MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

RENATO HIDAKA TORRES

DESENVOLVIMENTO E ANÁLISE DE FUNÇÕES CRIPTOGRÁFICAS PARA OTIMIZAÇÃO DOS PADRÕES DE DISPERSÃO EM CRIPTOGRAMAS

Rio de Janeiro
2011

INSTITUTO MILITAR DE ENGENHARIA

RENATO HIDAKA TORRES

DESENVOLVIMENTO E ANÁLISE DE FUNÇÕES CRIPTOGRÁFICAS PARA OTIMIZAÇÃO DOS PADRÕES DE DISPERSÃO EM CRIPTOGRAMAS

Dissertação de Mestrado apresentada ao Curso de Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Orientador - José Antônio Moreira Xexéo - D. Sc,

Rio de Janeiro
2011

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

Renato Hidaka Torres.
005.82 Desenvolvimento e análise de funções criptográficas
T689d para otimização dos padrões de dispersão em criptogramas/ Renato Hidaka Torres, orientador José Antônio Moreira Xexéo.
– Rio de Janeiro: Instituto Militar de Engenharia, 2011.
116 p.: il.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2011.

1. Sistemas e computação - teses, dissertações 2. Criptografia 3.Transmissão de Assinaturas. I. Xexéo, J. A. M. II. Título. III. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

RENATO HIDAKA TORRES

DESENVOLVIMENTO E ANÁLISE DE FUNÇÕES CRIPTOGRÁFICAS PARA OTIMIZAÇÃO DOS PADRÕES DE DISPERSÃO EM CRIPTOGRAMAS

Dissertação de Mestrado apresentada ao Curso de Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Orientador - José Antônio Moreira Xexéo - D. Sc,

Aprovada em 11 de Novembro de 2011 pela seguinte Banca Examinadora:

Prof. Orientador - José Antônio Moreira Xexéo - D. Sc, do IME - Presidente

---

Prof. Paulo Afonso Lopes da Silva - Ph.D, do IME

Prof. Claudio Gomes de Mello - D. Sc, do IME

Prof. Geraldo Bonorino Xexéo - D. Sc, da COPPE/UFRJ

Rio de Janeiro
2011

Dedico esta dissertação aos meus pais.

AGRADECIMENTOS

Agradeço a todas as pessoas que contribuíram com o desenvolvimento desta dissertação, em especial ao meu orientador José A. M. Xexéo, pela confiança, competência e orientação objetiva. Ao professor Paulo Afonso Lopes, pelas reuniões e conselhos em momentos crucias do desenvolvimento desta dissertação. Ao IME e a CAPES pela oportunidade e financiamento deste trabalho. Aos meus pais Sérgio e Ilda, por toda a educação e carinho oferecidas ao longo de minha jornada. A minha namorada Júlia Oliveira, pela confiança, carinho e compreensão nos momentos em que estive ausente. A todos os colegas de mestrado, em especial ao colega Gláucio Oliveira pelos momentos de pesquisa vividos durante essa jornada. A todos os professores e funcionários do Departamento de Engenharia de Sistemas (SE/8) do Instituto Militar de Engenharia. E principalmente ao mestre dos mestres, Deus.

Renato Hidaka Torres

Para um bom sistema criptográfico passos devem ser realizados ou difundir ou confundir a redundância (ou ambos).

Claude Elwood Shannon

RESUMO

Este trabalho apresenta duas novas funções para algoritmos criptográficos que tem como finalidade impedir a transmissão de assinaturas. A proposta dessas novas funções surgiram mediante a análise das vulnerabilidades do algoritmo AES perante ao ataque teórico *Differential Fault Analysis*. Para a análise das funções, estas foram acopladas ao algoritmo AES e então foram geradas amostras de criptogramas a fim de verificar a transmissão de assinaturas existentes nos textos em claro. O algoritmo utilizado para a identificação de assinaturas também foi desenvolvido neste trabalho, assim como duas das quatro métricas de avaliação utilizadas no processo de avaliação da qualidade de dispersão das assinaturas. Com os resultados foi possível observar que o algoritmo AES, após as modificações realizadas, apresentou resultados satisfatórios em relação a dispersão das assinaturas existentes nos textos em claro, mesmo utilizando o modo de operação ECB.

ABSTRACT

This work presents two new functions for cryptographic algorithms that have as objective prevent the signatures transmission. The proposal these new functions have emerged through of analysis of vulnerabilities of AES algorithm against the teorical attack Differential Fault Analysis. For the analysis of functions, these were coupled to the AES algorithm and then cryptograms sample were generated with objective of check the signatures transmission existing in clear texts. The algorithm used to verify to the identification of signatures also were developed in this work, as well as two of the four evaluation metrics used in process of assessing the quality of dispersion of signatures. With the results we observed that the AES algorithm, after the changes made, satisfactory results were presented regarding the dispersion of existing signatures in clear text, even using the ECB operating mode.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABREVIATURAS

AES	-	*Advanced Encryption Standard*
CBC	-	*Cipher Block Chaining*
DES	-	*Data Encryption Standard*
DFA	-	*Differential Fault Analysis*
ECB	-	*Eletronic Codebook*
GSI/IME	-	*Grupo de Segurança da Informação do Instituto Militar de Engenharia*
NIST	-	*National Institute of Standards and Technology*
SAC	-	*Strict Avalanche Criterion*

1 INTRODUÇÃO

A identificação de assinaturas em criptogramas é uma área em desenvolvimento. Seu sucesso possibilita a execução do primeiro passo para um eventual ataque aos criptogramas, que condiz com a identificação do algoritmo utilizado no processo de cifragem. Nesse sentido, os estudos de reconhecimento de padrões começaram a apresentar resultados significativos em relação a identificação de assinaturas existentes em criptogramas. Em relação aos estudos do grupo de segurança da informação do Instituto Militar de Engenharia (GSI/IME), (CARVALHO, 2006) foi o primeiro trabalho a explorar essa linha de pesquisa sendo sua principal contribuição, a consolidação da fase de pré-processamento dos criptogramas. Em seguida (SOUZA, 2007) deu continuidade aos estudos dando sua parcela de contribuição na análise de várias medidas de distância e similaridade. Além disso, contribuiu também com o desenvolvimento de um novo algoritmo de reconhecimento de padrões utilizando redes neurais. (OLIVEIRA, 2011) foi o último trabalho anterior a este, e também apresentou resultados importantes em relação ao reconhecimento de algoritmos criptográficos, sendo suas principais contribuições o desenvolvimento de um novo algoritmo de reconhecimento de padrões utilizando algoritmos genéticos, e as melhorias nos resultados em comparação aos resultados obtidos com o algoritmo de redes neurais desenvolvido por (SOUZA, 2007).

