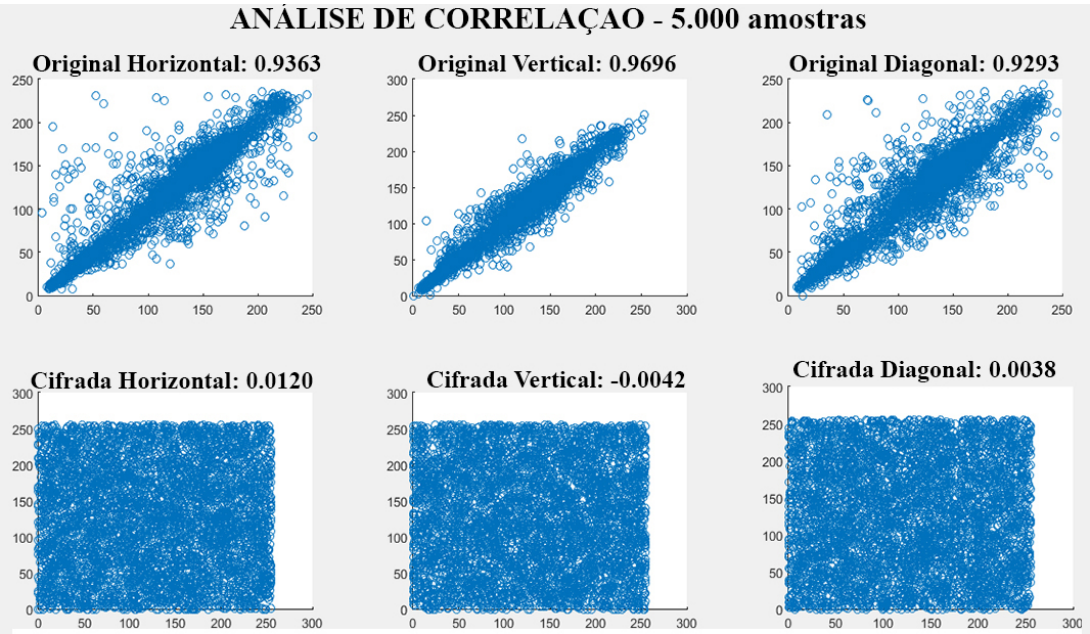
Arquivo	Original	Trivium	Grain	MICKEY
7.1.04	6,107418	7,999300 ± 0,0001	7,999289 ± 0,0001	7,999284 ± 0,0001
7.1.05	6,563196	7,999298 ± 0,0001	7,999287 ± 0,0001	7,999308 ± 0,0001
7.1.06	6,695283	7,999293 ± 0,0001	7,999303 ± 0,0001	7,999296 ± 0,0001
7.1.07	5,991599	7,999309 ± 0,0001	7,999299 ± 0,0001	7,999279 ± 0,0001
7.1.08	5,053448	7,999288 ± 0,0001	7,999316 ± 0,0001	7,999289 ± 0,0001
7.1.09	6,189814	7,999295 ± 0,0001	7,999301 ± 0,0001	7,999322 ± 0,0001
7.1.10	5,90879	7,999288 ± 0,0001	7,999313 ± 0,0001	7,999302 ± 0,0001
7.2.01	5,641454	7,999823 ± 0,0000	7,999821 ± 0,0000	7,999821 ± 0,0000
boat.512	7,19137	7,999305 ± 0,0001	7,999311 ± 0,0001	7,999308 ± 0,0001
elaine.512	7,505984	7,999313 ± 0,0001	7,999300 ± 0,0001	7,999293 ± 0,0001
gray21.512	4,392295	7,999306 ± 0,0001	7,999292 ± 0,0001	7,999304 ± 0,0001
numbers.512	7,729247	7,999315 ± 0,0001	7,999291 ± 0,0001	7,999308 ± 0,0001
ruler.512	0,500033	7,999309 ± 0,0001	7,999291 ± 0,0001	7,999293 ± 0,0001
testpat.1k	4,407726	7,999827 ± 0,0000	7,999825 ± 0,0000	7,999827 ± 0,0000

Fonte: Do autor (2017)

5.3.4 Correlação

Uma baixa correlação entre pixels adjacentes indica um sistema de criptografia robusto. Nesse trabalho, foram analisadas as correlações para *pixels* adjacentes horizontalmente, verticalmente e diagonalmente, para um total de 5000 *pixels*. A Tabela 5.16 exibe o valor de correlação calculado para às imagens coloridas originais, já a Tabela 5.17 exibe o valor para as mesmas imagens após o processo de cifragem. A Tabela 5.19 e a Tabela 5.18 exibem os mesmos dados porém, dessa vez, para as imagens em tons de cinza. A análise desses resultados mostram que ambos os algoritmos possuem níveis de correlação próximos de 0 após processo de cifragem. Por fim, a Figura 5.16 exibe de forma numérica e gráfica a correlação antes e depois da cifragem da imagem "boat.512". Em destaque nas tabelas estão os valores encontrados para as imagens exibidas no processo de análise visual na Subseção 5.3.1.

Figura 5.16 – Análise de correlação da imagem "boat.512".



Fonte: Do autor (2017)

Tabela 5.16 – Correlação de todos os canais das imagens originais - RGB.

Arquivo	Horz	Vert	Diag
4.1.01	0,967732	0,960199	0,945419
4.1.02	0,933158	0,951909	0,902601
4.1.03	0,975691	0,916755	0,902670
4.1.04	0,965104	0,980497	0,949444
4.1.05	0,976626	0,952147	0,936292
4.1.06	0,962991	0,941012	0,924544
4.1.07	0,979414	0,981844	0,965058
4.1.08	0,974270	0,975334	0,951271

Arquivo	Horz	Vert	Diag
4.2.01	0,985846	0,987035	0,974450
4.2.02	0,923456	0,939798	0,880463
4.2.03	0,898946	0,836395	0,809995
4.2.04	0,961044	0,976239	0,946753
4.2.05	0,964751	0,953425	0,926950
4.2.06	0,966231	0,963327	0,949070
4.2.07	0,970779	0,971657	0,957565
house	0,955053	0,956727	0,918094

Fonte: Do autor (2017)

Tabela 5.17 – Correlação de todos os canais das imagens cifradas - RGB.

	Trivium			Grain			MICKEY		
Arquivo	**Horz**	**Vert**	**Diag**	**Horz**	**Vert**	**Diag**	**Horz**	**Vert**	**Diag**
4.1.01	-0,000083 ± 0,0079	-0,000027 ± 0,0096	-0,000176 ± 0,0091	-0,000926 ± 0,0080	0,001169 ± 0,0082	-0,000838 ± 0,0075	0,000676 ± 0,0079	0,000062 ± 0,0077	-0,000533 ± 0,0082
4.1.02	-0,000306 ± 0,0071	-0,000040 ± 0,0084	-0,000339 ± 0,0080	-0,000734 ± 0,0075	0,001016 ± 0,0074	-0,000782 ± 0,0079	-0,000182 ± 0,0070	0,000942 ± 0,0080	0,000085 ± 0,0078
4.1.03	-0,001031 ± 0,0094	-0,000418 ± 0,0092	0,000418 ± 0,0088	0,000208 ± 0,0082	0,002067 ± 0,0079	-0,000948 ± 0,0086	0,001078 ± 0,0085	0,002020 ± 0,0093	-0,000864 ± 0,0091
4.1.04	-0,000637 ± 0,0076	0,001166 ± 0,0086	0,001630 ± 0,0085	0,000921 ± 0,0090	0,001726 ± 0,0100	0,000223 ± 0,0080	0,000688 ± 0,0094	0,001822 ± 0,0095	-0,000551 ± 0,0099
4.1.05	0,000562 ± 0,0080	-0,001346 ± 0,0082	-0,000539 ± 0,0090	-0,000978 ± 0,0081	0,001326 ± 0,0083	-0,000973 ± 0,0093	-0,001221 ± 0,0084	-0,000423 ± 0,0085	0,000668 ± 0,0089
4.1.06	-0,000112 ± 0,0081	0,000524 ± 0,0098	-0,001551 ± 0,0075	-0,000905 ± 0,0078	0,002737 ± 0,0083	-0,000967 ± 0,0081	-0,001325 ± 0,0077	0,000052 ± 0,0069	-0,000067 ± 0,0094
4.1.07	-0,000079 ± 0,0086	0,000826 ± 0,0084	0,000287 ± 0,0085	-0,000681 ± 0,0080	0,001716 ± 0,0091	-0,000640 ± 0,0080	0,001198 ± 0,0081	0,002065 ± 0,0078	0,001155 ± 0,0082
4.1.08	-0,001417 ± 0,0086	0,000508 ± 0,0090	0,000836 ± 0,0085	-0,000305 ± 0,0082	0,002300 ± 0,0096	-0,001121 ± 0,0085	0,000141 ± 0,0086	0,001436 ± 0,0082	0,001072 ± 0,0105
4.2.01	0,001050 ± 0,0083	0,001225 ± 0,0090	0,000619 ± 0,0093	-0,001514 ± 0,0065	0,001973 ± 0,0081	-0,002315 ± 0,0092	0,000694 ± 0,0090	-0,000453 ± 0,0073	-0,000190 ± 0,0078
4.2.02	0,001310 ± 0,0081	0,001112 ± 0,0091	-0,000597 ± 0,0085	-0,001640 ± 0,0078	-0,000110 ± 0,0082	0,000226 ± 0,0068	0,001616 ± 0,0082	-0,001112 ± 0,0082	0,001738 ± 0,0089
4.2.03	0,000942 ± 0,0084	0,001316 ± 0,0092	-0,000476 ± 0,0091	-0,000016 ± 0,0071	-0,001124 ± 0,0082	-0,000507 ± 0,0085	0,002206 ± 0,0079	-0,000263 ± 0,0079	0,001149 ± 0,0077
4.2.04	0,002112 ± 0,0090	-0,000955 ± 0,0071	0,001395 ± 0,0092	-0,001726 ± 0,0077	0,000492 ± 0,0069	-0,000633 ± 0,0078	0,000217 ± 0,0089	-0,000785 ± 0,0079	0,000705 ± 0,0098
4.2.05	0,000814 ± 0,0087	0,001079 ± 0,0081	0,000440 ± 0,0089	0,000273 ± 0,0076	-0,000178 ± 0,0078	-0,001223 ± 0,0092	0,000810 ± 0,0081	-0,000338 ± 0,0082	0,001814 ± 0,0087
4.2.06	0,000530 ± 0,0077	-0,000909 ± 0,0079	0,000324 ± 0,0092	-0,001419 ± 0,0073	0,000932 ± 0,0081	-0,001648 ± 0,0095	0,000832 ± 0,0073	-0,001337 ± 0,0072	-0,000923 ± 0,0075
4.2.07	0,000117 ± 0,0084	0,000084 ± 0,0093	0,000219 ± 0,0097	0,000163 ± 0,0061	0,001490 ± 0,0086	-0,002419 ± 0,0077	0,000436 ± 0,0085	-0,000842 ± 0,0088	0,000260 ± 0,0088
house	0,000586 ± 0,0086	0,000904 ± 0,0081	0,000549 ± 0,0094	0,000077 ± 0,0077	0,001106 ± 0,0087	0,000329 ± 0,0085	0,000413 ± 0,0086	-0,001001 ± 0,0081	0,002044 ± 0,0081