Alguns trabalhos desenvolvidos fora do GSI/IME também apresentaram suas contribuições nessa linha de pesquisa. (KNUDSEN, 2001) realizaram o chamado "ataque de distinção", para criptogramas gerados pelo algoritmo RC6 com o auxílio da estatística do qui-quadrado. (DILEEP, 2006) usaram Máquinas de Vetor de Suporte (SVM) para identificar as cifras de bloco DES, Triple DES, Blowfish, AES e RC5, a partir de conjuntos de criptogramas gerados por eles. (UEDA, 2007) propuseram um método para fortalecer o algoritmo RC6, como uma contramedida ao ataque com a estatística do qui-quadrado. (NAGIREDDY, 2008) desenvolveu métodos de histograma e de predição de bloco, técnicas de expansão de dados e técnicas baseadas em ataques secundários para a identificação das cifras de bloco DES, AES, Blowfish, Triple DES e RC5. Os resultados desses trabalhos reforçou a hipótese da existência de "assinaturas" transmitidas para os criptogramas pelas chaves e pelos próprios algoritmos de cifragem.

Nesse contexto, o presente trabalho tem como objetivo analisar e propor modificações no algoritmo AES a fim de impedir a transmissão de assinaturas para as cifras. Para análise do algoritmo foi utilizado como ferramenta o ataque teórico *Differential Fault Analysis*(DFA) proposto por (GIRAUD, 2004). Feita a análise, foi possível perceber que os principais pontos que podem comprometer o algoritmo AES são as funções de substituição, keySchedule e também a ausência da função Mixcolumns na última rodada do algoritmo.

Sendo assim, no capítulo 5 será apresentada uma nova proposta de construção de caixas-S baseada no conceito de caminhos aleatórios e dependente da chave secreta. Após a modelagem da caixa-S, foi necessário realizar a avaliação da mesma no que se refere a segurança e ao grau de incerteza dos dados produzidos. Para isso, foram utilizados um conjuntos de métricas de avaliação conforme descritas no capítulo 3.

Em relação keySchedule, foi realizada uma análise utilizando o método de classificação de funções de expansão de chaves proposto por (CARTER, 1999) e a partir dessa análise foi possível observar que a função keySchedule do algoritmo AES permite o conhecimento da chave secreta a partir de suas subchaves, constatando dessa forma o sucesso do teórico ataque DFA. Para corrigir esse ponto fraco do algoritmo, o presente trabalho propôs modificações na função de expansão de chave do algoritmo AES, de tal forma que após as modificações, a nova função fosse classificada pelo método proposto por (CARTER, 1999), como uma função que a partir de uma subchave, fosse impossível ou muito difícil o conhecimento da chave secreta.

Após essas modificações, foram realizados experimentos com o objetivo de verificar se as cifras produzidas pelo algoritmo AES modificado ainda apresentavam as assinaturas existentes nos textos em claro. Para a execução do experimento foi utilizando o algoritmo de reconhecimento de padrões em criptogramas que também foi desenvolvido neste trabalho. O novo algoritmo é baseado em teoria dos grafos e sua principal vantagem em relação aos demais algoritmos de reconhecimento de padrões desenvolvidos nos trabalhos de (CARVALHO, 2006), (SOUZA, 2007) e (OLIVEIRA, 2011), é o tempo de processamento necessário para a presentar os mesmos resultados em relação a qualidade dos grupos dos criptogramas assinados.

O resultado dos experimentos demonstrou que as cifras provenientes do algoritmo AES modificado apresentaram um nível de dispersão satisfatório, ou seja, o número de grupos foi muito maior do que os apresentados pelos cinco algoritmos finalistas do concurso

do AES. Essa característica indica que o algoritmo conseguiu dispersar as assinaturas existentes nos textos em claro mesmo que o modo de operação utilizado pelo algoritmo fosse o modo ECB.

Para avaliar os resultados dos experimentos, foram utilizadas quatro métricas de avaliação: revocação, precisão, grau de conectividade dos documentos e entropia amostral. Sendo as duas últimas, duas novas métricas que estão sendo propostas neste trabalho. O grau de conectividade dos documentos, tem como objetivo ser uma métrica de avaliação complementar as métricas de precisão e revocação. Já a métrica de entropia amostral tem por finalidade avaliar o grau de incerteza dos criptogramas levando em consideração a distribuição dos pesos de hamming da amostra.

1.1 MOTIVAÇÃO

A principal motivação para o desenvolvimento deste trabalho está na possibilidade de propor melhorias no algoritmo AES, a fim garantir maior segurança aos criptogramas independentemente do modo de operação utilizado no processo de cifragem.

Os modos de operações mais utilizados pelos algoritmos de cifra de bloco são o *Electronic CodeBook* (ECB) e o *Cipher-block chaining* (CBC), entretanto, os algoritmos que operam no modo ECB geram blocos criptografados idênticos para textos em claro idênticos, sendo esta características um dos grandes questionamentos na utilização desse modo de operação. Para suprir esta característica, o modo CBC e outros modos existentes, utilizam funções que alteram o dado de entrada do algoritmo para que sejam produzidos diferentes blocos de cifras para o mesmo texto em claro.