Fonte: Do autor (2017)

Tabela 5.18 – Correlação das imagens cifradas - *grayscale*.

	Trivium			Grain			MICKEY		
Arquivo	**Horz**	**Vert**	**Diag**	**Horz**	**Vert**	**Diag**	**Horz**	**Vert**	**Diag**
5.1.09	0,000975 ± 0,0150	0,002444 ± 0,0140	0,000191 ± 0,0150	-0,002322 ± 0,0146	0,000359 ± 0,0127	-0,001816 ± 0,0145	-0,002514 ± 0,0128	0,003023 ± 0,0171	-0,000138 ± 0,0126
5.1.10	0,002624 ± 0,0131	0,001146 ± 0,0123	0,000627 ± 0,0153	-0,000188 ± 0,0140	0,001720 ± 0,0145	-0,002676 ± 0,0152	-0,000327 ± 0,0147	0,002403 ± 0,0133	0,000794 ± 0,0127
5.1.11	0,000567 ± 0,0150	0,000380 ± 0,0128	-0,000223 ± 0,0129	0,000165 ± 0,0148	0,001326 ± 0,0147	-0,001650 ± 0,0128	0,002510 ± 0,0137	0,001428 ± 0,0154	-0,000590 ± 0,0161
5.1.12	0,000653 ± 0,0134	0,002500 ± 0,0147	0,001834 ± 0,0141	-0,001919 ± 0,0144	0,004227 ± 0,0126	-0,003636 ± 0,0145	0,001852 ± 0,0122	-0,000695 ± 0,0143	-0,001005 ± 0,0152
5.1.13	0,001380 ± 0,0162	0,001817 ± 0,0115	0,001602 ± 0,0142	-0,001077 ± 0,0121	0,004640 ± 0,0152	-0,003708 ± 0,0134	-0,000694 ± 0,0132	-0,001178 ± 0,0152	0,000350 ± 0,0137
5.1.14	-0,001896 ± 0,0135	0,002332 ± 0,0142	-0,003635 ± 0,0121	-0,001199 ± 0,0141	-0,000259 ± 0,0139	-0,002020 ± 0,0151	0,001356 ± 0,0121	-0,000900 ± 0,0157	0,002095 ± 0,0146
5.2.08	0,000934 ± 0,0165	-0,004065 ± 0,0153	-0,000624 ± 0,0155	0,000724 ± 0,0138	0,001612 ± 0,0149	0,001032 ± 0,0150	0,002256 ± 0,0157	-0,000648 ± 0,0139	-0,000312 ± 0,0155
5.2.09	0,002145 ± 0,0152	0,001938 ± 0,0133	0,000067 ± 0,0132	-0,000933 ± 0,0147	-0,001825 ± 0,0122	0,000669 ± 0,0145	0,000782 ± 0,0114	0,000207 ± 0,0154	-0,000123 ± 0,0143
5.2.10	0,002256 ± 0,0152	-0,004124 ± 0,0116	0,000465 ± 0,0136	-0,000046 ± 0,0150	0,001262 ± 0,0141	0,000221 ± 0,0143	0,002289 ± 0,0173	-0,000954 ± 0,0131	-0,002748 ± 0,0143
5.3.01	-0,001789 ± 0,0144	-0,000407 ± 0,0132	0,003963 ± 0,0148	-0,001386 ± 0,0113	0,001070 ± 0,0119	-0,000513 ± 0,0132	0,001847 ± 0,0139	0,001453 ± 0,0148	0,001074 ± 0,0150
5.3.02	-0,002305 ± 0,0129	0,001529 ± 0,0160	0,005769 ± 0,0148	-0,000245 ± 0,0145	-0,000775 ± 0,0150	-0,000423 ± 0,0135	0,002939 ± 0,0129	-0,002537 ± 0,0146	0,002025 ± 0,0154
7.1.01	-0,002890 ± 0,0151	-0,004169 ± 0,0136	-0,000187 ± 0,0145	0,002120 ± 0,0146	-0,000752 ± 0,0156	-0,003492 ± 0,0145	0,000322 ± 0,0133	0,001588 ± 0,0134	-0,000420 ± 0,0158
7.1.02	-0,000756 ± 0,0158	-0,002667 ± 0,0134	0,000298 ± 0,0155	-0,000433 ± 0,0123	0,000193 ± 0,0145	-0,002695 ± 0,0152	-0,000637 ± 0,0169	-0,001005 ± 0,0138	0,001643 ± 0,0140
7.1.03	-0,001292 ± 0,0173	-0,002811 ± 0,0143	0,000529 ± 0,0171	-0,000158 ± 0,0133	0,001621 ± 0,0134	-0,002899 ± 0,0149	0,001927 ± 0,0142	0,000471 ± 0,0120	-0,001055 ± 0,0140
7.1.04	-0,003131 ± 0,0146	-0,005295 ± 0,0127	0,002684 ± 0,0143	0,002427 ± 0,0131	0,003092 ± 0,0135	-0,001691 ± 0,0148	0,001761 ± 0,0134	0,001044 ± 0,0151	0,000700 ± 0,0130
7.1.05	-0,000491 ± 0,0139	-0,001897 ± 0,0129	0,003848 ± 0,0128	0,002763 ± 0,0114	0,001771 ± 0,0129	0,000120 ± 0,0163	0,001962 ± 0,0142	-0,000857 ± 0,0129	0,002067 ± 0,0134
7.1.06	0,000441 ± 0,0146	-0,001428 ± 0,0137	0,002012 ± 0,0180	-0,000525 ± 0,0108	-0,000158 ± 0,0148	-0,000947 ± 0,0123	0,001679 ± 0,0135	-0,002374 ± 0,0128	-0,000943 ± 0,0158
7.1.07	-0,000545 ± 0,0140	-0,001025 ± 0,0126	0,000168 ± 0,0140	-0,002214 ± 0,0135	-0,000769 ± 0,0132	-0,000722 ± 0,0141	0,001088 ± 0,0131	0,001606 ± 0,0159	0,000857 ± 0,0141
7.1.08	-0,001713 ± 0,0151	-0,002608 ± 0,0129	0,001027 ± 0,0131	-0,000283 ± 0,0127	0,000909 ± 0,0127	-0,001316 ± 0,0153	0,003337 ± 0,0180	0,000280 ± 0,0149	0,001909 ± 0,0133
7.1.09	-0,001559 ± 0,0157	-0,001121 ± 0,0145	0,001058 ± 0,0141	0,000287 ± 0,0136	0,000312 ± 0,0128	0,001031 ± 0,0165	0,002527 ± 0,0155	0,000146 ± 0,0149	-0,000973 ± 0,0155
7.1.10	0,000104 ± 0,0139	-0,001262 ± 0,0140	-0,001699 ± 0,0147	0,000349 ± 0,0122	0,000286 ± 0,0123	-0,001595 ± 0,0144	0,002274 ± 0,0139	-0,002787 ± 0,0142	0,001312 ± 0,0137
7.2.01	-0,002217 ± 0,0154	0,003714 ± 0,0156	0,004410 ± 0,0129	-0,000459 ± 0,0146	0,000295 ± 0,0143	-0,002521 ± 0,0128	0,003013 ± 0,0121	0,002265 ± 0,0151	0,001363 ± 0,0165
boat.512	-0,000527 ± 0,0135	-0,001804 ± 0,0146	-0,000186 ± 0,0143	-0,000458 ± 0,0120	-0,000953 ± 0,0151	-0,000136 ± 0,0145	0,003101 ± 0,0138	-0,001335 ± 0,0128	-0,001126 ± 0,0144
elaine.512	-0,000697 ± 0,0138	-0,004354 ± 0,0138	0,002393 ± 0,0139	0,000247 ± 0,0124	0,003093 ± 0,0156	-0,000257 ± 0,0147	0,000989 ± 0,0143	-0,001635 ± 0,0144	-0,000076 ± 0,0120
gray21.512	0,001265 ± 0,0157	0,001121 ± 0,0147	0,000549 ± 0,0172	0,000983 ± 0,0147	0,000184 ± 0,0133	0,001039 ± 0,0146	-0,000390 ± 0,0140	-0,001287 ± 0,0135	0,001165 ± 0,0147
numbers.512	0,001532 ± 0,0140	-0,001235 ± 0,0136	0,000078 ± 0,0151	-0,000193 ± 0,0096	0,004322 ± 0,0162	0,001456 ± 0,0156	0,003258 ± 0,0125	-0,002573 ± 0,0129	0,000755 ± 0,0142
ruler.512	0,005098 ± 0,0138	0,004412 ± 0,0159	-0,001250 ± 0,0130	0,001101 ± 0,0147	-0,000086 ± 0,0132	0,001180 ± 0,0117	-0,001395 ± 0,0122	-0,002032 ± 0,0156	0,000426 ± 0,0168
testpat.1k	0,001227 ± 0,0140	0,002198 ± 0,0141	0,007107 ± 0,0129	0,001164 ± 0,0155	0,001571 ± 0,0145	-0,000519 ± 0,0135	0,001136 ± 0,0145	-0,000185 ± 0,0143	0,001767 ± 0,0156

Fonte: Do autor (2017)

Tabela 5.19 – Correlação das imagens originais - *grayscale*.

Arquivo	Horz	Vert	Diag
5.1.09	0,901067	0,938735	0,902345
5.1.10	0,904973	0,858460	0,822875
5.1.11	0,957352	0,936426	0,892906
5.1.12	0,956422	0,973924	0,938591
5.1.13	0,870743	0,865906	0,754070
5.1.14	0,946202	0,898915	0,852829
5.2.08	0,936386	0,889866	0,857222
5.2.09	0,900711	0,861543	0,803352
5.2.10	0,939777	0,928131	0,897543
5.3.01	0,977117	0,981371	0,967051
5.3.02	0,909940	0,903223	0,859077
7.1.01	0,961879	0,920185	0,907177
7.1.02	0,943854	0,945249	0,898709
7.1.03	0,946199	0,932608	0,901986
7.1.04	0,976871	0,968086	0,955164
7.1.05	0,941382	0,912633	0,893636
7.1.06	0,939787	0,906348	0,886871
7.1.07	0,885908	0,877992	0,839258
7.1.08	0.957898	0.929142	0.922962
7.1.09	0,965641	0,930384	0,916069
7.1.10	0,964246	0,947565	0,931634
7.2.01	0,964735	0,947122	0,944765
boat.512	0,937101	0,971190	0,922101
elaine.512	0,975818	0,973217	0,969416
gray21.512	0,996556	0,999842	0,996442
numbers.512	0,741318	0,716631	0,627326
ruler.512	0,451302	0,463764	-0,025121
testpat.1k	0,762352	0,798151	0,698064

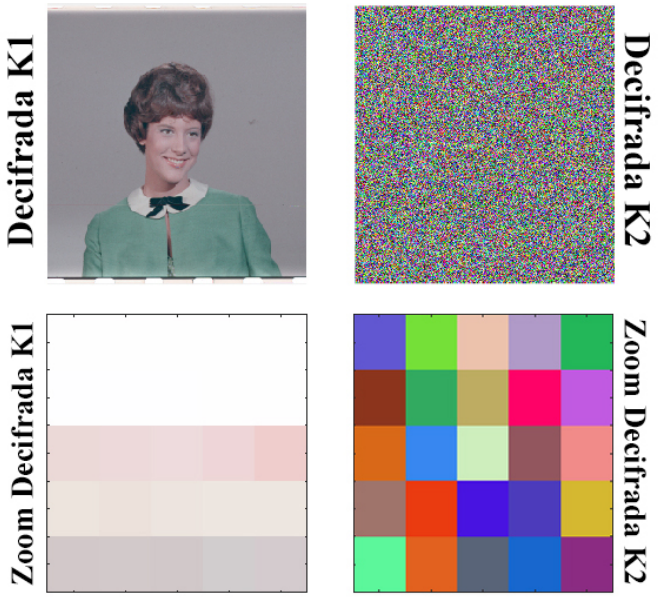
Fonte: Do autor (2017)

5.3.5 Análise de Sensibilidade

Nessa análise, a Tabela 5.20 e a Tabela 5.21 mostram o valor de correlação para imagens coloridas, no processo de cifragem e decifragem, respectivamente. Já a Tabela 5.22 e a Tabela 5.23 mostram esses resultados para imagens em tons de cinza. A análise dessas tabelas mostra que todos os três algoritmos possuem correlação próxima de 0, reforçando a qualidade desses nessa aplicação.

Além disso, a Figura 5.17 demonstra a alta sensibilidade encontrada no processo de decifragem utilizando o algoritmo Trivium. Na imagem de título "Decifrada K1" a amostra "4.2.03" do *dataset* é decifrada com a chave original (K1), em detalhe na figura "Zoom Decifrada K1" é mostrada essa imagem ampliada. Já a imagem "Decifrada K2" exibe a mesma amostra "4.2.03", porém, decifrada com a chave alterada em um *bit* (K2). Comparando as imagens resultantes dos processos de decifragem observa-se que elas não são de modo algum semelhantes. Em destaque nas tabelas estão os valores encontrados para as imagens exibidas no processo de análise visual na Subseção 5.3.1.

Figura 5.17 – Alta sensibilidade a variação de chave.



Fonte: Do autor (2017)

Tabela 5.20 – Correlação entre a imagem cifrada com a chave original (K) e a imagem cifrada com chave K modificada em apenas 1 *bit* (K') - RGB.

Arquivo	Trivium	Grain	MICKEY
4.1.01	0,000004 ± 0,0020	0,000020 ± 0,0022	0,000006 ± 0,0025
4.1.02	-0,000259 ± 0,0022	0,000045 ± 0,0027	-0,000284 ± 0,0022
4.1.03	-0,000217 ± 0,0020	-0,000129 ± 0,0021	0,000279 ± 0,0025
4.1.04	-0,000227 ± 0,0017	-0,000007 ± 0,0024	0,000414 ± 0,0024
4.1.05	0,000148 ± 0,0019	0,000246 ± 0,0019	0,000414 ± 0,0022
4.1.06	-0,000177 ± 0,0023	0,000537 ± 0,0023	0,000046 ± 0,0023
4.1.07	-0,000131 ± 0,0019	0,000106 ± 0,0024	0,000128 ± 0,0023
4.1.08	-0,000269 ± 0,0021	0,000048 ± 0,0024	0,000114 ± 0,0024

Arquivo	Trivium	Grain	MICKEY
4.2.01	-0,000071 ± 0,0012	-0,000107 ± 0,0012	0,000016 ± 0,0010
4.2.02	-0,000123 ± 0,0012	-0,000151 ± 0,0010	-0,000123 ± 0,0010
4.2.03	-0,000046 ± 0,0010	-0,000309 ± 0,0010	-0,000197 ± 0,0010
4.2.04	0,000134 ± 0,0012	-0,000190 ± 0,0011	-0,000217 ± 0,0009
4.2.05	0,000036 ± 0,0013	-0,000198 ± 0,0011	0,000008 ± 0,0010
4.2.06	-0,000046 ± 0,0012	-0,000232 ± 0,0012	-0,000252 ± 0,0011
4.2.07	-0,000121 ± 0,0011	0,000062 ± 0,0013	-0,000132 ± 0,0011
house	-0.000174 ± 0.0011	-0,000297 ± 0,0013	-0,000140 ± 0,0012

Fonte: Do autor (2017)

Tabela 5.21 – Correlação entre a imagem decifrada com a chave original (K) e a imagem decifrada com a chave K modificada em apenas 1 *bit* (K') - RGB.

Arquivo	Trivium	Grain	MICKEY
4.1.01	-0,000516 ± 0,0030	-0,000287 ± 0,0030	0,000561 ± 0,0029
4.1.02	-0,000392 ± 0,0035	-0,000544 ± 0,0034	-0,000128 ± 0,0030
4.1.03	-0,000333 ± 0,0027	-0,000126 ± 0,0028	-0,000275 ± 0,0031
4.1.04	-0,000166 ± 0,0026	-0,000516 ± 0,0029	0,000187 ± 0,0030
4.1.05	-0,000243 ± 0,0030	-0,000127 ± 0,0032	-0,000095 ± 0,0031
4.1.06	-0,000324 ± 0,0028	0,000124 ± 0,0028	-0,000377 ± 0,0028
4.1.07	-0,000532 ± 0,0027	-0,000172 ± 0,0029	-0,000289 ± 0,0028
4.1.08	0,000294 ± 0,0032	-0,000312 ± 0,0030	-0,000625 ± 0,0032

Arquivo	Trivium	Grain	MICKEY
4.2.01	-0,000179 ± 0,0014	-0,000071 ± 0,0013	0,000057 ± 0,0015
4.2.02	0,000254 ± 0,0015	0,000189 ± 0,0014	-0,000013 ± 0,0015
4.2.03	0,000016 ± 0,0015	-0,000092 ± 0,0015	-0,000016 ± 0,0012
4.2.04	0,000064 ± 0,0013	-0,000243 ± 0,0015	0,000033 ± 0,0014
4.2.05	-0,000346 ± 0,0016	-0,000048 ± 0,0014	0,000137 ± 0,0012
4.2.06	-0,000329 ± 0,0013	-0,000111 ± 0,0014	0,000019 ± 0,0013
4.2.07	-0,000277 ± 0,0015	-0,000036 ± 0,0015	-0,000065 ± 0,0012
house	0,000002 ± 0,0015	-0,000090 ± 0,0014	0,000110 ± 0,0015

Fonte: Do autor (2017)

Tabela 5.22 – Correlação entre a imagem cifrada com a chave original (K) e a imagem cifrada com a chave K modificada em apenas 1 *bit* (K') - *grayscale*.