Entretanto, estes procedimentos não alteram a estrutura do algoritmo, logo a segurança das cifras fica subordinada ao modo de operação. Para suprir estas dependências, o ideal é alterar a estrutura do algoritmo, fazendo com que as "assinaturas" não fossem transmitidas independentemente do modo de operação utilizado, sendo este o propósito deste trabalho.

1.2 CONTRIBUIÇÕES

Para alcançar o objetivo proposto, este trabalho apresenta as seguintes contribuições:

a) Constatação da afirmação de (DUNKELMAN, 2010) de que para certos ataques aplicados sobre o AES, a ausência da transformação *MixColumns* na última rodada

pode diminuir a complexidade do ataque. Esse fato foi observado a partir da análise realizada no capítulo 3.

b) Desenvolvimento de uma nova metodologia para construção de caixas-S apresentada no capítulo 5.

c) Desenvolvimento de melhorias na função de expansão de chave do algoritmo AES apresentada no capítulo 6.

d) Reclassificação da função de expansão de chave do algoritmo AES após as modificações realizadas, levando em consideração os critérios de classificação proposto por (CARTER, 1999).

e) Desenvolvimento de um novo algoritmo e reconhecimento de padrões em criptogramas conforme apresentado no capítulo 7.

f) Resistência do algoritmo AES modificado, ao ataque teórico DFA proposto por (GIRAUD, 2004).

g) Proposta da métrica grau de conectividade dos documentos conforme apresentada no capítulo 7.

h) Proposta da métrica entropia amostral também apresentada no capítulo 7.

i) Dispersão dos padrões dos criptogramas cifrados pelo algoritmo AES após realizadas as modificações.

j) Contestação da afirmação de (DWORKIN, 2001) de que dois blocos idênticos cifrados ou decifradas com a mesma chave utilizando o modo ECB geram blocos idênticos como resultado do processo.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

O capítulo 2 apresenta a contextualização do presente trabalho em relação aos trabalhos que abordam a análise e desenvolvimento de novos algoritmos ou funções criptográficas.

No capítulo 3 são apresentados os conceitos necessários para o entendimento do processo de avaliação de funções booleanas, bem como as métricas utilizadas para a avaliação das funções que comporão a caixa-S desenvolvida neste trabalho.

O capítulo 4 a presenta o processo de avaliação realizado no algoritmo AES, a fim de encontrar fraquezas e possíveis causas para a geração de padrões, conforme apresentadas nos trabalhos de (CARVALHO, 2006); (DILEEP, 2006); (SOUZA, 2007); (SOUZA, 2008); (NAGIREDDY, 2008); (TORRES, 2010) e (OLIVEIRA, 2011). Utilizando para isso, o ataque *Differential Fault Analysis* (DFA) proposto por (GIRAUD, 2004) como ferramenta de análise .

O capítulo 5 descreve um novo modelo para a construção de caixa-S baseadas em caminhos aleatórios. Para demonstrar a eficiência do modelo, foi realizada a construção de uma caixa-S a qual foi submetida à avaliação das métricas apresentadas no capítulo 3. Com os resultados da avaliação, foi possível perceber o bom nível de aleatoriedade e não linearidade da caixa-S, assim como a maior resistência ao ataque DFA proposto por (GIRAUD, 2004).

O capítulo 6 descreve uma nova função de expansão de chave para o algoritmo AES, com o objetivo de classificá-lo como um algoritmo mais seguro levando em consideração os critérios apresentados por (CARTER, 1999). Para verificar a eficiência da nova função, foi realizada uma análise comparativa com as cinco funções de expansão de chave dos algoritmos finalistas do concurso do AES, levando em consideração critérios como quantidade de subchaves com peso de hamming balanceado; propagação ou correção de erros quando as subchaves fossem concatenadas; aleatoriedade das chaves quando submetidas ao testes do NIST e à verificação do SAC.

No capítulo 7 são apresentados os resultados da avaliação do algoritmo AES modificado no que diz respeito à dispersão de padrões dos criptogramas. Foram realizados experimentos em que foi observado a transmissão dos padrões do texto claro para as respectivas cifras dos cinco algoritmos finalistas do concurso do AES, e ao mesmo tempo também foi observado que para o mesmo conjunto textos em claro, as cifras correspondentes ao algoritmo AES modificado diminuíram os padrões encontrados. Para realizar o agrupamento foi elaborado um novo algoritmo baseado em teoria dos grafos que tem como objetivo agrupar os criptogramas pertencentes a subgrafos conexos. E para a avaliação dos grupos foram utilizadas as métricas de revocação, precisão, grau de conectividade dos vértices e entropia da amostra.

No capítulo 8 são feitas as conclusões e apresentadas as possibilidades de trabalhos futuros.

2 CONTEXTUALIZAÇÃO DO TRABALHO

Esse capítulo tem como objetivo contextualizar o presente trabalho em relação aos trabalhos que abordam a análise e desenvolvimento de novos algoritmos ou funções criptográficas.

2.1 ESTUDO DE CAIXAS-S

O estudo de caixas-S tornou-se uma área muito importante da criptoanálise, pois esta função em muitos casos é a única transformação responsável pela confusão dos bites na maioria dos algoritmos de criptografia simétrica. Pelo conhecimento que se tem, o primeiro algoritmo que utilizou essa metodologia de confusão de bites foi o algoritmo LUCIFER eleito em 1977 o padrão DES com algumas modificações, versão mais atual FIPS 46-3 (NIST, 1999). Todavia, os primeiros algoritmos de substituição monoalfabético e posteriormente os polialfabéticos, de certa forma também utilizaram essa metodologia ainda que de forma primaria, como por exemplo a simples rotação de caracteres.

Publicamente, a constatação da importância de se utilizar caixas-S foi ressaltada durante o concurso do algoritmo AES em 1999, quando foi observado que todos os cinco algoritmos finalistas faziam uso destas transformações. Desde então, muitos estudos e ataques foram concentrados nessa linha de pesquisa. Trabalhos como os de (LIU, 2005), (KUDOU, 2009), (ZAÏBI, 2009), (TRAN, 2008), (KAZMI, 2009), (CHEN, 2004) e (BURNETT, 2005), são exemplos de pesquisas de avaliação e desenvolvimento de novas caixas-S.