Arquivo	Trivium	Grain	MICKEY
5.1.09	0,000270 ± 0,0035	-0,000488 ± 0,0036	-0,000311 ± 0,0046
5.1.10	0,000251 ± 0,0037	-0,001121 ± 0,0042	-0,000384 ± 0,0045
5.1.11	-0,000698 ± 0,0035	-0,001116 ± 0,0040	-0,000748 ± 0,0040
5.1.12	-0,000221 ± 0,0030	0,000003 ± 0,0039	-0,000574 ± 0,0040
5.1.13	-0,000768 ± 0,0033	0,000010 ± 0,0047	-0,000760 ± 0,0042
5.1.14	-0,000549 ± 0,0035	-0,000805 ± 0,0039	-0,000148 ± 0,0039
5.2.08	-0,000087 ± 0,0019	-0,000341 ± 0,0020	0,000282 ± 0,0023
5.2.09	-0,000117 ± 0,0020	0,000081 ± 0,0021	-0,000453 ± 0,0020
5.2.10	-0,000051 ± 0,0018	-0,000081 ± 0,0018	-0,000029 ± 0,0018
5.3.01	0,000093 ± 0,0009	-0,000018 ± 0,0010	-0,000249 ± 0,0009
5.3.02	-0,000047 ± 0,0009	-0,000029 ± 0,0010	-0,000209 ± 0,0010
7.1.01	-0,000063 ± 0,0017	-0,000432 ± 0,0018	0,000243 ± 0,0020
7.1.02	0,000058 ± 0,0018	-0,000401 ± 0,0021	-0,000183 ± 0,0024
7.1.03	0,000033 ± 0,0018	-0,000361 ± 0,0019	0,000107 ± 0,0022

Arquivo	Trivium	Grain	MICKEY
7.1.04	-0,000072 ± 0,0017	-0,000155 ± 0,0020	0,000264 ± 0,0020
7.1.05	0,000134 ± 0,0018	-0,000326 ± 0,0019	0,000103 ± 0,0023
7.1.06	-0,000117 ± 0,0017	-0,000238 ± 0,0021	0,000323 ± 0,0021
7.1.07	0,000105 ± 0,0018	-0,000443 ± 0,0018	0,000156 ± 0,0020
7.1.08	-0,000074 ± 0,0018	-0,000367 ± 0,0019	0,000119 ± 0,0021
7.1.09	-0,000158 ± 0,0015	-0,000088 ± 0,0017	-0,000071 ± 0,0021
7.1.10	-0,000245 ± 0,0016	-0,000425 ± 0,0020	-0,000047 ± 0,0022
7.2.01	-0,000089 ± 0,0011	0,000021 ± 0,0010	-0,000233 ± 0,0009
boat.512	-0,000018 ± 0,0020	0,000075 ± 0,0018	0,000210 ± 0,0022
elaine.512	0,000108 ± 0,0017	-0,000338 ± 0,0021	0,000050 ± 0,0022
gray21.512	-0,000129 ± 0,0014	0,000140 ± 0,0021	0,000131 ± 0,0020
numbers.512	-0,000195 ± 0,0018	0,000116 ± 0,0019	-0,000087 ± 0,0018
ruler.512	-0,000004 ± 0,0017	0,000202 ± 0,0021	-0,000116 ± 0,0016
testpat.1k	0,000194 ± 0,0009	-0,000023 ± 0,0009	-0,000204 ± 0,0009

Fonte: Do autor (2017)

Tabela 5.23 – Correlação entre a imagem decifrada com a chave original (K) e a imagem decifrada com a chave K modificada em apenas 1 *bit* (K') - *grayscale*.

Arquivo	Trivium	Grain	MICKEY
5.1.09	-0,001203 ± 0,0032	-0,000066 ± 0,0039	-0,000381 ± 0,0051
5.1.10	-0,000647 ± 0,0039	0,000435 ± 0,0036	-0,000314 ± 0,0039
5.1.11	-0,000289 ± 0,0031	-0,000883 ± 0,0034	-0,000692 ± 0,0039
5.1.12	-0,000284 ± 0,0033	-0,000547 ± 0,0042	-0,000558 ± 0,0037
5.1.13	-0,000251 ± 0,0036	0,000099 ± 0,0041	-0,001243 ± 0,0042
5.1.14	-0,000836 ± 0,0042	-0,000235 ± 0,0033	-0,000276 ± 0,0041
5.2.08	-0,000089 ± 0,0016	0,000403 ± 0,0017	0,000031 ± 0,0023
5.2.09	0,000041 ± 0,0017	0,000245 ± 0,0016	0,000049 ± 0,0020
5.2.10	-0,000196 ± 0,0020	0,000241 ± 0,0021	0,000080 ± 0,0017
5.3.01	-0,000090 ± 0,0008	-0,000246 ± 0,0010	-0,000075 ± 0,0010
5.3.02	0,000085 ± 0,0009	-0,000428 ± 0,0009	-0,000148 ± 0,0008
7.1.01	-0,000080 ± 0,0018	-0,000185 ± 0,0020	0,000087 ± 0,0019
7.1.02	0,000056 ± 0,0021	0,000094 ± 0,0019	-0,000276 ± 0,0018
7.1.03	-0,000226 ± 0,0017	-0,000038 ± 0,0019	0,000287 ± 0,0021
7.1.04	-0,000360 ± 0,0017	0,000220 ± 0,0019	0,000041 ± 0,0019
7.1.05	-0,000067 ± 0,0020	0,000066 ± 0,0018	-0,000061 ± 0,0018
7.1.06	-0,000224 ± 0,0021	0,000068 ± 0,0020	0,000351 ± 0,0019
7.1.07	-0,000367 ± 0,0019	0,000170 ± 0,0020	0,000023 ± 0,0020
7.1.08	0,000096 ± 0,0015	0,000210 ± 0,0016	0,000186 ± 0,0016
7.1.09	-0,000451 ± 0,0017	-0,000087 ± 0,0017	0,000142 ± 0,0022
7.1.10	-0.000039 ± 0,0018	0,000322 ± 0,0018	-0,000190 ± 0,0018
7.2.01	-0,000071 ± 0,0009	-0,000085 ± 0,0009	-0,000055 ± 0,0010
boat.512	-0,000204 ± 0,0018	-0,000123 ± 0,0017	0,000148 ± 0,0018
elaine.512	-0,000094 ± 0,0018	-0,000092 ± 0,0018	-0,000057 ± 0,0023
gray21.512	-0,000093 ± 0,0020	-0,000109 ± 0,0022	-0,000131 ± 0,0019
numbers.512	0,000131 ± 0,0017	-0,000575 ± 0,0021	-0,000155 ± 0,0019
ruler.512	0,000075 ± 0,0023	-0,000139 ± 0,0020	-0,000356 ± 0,0019
testpat.1k	-0,000184 ± 0,0009	-0,000037 ± 0,0010	-0,000187 ± 0,0009

Fonte: Do autor (2017)

5.3.6 Análise Diferencial

Pequenas alterações na imagem original devem resultar em alterações significativas na imagem cifrada para que sejam evitadas quaisquer relações estatísticas entre a entrada e a saída do sistema de criptografia, A sensibilidade de um sistema quanto aos dados de entrada pode ser calculado pelas métricas *Unified Averaged Changed Intensity (UACI)* e *Number of Changing Pixel Rate (NPCR)* (MANSINGKA et al., 2014). Wu, Noonan e Agaian (2011) estabelece limites críticos para as métricas NPCR e UACI para a aceitação de hipótese nula N^*_α com um certo grau de significância. Assim, dada uma amostra x, a probabilidade de dizer que x não vem de uma fonte aleatória, quando na verdade ela vem de fato, é α, que pode ser 0.05, 0.01 ou 0.001. Os valores recomendados na literatura são exibidos na Tabela 5.24 e na Tabela 5.25.

Tabela 5.24 – Valores mínimos recomendados para NPCR.

Tamanho	$\mathbf{N^*_{0,05}}$	$\mathbf{N^*_{0,01}}$	$\mathbf{N^*_{0,001}}$
256x256	99,5693%	99,5527%	99,5341%
512x512	99,5893%	99,5810%	99,5717%
1024x1024	99,5994%	99,5952%	99,5906%

Fonte: Adaptado de (WU; NOONAN; AGAIAN, 2011)

Tabela 5.25 – Intervalos recomendados para UACI.

Tamanho	$\mathbf{U^{*-}_{0,05}}$ / $\mathbf{U^{*+}_{0,05}}$	$\mathbf{U^{*-}_{0,01}}$ / $\mathbf{U^{*+}_{0,01}}$	$\mathbf{U^{*-}_{0,001}}$ / $\mathbf{U^{*+}_{0,001}}$
256x256	$32,2824\%/33,6447\%$	$33,2255\%/33,7016\%$	$33,1594\%/33,7677\%$
512x512	$33,3730\%/33,5541\%$	$33,3445\%/33,5826\%$	$33,3115\%/33,6156\%$
1024x1024	$33,4183\%/33,5088\%$	$33,4040\%/33,5231\%$	$33,3875\%/33,5396\%$

Fonte: Adaptado de (WU; NOONAN; AGAIAN, 2011)

5.3.6.1 NPCR

A Tabela 5.26 e a Tabela 5.27 exibem o valor da métrica NPCR das imagens coloridas cifradas, para todos os canais e para cada canal de cor separado, respectivamente. Já a Tabela 5.28 exibe essa métrica para as imagens em escala de cinza. Os valores encontrados para NPCR das imagens são maiores que os limiares críticos $N^*_{0.05}$, $N^*_{0.01}$ e $N^*_{0.001}$. Essa análise demonstra que ambos os algoritmos são extremamente sensíveis a pequenas mudanças na imagem original, garantindo assim a segurança. Em destaque nas tabelas estão os valores encontrados para as imagens exibidas no processo de análise visual na Subseção 5.3.1.

Tabela 5.26 – NPCR de cada canal RGB das imagens cifradas.

Arquivo (tamanho)	Trivium			Grain			MICKEY		
	Red	**Green**	**Blue**	**Red**	**Green**	**Blue**	**Red**	**Green**	**Blue**
4.1.01	0,996131	0,996056	0,996068	0,996123	0,996117	0,996062	0,996108	0,996086	0,996149
(256)	± 0,0003	± 0,0002	± 0,0002	± 0,0003	± 0,0003	± 0,0003	± 0,0003	± 0,0002	± 0,0003
4.1.02	0,996131	0,996056	0,996068	0,996123	0,996117	0,996062	0,996108	0,996086	0,996149
(256)	± 0,0003	± 0,0002	± 0,0002	± 0,0003	± 0,0003	± 0,0003	± 0,0003	± 0,0002	± 0,0003
4.1.03	0,996131	0,996056	0,996068	0,996123	0,996117	0,996062	0,996108	0,996086	0,996149
(256)	± 0,0003	± 0,0002	± 0,0002	± 0,0003	± 0,0003	± 0,0003	± 0,0003	± 0,0002	± 0,0003
4.1.04	0,996131	0,996056	0,996068	0,996123	0,996117	0,996062	0,996108	0,996086	0,996149
(256)	± 0,0003	± 0,0002	± 0,0002	± 0,0003	± 0,0003	± 0,0003	± 0,0003	± 0,0002	± 0,0003
4.1.05	0,996131	0,996056	0,996068	0,996123	0,996117	0,996061	0,996108	0,996086	0,996149
(256)	± 0,0003	± 0,0002	± 0,0002	± 0,0003	± 0,0003	± 0,0003	± 0,0003	± 0,0002	± 0,0003
4.1.06	0,996131	0,996056	0,996068	0,996123	0,996116	0,996062	0,996108	0,996086	0,996149
(256)	± 0,0003	± 0,0002	± 0,0002	± 0,0003	± 0,0003	± 0,0003	± 0,0003	± 0,0002	± 0,0003
4.1.07	0,996131	0,996056	0,996068	0,996123	0,996117	0,996062	0,996108	0,996086	0,996149
(256)	± 0,0003	± 0,0002	± 0,0002	± 0,0003	± 0,0003	± 0,0003	± 0,0003	± 0,0002	± 0,0003
4.1.08	0,996131	0,996056	0,996068	0,996123	0,996116	0,996062	0,996108	0,996086	0,996149
(256)	± 0,0003	± 0,0002	± 0,0002	± 0,0003	± 0,0003	± 0,0003	± 0,0003	± 0,0002	± 0,0003
4.2.01	0,996065	0,996095	0,996092	0,996114	0,996082	0,996065	0,996108	0,996084	0,996098
(512)	± 0,0001	± 0,0001	± 0,0002	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001
4.2.02	0,996065	0,996095	0,996092	0,996114	0,996082	0,996065	0,996108	0,996084	0,996098
(512)	± 0,0001	± 0,0001	± 0,0002	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001
4.2.03	0,996065	0,996095	0,996092	0,996114	0,996082	0,996065	0,996108	0,996084	0,996098
(512)	± 0,0001	± 0,0001	± 0,0002	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001
4.2.04	0,996065	0,996095	0,996092	0,996114	0,996082	0,996065	0,996108	0,996084	0,996098
(512)	± 0,0001	± 0,0001	± 0,0002	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001
4.2.05	0,996065	0,996095	0,996092	0,996114	0,996082	0,996065	0,996108	0,996084	0,996098
(512)	± 0,0001	± 0,0001	± 0,0002	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001
4.2.06	0,996065	0,996095	0,996092	0,996114	0,996082	0,996065	0,996107	0,996084	0,996098
(512)	± 0,0001	± 0,0001	± 0,0002	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001
4.2.07	0,996065	0,996095	0,996092	0,996114	0,996082	0,996065	0,996108	0,996084	0,996098
(512)	± 0,0001	± 0,0001	± 0,0002	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001
house	0,996065	0,996095	0,996092	0,996114	0,996082	0,996065	0,996108	0,996084	0,996098
(512)	± 0,0001	± 0,0001	± 0,0002	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001	± 0,0001

Fonte: Do autor (2017)

Tabela 5.27 – NPCR de todos os canais das imagens cifradas - RGB.

Arquivo (tamanho)	Trivium	Grain	MICKEY
4.1.01 (256)	0,996085 ± 0,0001	0,996101 ± 0,0002	0,996114 ± 0,0002
4.1.02 (256)	0,996085 ± 0,0001	0,996101 ± 0,0002	0,996114 ± 0,0002
4.1.03 (256)	0,996085 ± 0,0001	0,996101 ± 0,0002	0,996114 ± 0,0002
4.1.04 (256)	0,996085 ± 0,0001	0,996101 ± 0,0002	0,996114 ± 0,0002
4.1.05 (256)	0,996085 ± 0,0001	0,996100 ± 0,0002	0,996114 ± 0,0002
4.1.06 (256)	0,996085 ± 0,0001	0,996100 ± 0,0002	0,996114 ± 0,0002
4.1.07 (256)	0,996085 ± 0,0001	0,996101 ± 0,0002	0,996114 ± 0,0002
4.1.08 (256)	0,996085 ± 0,0001	0,996100 ± 0,0002	0,996114 ± 0,0002
4.2.01 (512)	0,996084 ± 0,0001	0,996087 ± 0,0001	0,996096 ± 0,0001
4.2.02 (512)	0,996084 ± 0,0001	0,996087 ± 0,0001	0,996096 ± 0,0001
4.2.03 (512)	0,996084 ± 0,0001	0,996087 ± 0,0001	0,996096 ± 0,0001
4.2.04 (512)	0,996084 ± 0,0001	0,996087 ± 0,0001	0,996096 ± 0,0001
4.2.05 (512)	0,996084 ± 0,0001	0,996087 ± 0,0001	0,996096 ± 0,0001
4.2.06 (512)	0,996084 ± 0,0001	0,996087 ± 0,0001	0,996096 ± 0,0001
4.2.07 (512)	0,996084 ± 0,0001	0,996087 ± 0,0001	0,996096 ± 0,0001
house (512)	0,996084 ± 0,0001	0,996087 ± 0,0001	0,996096 ± 0,0001

Fonte: Do autor (2017)

Tabela 5.28 – NPCR das imagens cifradas - *grayscale*,

Arquivo (tamanho)	Trivium	Grain	MICKEY
5.1.09 (256)	0,996131 ± 0,0003	0,996123 ± 0,0003	0,996108 ± 0,0003
5.1.10 (256)	0,996131 ± 0,0003	0,996123 ± 0,0003	0,996108 ± 0,0003
5.1.11 (256)	0,996131 ± 0,0003	0,996123 ± 0,0003	0,996108 ± 0,0003
5.1.12 (256)	0,996131 ± 0,0003	0,996123 ± 0,0003	0,996108 ± 0,0003
5.1.13 (256)	0,996131 ± 0,0003	0,996123 ± 0,0003	0,996108 ± 0,0003
5.1.14 (256)	0,996131 ± 0,0003	0,996123 ± 0,0003	0,996108 ± 0,0003
5.2.08 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
5.2.09 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
5.2.10 (512)	0,996064 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
5.3.01 (1024)	0,996085 ± 0,0001	0,996090 ± 0,0001	0,996094 ± 0,0001
5.3.02 (1024)	0,996085 ± 0,0001	0,996090 ± 0,0001	0,996094 ± 0,0001
7.1.01 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
7.1.02 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
7.1.03 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996107 ± 0,0001
7.1.04 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996107 ± 0,0001
7.1.05 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
7.1.06 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996107 ± 0,0001
7.1.07 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
7.1.08 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
7.1.09 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996107 ± 0,0001
7.1.10 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
7.2.01 (1024)	0,996085 ± 0,0001	0,996090 ± 0,0001	0,996094 ± 0,0001
boat.512 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
elaine.512 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
gray21.512 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
numbers.512 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
ruler.512 (512)	0,996065 ± 0,0001	0,996114 ± 0,0001	0,996108 ± 0,0001
testpat.1k (1024)	0,996085 ± 0,0001	0,996090 ± 0,0001	0,996094 ± 0,0001

Fonte: Do autor (2017)

5.3.6.2 UACI

A Tabela 5.29 e Tabela 5.30 exibem o valor da métrica UACI das imagens coloridas cifradas, para todos os canais e para cada canal de cor separado, respectivamente. Já a Tabela 5.31 exibe essa métrica para as imagens em escala de cinza. De maneira análoga. Os valores encontrados para UACI encontram-se contidos no intervalos de confiança estabelecidos na 5.25. Essa análise, assim como a realizada para a métrica NPCR, demonstra que ambos os algoritmos são extremamente sensíveis a pequenas mudanças na imagem original, garantindo assim a segurança. Em destaque nas tabelas estão os valores encontrados para as imagens exibidas no processo de análise visual na Subseção 5.3.1.

Tabela 5.29 – UACI de cada canal RGB das imagens cifradas.