Em especial, os trabalhos de (KAZMI, 2009) e (BURNETT, 2005) podem ser destacados pois são bem próximos em relação a metodologia de caixas-S apresentada neste trabalho. (KAZMI, 2009) utiliza a mesma teoria de caminhos aleatórios para a confecção de caixas-S, porém com uma metodologia e lógica totalmente diferente da apresentada neste trabalho. Já (BURNETT, 2005) se semelha pelo fato de também utilizar funções booleanas como núcleo de suas caixas-S.

Em relação aos ataques que exploram as fraquezas das funções das caixas-S, dentre os mais conhecidos está o ataque diferencial propostos por (BIHAM, 1991), o ataque linear proposto por (MATSUI, 1994).

2.2 ESTUDO DE FUNÇÕES DE EXPANSÃO DE CHAVES

Em relação as funções de expansão de chave, pelo conhecimento que se tem, elas também surgiram a partir do algoritmo LUCIFER e sua popularização também foi muito semelhante a das caixas-S. Entretanto, observando o contexto em que o algoritmo LUCIFER foi projetado, acredita-se que a função de expansão de chave surgiu por limitações computacionais, uma vez que para garantir a segurança da cifra é necessário a utilização de chaves com uma grande sequência de bites, e na época o recurso para armazenamento e processamento dessas chaves seria um tanto limitado. Logo, as funções de expansão de chave surgiram com o propósito de a partir de uma chave secreta pequena, gerar subchaves a cada rodada tal que no final, a concatenação das subchaves fosse equivalente a utilização de uma única chave de tamanho necessário para garantir a segurança das cifras.

Não obstante, os recursos computacionais disponíveis já permitem a utilização das chaves sem a necessidade do algoritmos de produção de subchaves. Porém, como observado na literatura, os atuais algoritmos, como por exemplo o próprio AES, ainda fazem uso dessas funções, logo, foi possível verificar que além da economia de recursos computacionais, as funções de expansão de chave agregam outros benefícios, como por exemplo o nível de aleatoriedade das chaves, ou seja, utilizando uma chave secreta considerada fraca para a cifragem de uma mensagem, a função de expansão de chave tem como objetivo conseguir a partir dessas chaves fracas, gerar subchaves com um bom nível de aleatoriedade.

Entretanto, pode-se observar que nem todas as funções de expansão de chave conseguem cumprir esse objetivo. A função utilizada pelo algoritmo *Tiny Encryption Algorithm* (TEA) proposta por (WHEELER, 1994), é um exemplo de função que não consegue gerar subchaves aleatórias a partir de chaves fracas. Nesse sentido, algumas pesquisas vem sendo realizadas com a finalidade de avaliar e identificar possíveis fraquezas e melhorias para os algoritmos de expansão de chaves. O trabalho de (CARTER, 1999) por exemplo, sugere critérios de avaliação para classificar os algoritmos finalistas do concurso do AES levando em consideração a segurança das funções de expansão de chaves.

Sendo assim, o presente trabalho teve como um de seus objetivos avaliar a função de expansão de chave do algoritmo AES e propor melhorias com o intuito de melhorar a classificação da função de acordo com os critérios propostos por (CARTER, 1999). Além disso, foram realizadas análises em todas as funções de expansão de chave dos cinco algoritmos finalistas do concurso do AES e os resultados encontram-se descritos no

capítulo 6.

2.3 MODOS DE OPERAÇÕES

No contexto da criptografia simétrica em bloco, os modos de operações são de fundamental importância. Dentre os vários modos de operações, os modos ECB e CBC são os mais populares. (DWORKIN, 2001), desenvolveu um relatório técnico para as devidas recomendações e especificações dos modos de operações mais utilizados pelas cifras de criptografia simétrica em bloco. No modo ECB a mensagem é divida em blocos e cada bloco é cifrado ou decifrado diretamente e independentemente de outros blocos. Por esse motivo o modo de operação ECB permite a paralelização do algoritmo que o utilize. Contudo, conforme observado por (DWORKIN, 2001), se dois blocos idênticos forem cifrados ou decifrados com a mesma chave utilizando o modo ECB, como resultado também seriam obtidos blocos idênticos.

Já para o modo CBC essa característica não é observada, pois nesse modo de operação o primeiro bloco de entrada do processo de cifragem é formado pela operação de ou-exclusivo entre o bloco em claro e um vetor de inicialização. E em seguida, em processo cascata cada bloco de saída é submetido a operação de ou-exclusivo com o próximo bloco em claro para gerar atual bloco a ser cifrado. Sendo assim, pode-se perceber que para o modo CBC no processo de cifragem a paralelização não pode ser realizada.

Dessa forma, como resultado dos trabalhos de reconhecimento de padrões, pode-se perceber que os algoritmos que utilizam o modo de operação ECB estão mais sujeitos às transmissões de padrões do que o modo de operação CBC por exemplo. Sendo assim, o presente trabalho está inserido dentro do contexto de modos de operação, uma vez que tem como objetivo desenvolver funções que quando acopladas a algum algoritmo criptográfico, consiga dispersar os padrões existentes nos textos em claro independentemente do modo de operação utilizado. Contestando dessa forma a observação feita por (DWORKIN, 2001) de que dois blocos idênticos cifrados ou decifradas com a mesma chave utilizando o modo ECB irão gerar blocos idênticos como resultado do processo.

2.4 CONCLUSÕES DO CAPÍTULO

A avaliação e desenvolvimento de novas funções e algoritmos criptográficos é um área importante para a segurança da informação. Manter a segurança e confidencialidade da

informação são algumas das grandes preocupações da criptologia. Nesse contexto, o presente trabalho está inserido no campo de estudo das avaliações de algoritmos criptográficos a fim de contribuir para o fortalecimento de seus componentes. Sabe-se que nem sempre os estudos realizados na área da criptologia tem como finalidade a divulgação do conhecimento adquirido, porém, como esta pesquisa faz parte do meio acadêmico, a prioridade é a total divulgação e colaboração para o fortalecimento da criptografia.