Arquivo (tamanho)	Trivium			Grain			MICKEY		
	Red	**Green**	**Blue**	**Red**	**Green**	**Blue**	**Red**	**Green**	**Blue**
4.1.01	0,334824	0,334629	0,334491	0,334673	0,334560	0,334686	0,334617	0,334541	0,334763
(256)	± 0,0008	± 0,0008	± 0,0008	± 0,0008	± 0,0008	± 0,0011	± 0,0009	± 0,0010	± 0,0010
4.1.02	0,334747	0,334522	0,334599	0,334648	0,334578	0,334587	0,334623	0,334607	0,334697
(256)	± 0,0009	± 0,0007	± 0,0009	± 0,0010	± 0,0009	± 0,0010	± 0,0008	± 0,0010	± 0,0009
4.1.03	0,334795	0,334582	0,334680	0,334646	0,334440	0,334673	0,334463	0,334399	0,334919
(256)	± 0,0010	± 0,0010	± 0,0008	± 0,0009	± 0,0008	± 0,0011	± 0,0009	± 0,0011	± 0,0009
4.1.04	0,334779	0,334594	0,334704	0,334755	0,334705	0,334621	0,334484	0,334488	0,334889
(256)	± 0,0009	± 0,0009	± 0,0010	± 0,0008	± 0,0009	± 0,0011	± 0,0009	± 0,0011	± 0,0009
4.1.05	0,334814	0,334426	0,334792	0,334658	0,334671	0,334691	0,334507	0,334407	0,334738
(256)	± 0,0009	± 0,0009	± 0,0008	± 0,0009	± 0,0008	± 0,0010	± 0,0009	± 0,0009	± 0,0009
4.1.06	0,334932	0,334519	0,334720	0,334695	0,334732	0,334578	0,334465	0,334441	0,334895
(256)	± 0,0009	± 0,0008	± 0,0010	± 0,0009	± 0,0010	± 0,0011	± 0,0008	± 0,0010	± 0,0010
4.1.07	0,334920	0,334434	0,334610	0,334778	0,334493	0,334720	0,334487	0,334616	0,334810
(256)	± 0,0008	± 0,0009	± 0,0010	± 0,0009	± 0,0010	± 0,0010	± 0,0011	± 0,0009	± 0,0008
4.1.08	0,334871	0,334402	0,334600	0,334720	0,334510	0,334652	0,334464	0,334467	0,334810
(256)	± 0,0008	± 0,0009	± 0,0010	± 0,0009	± 0,0011	± 0,0009	± 0,0009	± 0,0010	± 0,0010
4.2.01	0,334660	0,334616	0,334670	0,334610	0,334519	0,334605	0,334653	0,334625	0,334583
(512)	± 0,0004	± 0,0004	± 0,0005	± 0,0005	± 0,0004	± 0,0005	± 0,0005	± 0,0005	± 0,0005
4.2.02	0,334700	0,334636	0,334672	0,334707	0,334563	0,334671	0,334667	0,334595	0,334659
(512)	± 0,0004	± 0,0004	± 0,0005	± 0,0005	± 0,0004	± 0,0005	± 0,0005	± 0,0004	± 0,0004
4.2.03	0,334665	0,334708	0,334612	0,334665	0,334513	0,334678	0,334636	0,334609	0,334676
(512)	± 0,0004	± 0,0004	± 0,0005	± 0,0005	± 0,0005	± 0,0005	± 0,0004	± 0,0005	± 0,0005
4.2.04	0,334652	0,334658	0,334669	0,334680	0,334550	0,334629	0,334650	0,334639	0,334692
(512)	± 0,0004	± 0,0004	± 0,0005	± 0,0005	± 0,0005	± 0,0005	± 0,0005	± 0,0004	± 0,0005
4.2.05	0,334711	0,334541	0,334583	0,334571	0,334529	0,334566	0,334653	0,334663	0,334523
(512)	± 0,0004	± 0,0004	± 0,0004	± 0,0005	± 0,0004	± 0,0005	± 0,0005	± 0,0005	± 0,0004
4.2.06	0,334765	0,334640	0,334627	0,334661	0,334579	0,334637	0,334633	0,334561	0,334573
(512)	± 0,0004	± 0,0004	± 0,0005	± 0,0004	± 0,0004	± 0,0005	± 0,0004	± 0,0005	± 0,0005
4.2.07	0,334695	0,334596	0,334631	0,334623	0,334503	0,334567	0,334591	0,334616	0,334636
(512)	± 0,0004	± 0,0004	± 0,0005	± 0,0005	± 0,0004	± 0,0005	± 0,0005	± 0,0005	± 0,0004
house	0,334722	0,334684	0,334650	0,334680	0,334576	0,334616	0,334600	0,334581	0,334597
(512)	± 0,0005	± 0,0004	± 0,0004	± 0,0006	± 0,0004	± 0,0005	± 0,0005	± 0,0004	± 0,0005

Fonte: Do autor (2017)

Tabela 5.30 – UACI de todos os canais das imagens cifradas - RGB.

Arquivo (tamanho)	Trivium	Grain	MICKEY
4.1.01 (256)	0,334648 ± 0,0005	0,334640 ± 0,0006	0,334640 ± 0,0006
4.1.02 (256)	0,334623 ± 0,0005	0,334605 ± 0,0006	0,334642 ± 0,0005
4.1.03 (256)	0,334686 ± 0,0004	0,334587 ± 0,0006	0,334593 ± 0,0005
4.1.04 (256)	0,334692 ± 0,0005	0,334694 ± 0,0006	0,334621 ± 0,0006
4.1.05 (256)	0,334677 ± 0,0004	0,334674 ± 0,0005	0,334551 ± 0,0005
4.1.06 (256)	0,334724 ± 0,0005	0,334668 ± 0,0005	0,334600 ± 0,0006
4.1.07 (256)	0,334654 ± 0,0005	0,334664 ± 0,0005	0,334638 ± 0,0006
4.1.08 (256)	0,334624 ± 0,0005	0,334627 ± 0,0005	0,334580 ± 0,0006

Arquivo (tamanho)	Trivium	Grain	MICKEY
4.2.01 (512)	0,334649 ± 0,0002	0,334578 ± 0,0003	0,334620 ± 0,0003
4.2.02 (512)	0,334670 ± 0,0002	0,334647 ± 0,0003	0,334640 ± 0,0002
4.2.03 (512)	0,334661 ± 0,0003	0,334619 ± 0,0003	0,334640 ± 0,0002
4.2.04 (512)	0,334660 ± 0,0002	0,334620 ± 0,0003	0,334660 ± 0,0003
4.2.05 (512)	0,334612 ± 0,0002	0,334555 ± 0,0003	0,334613 ± 0,0003
4.2.06 (512)	0,334677 ± 0,0003	0,334626 ± 0,0003	0,334589 ± 0,0002
4.2.07 (512)	0,334641 ± 0,0002	0,334564 ± 0,0003	0,334614 ± 0,0003
house (512)	0,334685 ± 0,0003	0,334624 ± 0,0003	0,334592 ± 0,0003

Fonte: Do autor (2017)

Tabela 5.31 – UACI das imagens cifradas - *grayscale*.

Arquivo (tamanho)	Trivium	Grain	MICKEY
5.1.09 (256)	0,334825 ± 0,0009	0,334709 ± 0,0009	0,334496 ± 0,0010
5.1.10 (256)	0,334815 ± 0,0009	0,334620 ± 0,0008	0,334674 ± 0,0009
5.1.11 (256)	0,334729 ± 0,0008	0,334803 ± 0,0009	0,334541 ± 0,0009
5.1.12 (256)	0,334804 ± 0,0010	0,334666 ± 0,0008	0,334512 ± 0,0010
5.1.13 (256)	0,334734 ± 0,0008	0,334746 ± 0,0009	0,334684 ± 0,0008
5.1.14 (256)	0,334722 ± 0,0010	0,334619 ± 0,0009	0,334492 ± 0,0009
5.2.08 (512)	0,334683 ± 0,0004	0,334637 ± 0,0005	0,334659 ± 0,0004
5.2.09 (512)	0,334713 ± 0,0004	0,334730 ± 0,0005	0,334653 ± 0,0005
5.2.10 (512)	0,334706 ± 0,0003	0,334676 ± 0,0005	0,334650 ± 0,0005
5.3.01 (1024)	0,334646 ± 0,0002	0,334624 ± 0,0003	0,334666 ± 0,0002
5.3.02 (1024)	0,334667 ± 0,0002	0,334596 ± 0,0002	0,334634 ± 0,0002
7.1.01 (512)	0,334772 ± 0,0004	0,334670 ± 0,0005	0,334615 ± 0,0004
7.1.02 (512)	0,334744 ± 0,0005	0,334681 ± 0,0005	0,334657 ± 0,0004
7.1.03 (512)	0,334741 ± 0,0004	0,334627 ± 0,0005	0,334549 ± 0,0004

Arquivo (tamanho)	Trivium	Grain	MICKEY
7.1.04 (512)	0,334761 ± 0,0004	0,334600 ± 0,0005	0,334577 ± 0,0004
7.1.05 (512)	0,334707 ± 0,0004	0,334670 ± 0,0005	0,334638 ± 0,0005
7.1.06 (512)	0,334710 ± 0,0004	0,334677 ± 0,0005	0,334628 ± 0,0004
7.1.07 (512)	0,334683 ± 0,0004	0,334616 ± 0,0005	0,334621 ± 0.0004
7.1.08 (512)	0,334714 ± 0,0004	0,334660 ± 0,0005	0,334620 ± 0,0004
7.1.09 (512)	0,334763 ± 0,0004	0,334647 ± 0,0005	0,334618 ± 0,0004
7.1.10 (512)	0,334702 ± 0,0004	0,334628 ± 0,0005	0,334609 ± 0,0005
7.2.01 (1024)	0,334596 ± 0,0002	0,334624 ± 0,0002	0,334643 ± 0,0002
boat.512 (512)	0,334730 ± 0,0004	0,334695 ± 0,0004	0,334666 ± 0,0005
elaine.512 (512)	0,334697 ± 0,0004	0,334684 ± 0,0005	0,334573 ± 0,0004
gray21.512 (512)	0,334691 ± 0,0004	0,334685 ± 0,0005	0,334592 ± 0,0005
numbers.512 (512)	0,334744 ± 0,0004	0,334700 ± 0,0005	0,334643 ± 0,0004
ruler.512 (512)	0,334669 ± 0,0004	0,334709 ± 0,0005	0,334673 ± 0,0005
testpat.1k (1024)	0,334634 ± 0,0002	0,334587 ± 0,0002	0,334628 ± 0,0003