3 MÉTRICAS PARA AVALIAÇÃO DE FUNÇÕES BOOLEANAS

Este capítulo tem como objetivo apresentar os conceitos necessários para o entendimento do processo de avaliação de funções booleanas. A primeira seção deste capítulo descreverá o modelo de representação das funções bem como algumas definições. A segunda seção explanará as métricas utilizadas para a avaliação das funções. E, a última seção abordará a importância dessas métricas no que diz respeito à segurança de algoritmos criptográficos.

3.1 MODELO DE REPRESENTAÇÃO DAS FUNÇÕES

Seja $V_2^n = GF(2^n)$ um vetor binário de tamanho n. Uma função booleana $f(x)$ é mapeada de $f(x) : V_2^n \rightarrow V_2$ em que $x = (x_{n-1}, ...x_0)$. A representação destas funções geralmente é realizada por tabela verdade, tabela verdade polar ou forma normal algébrica. A tabela verdade é modelo clássico para representação de funções booleanas e por este motivo esta será a representação adotada neste trabalho.

3.1.1 TABELA VERDADE

Uma função $f(x)$, sendo x de tamanho n, pode obter 2^n diferentes vetores de entrada. Como exemplo, considere $n = 3$, dessa forma o número de possíveis entradas é igual a 8. A TAB 3.1 ilustra o exemplo.

TAB. 3.1: Possíveis entradas para n = 3

x_2	x_1	x_0	f(x)
0	0	0	$\{0,1\}$
0	0	1	$\{0,1\}$
0	1	0	$\{0,1\}$
0	1	1	$\{0,1\}$
1	0	0	$\{0,1\}$
1	0	1	$\{0,1\}$
1	1	0	$\{0,1\}$
1	1	1	$\{0,1\}$

Sendo assim, é possível perceber que o número de diferentes funções booleanas de n variáveis que podem ser construídas é igual a 2^{2^n}. Para $n = 3$, existem 256 possibilidades.

3.1.2 PESO DE HAMMING

Seja $hw(f)$ o peso de hamming de uma função booleana $f(x)$ de n variáveis. Então $hw(f) = \sum_{x=0}^{2^n-1} f(x)$, que representa o número de bites uns na tabela verdade da função.

3.1.3 DISTÂNCIA ENTRE DUAS FUNÇÕES

Seja $d(f,g)$ a distância entre as funções f e g de n variáveis, f(x) e g(x). Então $d(f,g) = hw(f \oplus g)$, ou seja, a distância entre as duas funções é calculada pelo número de bites que diferem nas correspondentes posições de suas tabelas verdades. A TAB 3.2 ilustra um exemplo para duas funções hipotéticas.

TAB. 3.2: Possíveis entradas para n = 3

x_2	x_1	x_0	f(x)	g(x)	$f \oplus g$
0	0	0	0	1	1
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	1	0
1	1	1	0	1	1
			hw(f) = 4	hw(g) = 5	d(f,g) = 3

3.1.4 FUNÇÕES BALANCEADAS

Uma função booleana $f(x)$ é dita balanceada quando a seguinte propriedade é estabelecida: $hw(f) = 2^{n-1}$. Sendo assim, o número de funções balanceadas com n variáveis pode ser definido conforme a equação 3.1

$$N_b = \frac{2^n!}{2^{n-1}! * 2^{n-1}!} \tag{3.1}$$

3.1.5 FUNÇÕES LINEARES

Seja a e x vetores $\in V_2^n$. Uma função booleana da forma $l_a(x) = a_1 \bullet x_1 \oplus a_2 \bullet x_2 \oplus ... \oplus a_n \bullet x_n$ é chamada de função linear de n variáveis sendo $\bullet$ o produto binário e $\oplus$ a soma módulo 2.

Dessa forma é possível perceber que para uma função de n variáveis, o número de funções lineares pode ser definido conforma a equação 3.2.

$$N_l = \sum_{i=1}^{n} \frac{n!}{i! * (n-i)!} \tag{3.2}$$

3.1.6 FUNÇÕES AFINS

Seja a e x vetores $\in V_2^n$ e $a_0 \in V_2$. Uma função booleana da forma $f(x) = a_0 \oplus l_a(x) = a_0 \oplus a_1 \bullet x_1 \oplus a_2 \bullet x_2 \oplus ... \oplus a_n \bullet x_n$ é chamada de função afim de n variáveis sendo $\bullet$ o produto binário e $\oplus$ a soma módulo 2.

Pode-se perceber que se $a_0 = 0$, então as funções são lineares. Nesse diapasão, o conjunto das funções afins é composta por todas as funções lineares e seus complementos. A FIG 3.1 ilustra a proporção de funções lineares e afins em relação ao número total de funções booleanas de n variáveis.

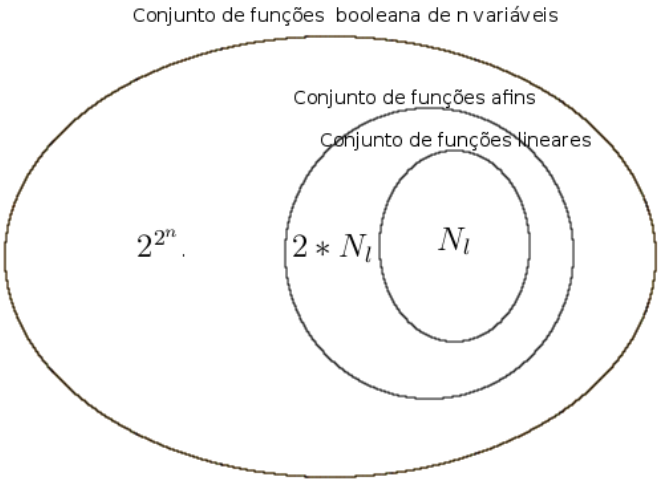

FIG. 3.1: Conjunto de funções com n variáveis

3.2 MÉTRICAS UTILIZADAS PARA A AVALIAÇÃO DAS FUNÇÕES

Nesta seção serão explanadas as métricas utilizadas para avaliação das funções booleanas utilizadas no processo de construção da caixa-S e expansão de chaves realizadas neste trabalho. Uma vez que as funções satisfaçam todos os critérios de avaliação, pode-se concluir que elas possuirão alta aleatoriedade e resistência contra ataques clássicos como

os ataques linear e diferencial, haja vista que essas métricas tem como objetivo avaliar justamente os pontos explorados por esses ataques.

A dificuldade na seleção dessas métricas foi que os trabalhos inicialmente pesquisados no que tange a novas propostas de caixas-S, não apresentavam métricas para a avaliação da função desenvolvida. Na maioria dos casos a análise era feita em cima da otimização do algoritmo. Não obstante, realizando uma pesquisa mais aprofundada foi possível encontrar trabalhos que realizassem suas avaliações de maneira satisfatória. Os trabalhos de (PIEPRZYK, 1988); (MEIER, 1990); (KAM, 1979); (KIM, 1990); (FEISTEL, 1973); (WEBSTER, 1986); (FULLER, 2003); (BURNETT, 2005); (MENEZES, 1996)e (MAR, 2008) foram utilizados como arcabouço para a seleção das métricas de avaliação utilizadas nesse trabalho.

A seguir será realizada a descrição de cada métrica:

3.2.1 NÃO LINEARIDADE

Esse métrica foi inicialmente proposta por (PIEPRZYK, 1988) e (MEIER, 1990). Seu objetivo é avaliar o quão distante está uma determinada função booleana em relação ao conjunto das funções afins, ou seja, avaliar a dissimilaridade da função em relação ao conjunto das funções afins. Para isso, a equação 3.3 pode ser utilizada.

$$N(f) = \min_{\alpha \in A_n} d(f, \alpha) \tag{3.3}$$

Tal que A_n representa o conjunto das funções afins e $N(f)$ a não linearidade da função f que é representada pelo menor peso de hamming em relação ao conjunto das funções afins.

Medir a não linearidade da função é importante pois uma função com alta não linearidade dá a garantia de que será menos provável obter uma representação para essa função de forma linear. Para criptografia essa característica é importante pois quanto maior a não linearidade de uma função, maior será a dificuldade de realizar o ataque de criptoanálise linear proposto por (MATSUI, 1993).

A TAB 3.3 ilustra um exemplo do calculo da não linearidade de uma função hipotética com $n = 3$. Para a geração da tabela foram encontradas todas as funções afins e calculada a distância de cada uma em relação a função $f(x)$, a menor distância encontrada é considerada a não linearidade da função $f(x)$.

TAB. 3.3: Exemplo de não linearidade para uma função hipotética sendo n = 3

	000	001	010	011	100	101	110	111	distância
c	0	0	0	0	0	0	0	0	4
a_0	0	1	0	1	0	1	0	1	4
a_1	0	0	1	1	0	0	1	1	4
a_2	0	0	0	0	1	1	1	1	2
$a_1 \oplus a_0$	0	1	1	0	0	1	1	0	4
$a_2 \oplus a_0$	0	1	0	1	1	0	1	0	6
$a_2 \oplus a_1$	0	0	1	1	1	1	0	0	2
$a_2 \oplus a_1 \oplus a_0$	0	1	1	0	1	0	0	1	2
$\overline{c}$	1	1	1	1	1	1	1	1	4
$\overline{a_0}$	1	0	1	0	1	0	1	0	4
$\overline{a_1}$	1	1	0	0	1	1	0	0	4
$\overline{a_2}$	1	1	1	1	0	0	0	0	6
$\overline{a_1 \oplus a_0}$	1	0	0	1	1	0	0	1	4
$\overline{a_2 \oplus a_0}$	1	0	1	0	0	1	0	1	2
$\overline{a_2 \oplus a_1}$	1	1	0	0	0	0	1	1	6
$\overline{a_2 \oplus a_1 \oplus a_0}$	1	0	0	1	0	1	1	0	6
f(x)	0	0	1	0	1	1	0	1	min = 2

3.2.2 COMPLETUDE

Essa métrica foi introduzida por (KAM, 1979) e tem como objetivo verificar o grau de dependência entre os bites de saída e os bites de entrada de uma dada função. Funções que possuem a completude, ou seja, funções que gerem bites de saída dependentes de todos os bites de entrada, são consideradas mais resistentes a ataques de texto em claro conhecido.

Uma função $f(x) : V_2^n \rightarrow V_2^n$ possui completude se para todo $i, j \in \{1, ..., n\}$, existe dois vetores de n bites X e X_i, tal que X e X_i diferem apenas na i-ésima posição e $f(X)$ difere de $f(X_i)$ ao menos na posição j.

Segundo (KIM, 1990), essa definição pode ser expressada da seguinte forma: Uma função $f(x) : V_2^n \rightarrow V_2^n$ é completa, se e somente se a equação 3.4 for verdadeira.

$$\sum_{x \in V_2^n} f(x) \oplus f(x \oplus c_i^{(n)}) > (0, 0, ..., 0) \qquad \forall 1 \leq i \leq n \tag{3.4}$$

Como exemplo, considere $y = f(x)$ sendo x e y vetores de tamanho $n = 3$. A função

$f(x)$ realiza as seguintes transformações:

$$
\begin{aligned}
y_0 &= x_0 \cup x_2 \\
y_1 &= (x_0 \oplus x_1) \oplus x_0 \\
y_2 &= (x_1 \wedge x_0) \oplus x_2
\end{aligned}
\tag{3.5}
$$

A TAB 3.4 ilustra o mapeamento da função, e como pode ser observado a função não é completa pois exitem combinações que não obedecem a equação 3.4, como por exemplo: $f(100) \oplus f(100 \oplus 1_0^{(3)}) = (0,0,0)$, ou seja, $f(100) = f(101)$.