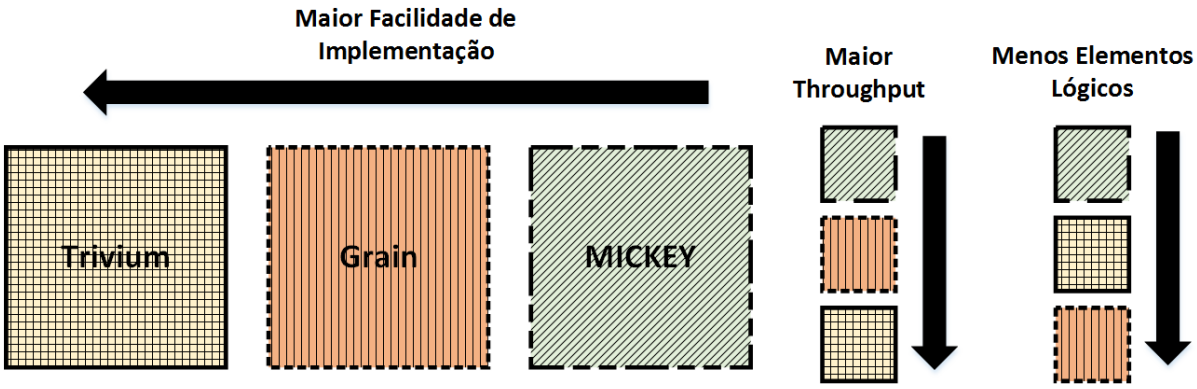
Fonte: Do autor (2017)

6 CONSIDERAÇÕES

Após o desenvolvimento da plataforma de análises e a execução de todos os testes de validação em *hardware* e *software*, fica claro com os resultados a qualidade e confiabilidade dos algoritmos de Perfil II no processo de criptografia de imagens. Essa comprovação é de grande valia para o desenvolvimento futuro de sistemas de segurança em *hardware*, em especial aqueles com limitação de recursos e necessidade de uma vazão de dados consideravelmente alta. A aplicação de sistemas seguros para transmissão de imagens, seja de maneira estática (foto) ou dinâmica (vídeo), em tempo real ou não, é hoje área de grande interesse no meio acadêmico e também cada vez mais presente na sociedade. Esse trabalho contribui para a comunidade acadêmica fornecendo uma base empírica e estocástica para futuros trabalhos utilizando os algoritmos de Perfil II do projeto eSTREAM ao mostrar a qualidade dos mesmos.

Ambos os sistemas são seguros para o processo de criptografia de imagens, porém cada um possui seus pontos fortes e fracos. Observou-se durante a implementação um nível maior de facilidade no desenvolvimento do sistema Trivium, que é também o que proporciona maior *throughput*. Por outro lado o sistema com menor número de elementos lógicos foi o Grain. Um esquemático que ilustra esses comparativos é mostrado na Figura 6.1.

Figura 6.1 – Comparativo entre facilidade, número de elementos e vazão.

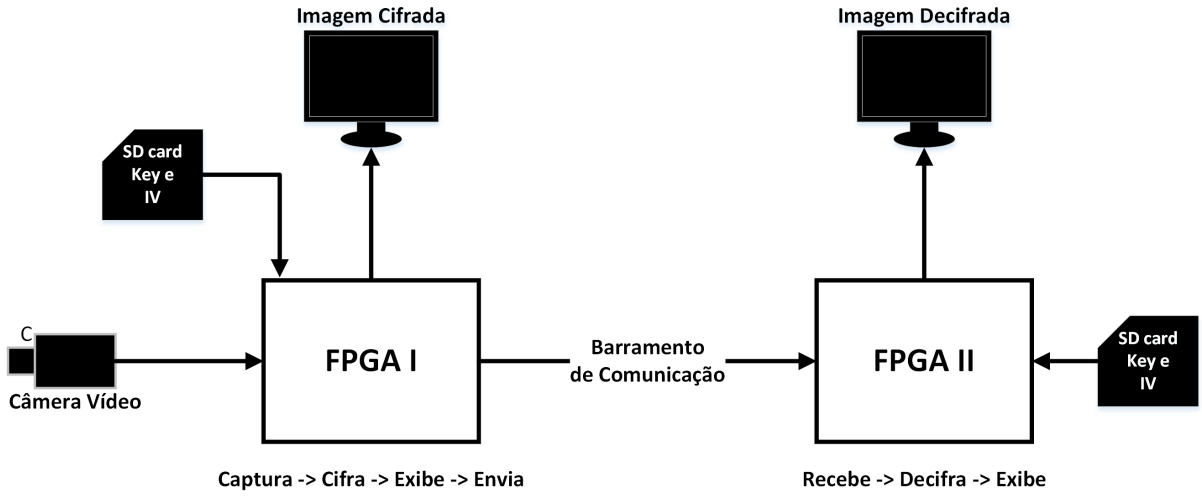


Fonte: Do autor (2017)

Como trabalho futuro, fica a integração e validação dos sistemas descritos em VHDL com uma câmera também instalada diretamente no FPGA, como mostrado na Figura 6.2. Nesse estágio serão utilizados dois FPGAs, o primeiro será responsável por capturar a imagem (foto ou vídeo) e cifrá-la utilizando um dos algoritmos. Após o processo de cifragem, a imagem resultante será exibida no monitor VGA conectado à essa primeira placa e então, utilizando alguma interface e padrão de comunicação, enviada para a segunda placa. O segundo FPGA

por sua vez, receberá o fluxo de *bits* que compõem a imagem cifrada e iniciará o processo de decifragem desses dados, por fim, a imagem decifrada será mostrada no monitor dessa placa. A configuração de chave e I.V. será feita utilizando o módulo de cartão SD disponível na plataforma DE2. Com isso espera-se aplicar com sucesso o processo de transmissão segura de imagens através de um meio inseguro, entre dois componentes de *hardware*.

Figura 6.2 – Esquemático de versão futura do sistema.



Fonte: Do autor (2017)

Outra melhoria possível nesse trabalho é a implementação e análise das versões dos sistemas de criptografia utilizando paralelização, para assim conseguir um maior *throughput*, ideal para aplicações com grande fluxo de dados como tratadas aqui. Por último, fica como trabalho futuro também a descrição e estudo de eficiência de novas interfaces de comunicação entre FPGA/PC ou FPGA/FPGA, como por exemplo Ethernet.

REFERÊNCIAS

ÅGREN, M. et al. Grain-128a: a new version of grain-128 with optional authentication. **International Journal of Wireless and Mobile Computing**, Inderscience Publishers, v. 5, n. 1, p. 48, 2011. Disponível em: <http://dx.doi.org/10.1504/IJWMC.2011.044106>.

Altera Corporation. **DE2 Development and Education Board: User Manual**. 1.4. ed. [s.n.], 2006. 72 p. Disponível em: <ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf>. Acesso em: 17 out. 2016.

BABBAGE, S.; DODD, M. The stream cipher mickey 2.0. **ECRYPT Stream Cipher**, 2006. Disponível em: <http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf>. Acesso em: 17 out. 2016.

BOTROS, N. **HDL Programming Fundamentals: VHDL and Verilog**. [S.l.]: Da Vinci Engineering Press, 2006. 506 p. ISBN 1584508558.

BOVIK, A. C. **Handbook of image and video processing**. 2. ed. Amsterdam; Boston, MA: Elsevier Academic Press, 2005. ISSN 9780080533612 0080533612 9780121197926 0121197921. ISBN 9780121197926.

CANNIÈRE, C. D. Trivium: A stream cipher construction inspired by block cipher design principles. In: **Lecture Notes in Computer Science**. Springer Science Business Media, 2006. p. 171–186. Disponível em: <http://dx.doi.org/10.1007/11836810_13>. Acesso em: 17 out. 2016.

CANNIÈRE, C. D.; PRENEEL, B. **Trivium specifications**. [S.l.], 2007. Disponível em: <http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf>. Acesso em: 17 out. 2016.

CASILLO, L. A. **Projeto e implementação em FPGA de um processador com conjunto de instrução reconfigurável utilizando VHDL**. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Norte, Natal, mai 2005. Disponível em: <http://repositorio.ufrn.br/handle/123456789/18071>. Acesso em: 17 out. 2016.

CHEN, G.; MAO, Y.; CHUI, C. K. A symmetric image encryption scheme based on 3D chaotic cat maps. **Chaos, Solitons & Fractals**, Elsevier BV, v. 21, n. 3, p. 749–761, jul 2004. Disponível em: <http://dx.doi.org/10.1016/j.chaos.2003.12.022>. Acesso em: 17 out. 2016.

CHIN, D.; BUER, M.; LUO, R. **Methods and apparatus for initialization vector pressing**. [S.l.]: Google Patents, 2011. US Patent 7,961,882.

CID, C.; ROBSHAW, M. **The eSTREAM Portfolio in 2012: ECRYPT II report D.SYM.10**. ECRYPT II, 2012. Disponível em: <http://www.ecrypt.eu.org/ecrypt2/documents/D.SYM.10-v1.pdf>. Acesso em: 17 out. 2016.

DINUR, I.; SHAMIR, A. Breaking grain-128 with dynamic cube attacks. In: **Fast Software Encryption**. Springer Science Business Media, 2011. p. 167–187. Disponível em: <http://dx.doi.org/10.1007/978-3-642-21702-9_10>. Acesso em: 17 out. 2016.