TAB. 3.4: Exemplo de completude para uma função hipotética sendo n = 3

$x_2x_1x_0$	$f(x)$	$y_2y_1y_0$
000	$\rightarrow$	000
001	$\rightarrow$	001
010	$\rightarrow$	010
011	$\rightarrow$	001
100	$\rightarrow$	101
101	$\rightarrow$	101
110	$\rightarrow$	111
111	$\rightarrow$	001

3.2.3 EFEITO AVALANCHE

Essa métrica foi proposta por (FEISTEL, 1973) e tem como objetivo verificar se pequenas propagações dos bites de entrada geram como saída vetores balanceados. Se isso for constatado, há indícios de que a função geradora provoque alta dependência dos bites de saída em relação aos bites de entrada.

Segundo (KIM, 1990), uma função $f(x) : V_2^n \rightarrow V_2^n$ possui o efeito avalanche se e somente se a equação 3.6 for verdadeira.

$$
\sum_{x \in V_2^n} hw(f(x) \oplus f(x \oplus c_i^{(n)})) = n2^{n-1} \qquad \forall 1 \leq i \leq n \tag{3.6}
$$

Utilizando como exemplo a mesma função, verificou-se que o efeito avalanche também não é satisfeito. A TAB 3.5 ilustra o processo de avaliação para o exemplo.

TAB. 3.5: Exemplo de efeito avalanche para uma função hipotética sendo n = 3

c_0^3	c_1^3	c_2^3
$f(000) \oplus f(001) = 001$	$f(000) \oplus f(010) = 010$	$f(000) \oplus f(100) = 101$
$f(001) \oplus f(000) = 001$	$f(001) \oplus f(011) = 000$	$f(001) \oplus f(101) = 100$
$f(010) \oplus f(011) = 011$	$f(010) \oplus f(000) = 010$	$f(010) \oplus f(110) = 101$
$f(011) \oplus f(010) = 011$	$f(011) \oplus f(001) = 000$	$f(011) \oplus f(111) = 000$
$f(100) \oplus f(101) = 000$	$f(100) \oplus f(110) = 010$	$f(100) \oplus f(000) = 101$
$f(101) \oplus f(100) = 000$	$f(101) \oplus f(111) = 100$	$f(101) \oplus f(001) = 100$
$f(110) \oplus f(111) = 110$	$f(110) \oplus f(100) = 010$	$f(110) \oplus f(010) = 101$
$f(111) \oplus f(110) = 110$	$f(111) \oplus f(101) = 100$	$f(111) \oplus f(011) = 000$
$hw = 10 \neq 3 * 2^{3-1}$	$hw = 6 \neq 3 * 2^{3-1}$	$hw = 10 \neq 3 * 2^{3-1}$

3.2.4 CRITÉRIO DE ESTREITO AVALANCHE (SAC)

Essa métrica foi proposta por (WEBSTER, 1986) e é considerada uma combinação das métricas de completude e efeito avalanche. Por esse motivo, seu objetivo também é avaliar a relação de dependência entre os bites de entrada e saída de uma dada função.

Se uma função satisfaz o SAC significa que os bites de saída devem ser substituídos com probabilidade igual à 0.5, sempre que um único bite de entrada é complementado. Em outras palavras, considerando X e X_i, duas entradas da função f, diferenciando apenas no bite i, seja:

$$V_i = Y \oplus Y_i \tag{3.7}$$

onde $Y = f(X)$ e $Y_i = f(X_i)$, a função f satisfaz o SAC se a probabilidade do número de 1's no vetor V_i for igual à 0.5 para todas as combinações de pares X e X_i.

(KIM, 1990) expressou essa definição da seguinte forma: Uma função $f(x) : V_2^n \to V_2^n$ satisfaz o SAC se a equação 5.5 for satisfeita.

$$\sum_{x \in V_2^n} f(x) \oplus f(x \oplus c_i^{(n)}) = (2^{n-1}, 2^{n-1}, ..., 2^{n-1}) \qquad \forall 1 \leq i \leq n \tag{3.8}$$

3.2.5 CORRELAÇÃO

Essa métrica também foi sugerida por (WEBSTER, 1986) e tem como objetivo avaliar se dois vetores são independentes, quando essa característica é verificada, o coeficiente de correlação entre dois vetores é igual a zero, porém, nem sempre que o coeficiente é igual a zero significa que os vetores são independentes. Para medir essa correlação pode ser usado o coeficiente de Pearson definido na equação 3.9.

$$corr = \frac{\sum_{i=0}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=0}^{n}(x_i - \overline{x})^2 \sum_{i=0}^{n}(y_i - \overline{y})^2}} \tag{3.9}$$

onde x e y são os vetores avalanches e $\overline{x}$ e $\overline{y}$ representam a média dos respectivos vetores.

Quando x e y são totalmente independentes, $corr = 0$, já quando $corr = 1$, significa que x e y são totalmente dependentes e, quando $corr = -1$, significa que x e y são complementares.

Não obstante, se x ou y forem vetores tais que todas as posições sejam iguais a um, ao utilizar a correlação de Pearson o denominador seria igual a zero. Assim, optou-se por utilizar o coeficiente de correlação conforme definido por (FULLER, 2003) e (BURNETT, 2005):

$$cc(f, g) = 1 - \frac{d(f, g)}{2^{n-1}} \tag{3.10}$$

3.2.6 BIJETIVIDADE

Segundo (MENEZES, 1996), uma função *one-to-one* é definida por dois conjuntos X e Y e uma regra f a qual associa para cada elemento de X um único elemento de Y.

O conjunto X é chamado de domínio da função, e Y o contradomínio. Se x é um elemento de X ($x \in X$), então a imagem de x é um elemento em Y o qual a função f associa com x. A imagem y de x é representada por $y = f(x)$. A notação utilizada para representar a associação f do conjunto X para o conjunto Y é $f : X \to Y$.

Sendo assim, se a função $f : X \to Y$ é *one-to-one* e a $Im(f) = Y$, então f é chamada função bijetora.

Avaliar a bijetividade é importante pois as funções com essas características são imunes aos ataques de colisão.