ECRYPT II. **eSTREAM Optimized Code HOWTO**. 2012. Acesso em: 2016-10-06. Disponível em: <http://www.ecrypt.eu.org/stream/perf/#results>. Acesso em: 17 out. 2016.

ECRYPT II. **eSTREAM: the ECRYPT Stream Cipher Project**. 2012. Disponível em: <http://www.ecrypt.eu.org/stream/>. Acesso em: 17 out. 2016.

ECRYPT II. **Grain v1**. 2012. Acesso em: 2016-10-06. Disponível em: <http://www.ecrypt.eu.org/stream/e2-grain.html>. Acesso em: 17 out. 2016.

eSCARGOT. **European Stream Ciphers Are Ready to GO**. 2008. <http://www.sheffield.ac.uk/eee/escargot>. Acesso em: 17 out. 2016.

GIERLICHS, B. et al. Susceptibility of estream candidates towards side channel analysis. **Proceedings of SASC**, Citeseer, p. 123–150, 2008. Disponível em: <https://securewww.esat.kuleuven.be/cosic/publications/article-1030.pdf>. Acesso em: 17 out. 2016.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 2. ed. [S.l.]: Prentice-Hall, Inc., 2002. ISBN 0201180758.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital De Imagens**. 3. ed. [S.l.]: Pearson Education, 2009. ISBN 9788576054016.

GOOD, T.; BENAISSA, M. Hardware results for selected stream cipher candidates. **State of the Art of Stream Ciphers**, p. 191–204, 2007. Disponível em: <http://www.ecrypt.eu.org/stream/papersdir/2007/023.pdf>. Acesso em: 17 out. 2016.

HAI, C. et al. Design and Realization of Image Encryption System Based on Compound Logistic Chaotic System. In: **2015 Third International Conference on Robot, Vision and Signal Processing (RVSP)**. IEEE, 2015. p. 75–77. ISBN 978-1-4673-9647-9. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7399151>. Acesso em: 17 out. 2016.

HELL, M. et al. A stream cipher proposal: Grain-128. In: **Proc. IEEE Int. Symp. Information Theory**. [s.n.], 2006. p. 1614–1618. ISSN 2157-8095. Disponível em: <dx.doi.org/10.1109/ISIT.2006.261549>. Acesso em: 17 out. 2016.

HELL, M.; JOHANSSON, T.; MEIER, W. Grain: a stream cipher for constrained environments. **International Journal of Wireless and Mobile Computing**, Inderscience Publishers, v. 2, n. 1, p. 86, 2007. Disponível em: <http://dx.doi.org/10.1504/IJWMC.2007.013798>. Acesso em: 17 out. 2016.

IPVM. **IPVM Video Surveillance Information**. 2017. Disponível em: <https://ipvm.com/>.

JAMES, F. A review of pseudorandom number generators. **Computer Physics Communications**, Elsevier, v. 60, n. 3, p. 329–344, out. 1990.

KITSOS, P. et al. FPGA-based performance analysis of stream ciphers ZUC, snow3g, grain v1, mickey v2, trivium and e0. **Microprocessors and Microsystems**, Elsevier BV, v. 37, n. 2, p. 235–245, mar 2013. Disponível em: <http://dx.doi.org/10.1016/j.micpro.2012.09.007>. Acesso em: 17 out. 2016.

KUON, I.; TESSIER, R.; ROSE, J. FPGA architecture: Survey and challenges. **FNT in Electronic Design Automation**, Now Publishers, v. 2, n. 2, p. 135–253, 2007. Disponível em: <http://dx.doi.org/10.1561/1000000005>. Acesso em: 17 out. 2016.

MANSINGKA, A. S. et al. Hardware stream cipher with controllable chaos generator for colour image encryption. **IET Image Processing**, v. 8, n. 1, p. 33–43, jan 2014. ISSN 1751-9659. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6704486>. Acesso em: 17 out. 2016.

Mathworks. **R2015a Release Highlights**. 2015. Disponível em: <https://www.mathworks.com/products/new_products/release2015a.html>.

MCKAY, K. A. et al. **NISTIR 8114 Report on Lightweight Cryptography**. [S.l.], 2017.

MENEZES, A. J. **Handbook of Applied Cryptography**. [S.l.]: CRC PR INC, 1996. ISBN 0849385237.

MERAH, L.; ALI-PACHA, A.; HADJ-SAID, N. Real-time cryptosystem based on synchronized chaotic systems. **Nonlinear Dynamics**, v. 82, n. 1-2, p. 877–890, oct 2015. ISSN 0924-090X. Disponível em: <http://dx.doi.org/10.1007/s11071-015-2202-2>. Acesso em: 17 out. 2016.

National Instruments. **Fundamentos da tecnologia FPGA**. 2013. Disponível em: <http://www.ni.com/white-paper/6983/pt/>. Acesso em: 17 out. 2016.

NYDIA, P. C. B. **Comparación de la Eficiencia en Hardware de los Cifradores de Flujo GRAIN, MICKEY-128 y TRIVIUM de ECRYPT.** 152 f p. Dissertação (Dissertação (mestrado)) — Instituto Nacional de Astrofísica, Óptica y Electrónica, 2008. Dissertação (Maestro en Ciencias Computacionales). Disponível em: <https://inaoe.repositorioinstitucional.mx/jspui/bitstream/1009/550/1/PerezCBN.pdf>.

PAAR, C.; PELZL, J. **Understanding Cryptography**. Springer Science Business Media, 2010. Disponível em: <http://dx.doi.org/10.1007/978-3-642-04101-3>.

PEDRONI, V. A. **Circuit Design with VHDL**. [S.l.]: MIT Press, 2004. 363 p. ISBN 0262162245.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. 2. ed. Novo Hamburgo - Rio Grande do Sul - Brasil: Editora Feevale, 2013. Disponível em: <https://www.feevale.br/cultura/editora-feevale/metodologia-do-trabalho-cientifico---2-edicao>.

RAJAGOPALA, S. et al. Dual Cellular Automata on FPGA: An Image Encryptors Chip. **Research Journal of Information Technology**, Academic Journals, v. 6, n. 3, p. 223–236, mar 2014. ISSN 1815-7432. Disponível em: <http://www.scialert.net/abstract/?doi=rjit.2014.223.236>. Acesso em: 17 out. 2016.

RAMIREZ-TORRES, M. T.; MURGUIA, J. S.; MEJIA-CARLOS, M. FPGA implementation of a reconfigurable image encryption system. In: **2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)**. IEEE, 2014. p. 1–4. ISBN 978-1-4799-5944-0. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7032524>. Acesso em: 17 out. 2016.

RIJMEN, V. Stream ciphers and the estream project. **The ISC Int'l Journal of Information Security**, v. 2, n. 1, p. 3–11, jan 2010.

ROBSHAW, M. The eSTREAM project. In: **Lecture Notes in Computer Science**. Springer Science Business Media, 2008. p. 1–6. Disponível em: <http://dx.doi.org/10.1007/978-3-540-68351-3_1>.

ROHITH, S.; BHAT, K. N. H.; SHARMA, A. N. Image encryption and decryption using chaotic key sequence generated by sequence of logistic map and sequence of states of linear feedback shift register. In: **2014 International Conference on Advances in Electronics Computers and Communications**. Institute of Electrical & Electronics Engineers (IEEE), 2014. Disponível em: <http://dx.doi.org/10.1109/ICAECC.2014.7002404>.

RUKHIN, A. et al. **A statistical test suite for random and pseudorandom number generators for cryptographic applications**. [S.l.], 2001.

SHAH, S. S. H.; RAJA, G. FPGA implementation of chaotic based AES image encryption algorithm. In: **2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)**. Department of Electrical Engineering, University of Engineering and Technology, Taxila, Pakistan: IEEE, 2015. p. 574–577. ISBN 978-1-4799-8996-6. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7412256>. Acesso em: 17 out. 2016.

SHRUTHI, K. M.; SHEELA, S.; SATHYANARAYANA, S. V. Image encryption scheme with key sequences based on chaotic functions. In: **2014 International Conference on Contemporary Computing and Informatics (IC3I)**. Institute of Electrical & Electronics Engineers (IEEE), 2014. Disponível em: <http://dx.doi.org/10.1109/IC3I.2014.7019667>.

SINGH, S. **O livro dos códigos**. 9. ed. Rio de Janeiro: Record, 2011. 448 p. ISBN 9788501055989.

SOLOMON, T. B. C. **Fundamentals of Digital Image Processing**. John Wiley & Sons, 2011. Disponível em: <http://www.ebook.de/de/product/18729473/chris_solomon_toby_breckon_fundamentals_of_digital_image_processing.html>.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0136097049, 9780136097044.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 7th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2016. 768 p. ISBN 9780134444284.

STOYANOV, B.; KORDOV, K. Novel image encryption scheme based on chebyshev polynomial and duffing map. **The Scientific World Journal**, Hindawi Publishing Corporation, v. 2014, p. 1–11, 2014.

TERADA, R. **Segurança de dados: criptografia em redes de computador**. 2. ed. São Paulo: Blucher, 2008. 305 p. ISBN 9788521204398.

University of Southern California. **SIPI Image Database**. 2017. Disponível em: <http://sipi.usc.edu/database/database.php>.

WARD, R.; MOLTENO, T. **Table of linear feedback shift register**. 2007. Disponível em: <http://courses.cse.tamu.edu/csce680/walker/lfsr_table.pdf>. Acesso em: 17 out. 2016.

WU, Y.; NOONAN, J. P.; AGAIAN, S. Npcr and uaci randomness tests for image encryption. **Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT)**, p. 31–38, 2011.