3.2.7 ANÁLISE DA FREQUÊNCIA DE VÁRIOS PESOS DE HAMMING

Essa métrica foi definida por (MAR, 2008) e tem como objetivo avaliar se para uma dada função $f(x) : V_2^n \to V_2^n$ a sua distribuição é caracterizada por uma distribuição binomial.

Para a utilização dessa métricas o seguinte procedimento foi estabelecido:

a) Entrada: caixa-S de tamanho m, sendo m o número de bites .

b) Saída: Frequência dos pesos de Hamming.

c) Algoritmo:

1) Escolher aleatoriamente um $x \in Z_2^m$ e encontra $y = S(x)$;

2) Escolher aleatoriamente outro $x' \in Z_2^m$ e encontra $y' = S(x')$;

3) Computar o valor diferencial das saídas: $\triangle y = y \oplus y'$;

4) Encontrar o peso de hamming do valor diferencial: $hw(\triangle y)$;

5) Repetir os passos de 1 à 4 para compor a amostra de análise;

6) Analisar a frequência dos pesos de hamming;

A FIG 3.2 ilustra o comportamento de uma função que satisfaz o critério de avaliação, enquanto a FIG 3.3 ilustra o comportamento de uma função não recomendável para utilização em algoritmos criptográficos. Quando a avaliação apresentar um comportamento semelhante ao da FIG 3.2, significa que a mesma possui boas propriedades de SAC.

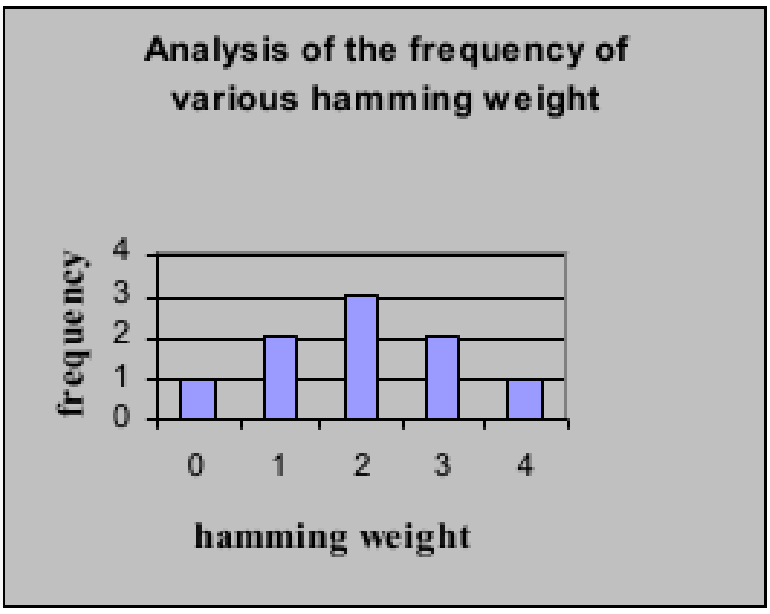

FIG. 3.2: Comportamento de uma recomendável caixa-S para o DES (MAR, 2008)

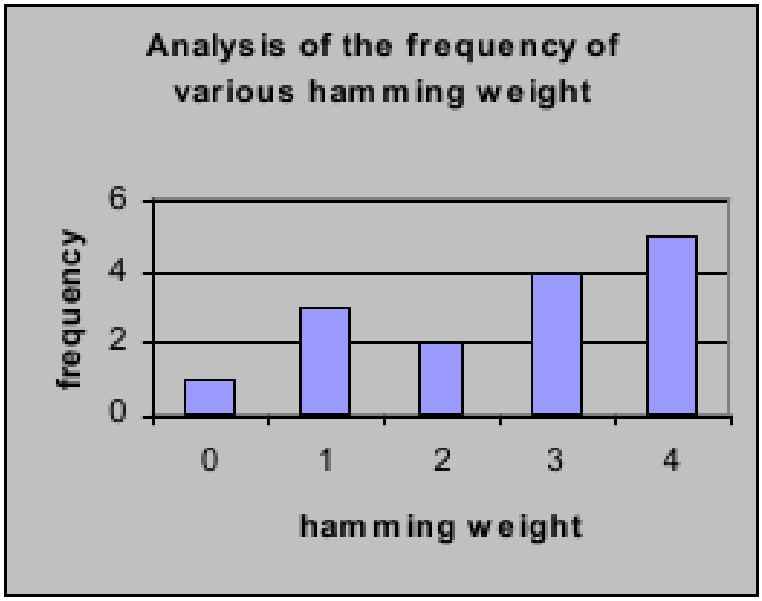

FIG. 3.3: Comportamento de uma não recomendável caixa-S para o DES (MAR, 2008)

3.2.8 ANÁLISE DA FREQUÊNCIA DE VÁRIOS VALORES DIFERENCIAIS

Essa métrica também foi definida por (MAR, 2008) e tem como objetivo analisar se a frequência dos vetores avalanches é caracterizada por uma distribuição uniforme. Uma função que satisfaz essa métrica torna-se mais resistente contra ataques de texto em claro conhecido.

Para a utilização dessa métricas o seguinte procedimento foi estabelecido:

a) Entrada: caixa-S de tamanho m, sendo m o número de bites .

b) Saída: Frequência de vários valores diferencias $\triangle y$.

c) Algoritmo:

1) Escolher aleatoriamente um $x \in Z_2^m$ e encontra $y = S(x)$;

2) Escolher aleatoriamente outro $x' \in Z_2^m$ e encontra $y' = S(x')$;

3) Computar o valor diferencial das saídas: $\triangle y = y \oplus y'$;

4) Repetir os passos de 1 à 3 para compor a amostra de análise;

5) Analisar a frequência dos valores diferencias;

A FIG 3.4 ilustra o comportamento de uma uma função que satisfaz o critério de avaliação, enquanto a FIG 3.5 ilustra o comportamento de uma função não recomendável para

algoritmos criptográficos. Quando a avaliação apresentar um comportamento semelhante ao da FIG 3.4, significa que a mesma possui boas propriedades de completude.

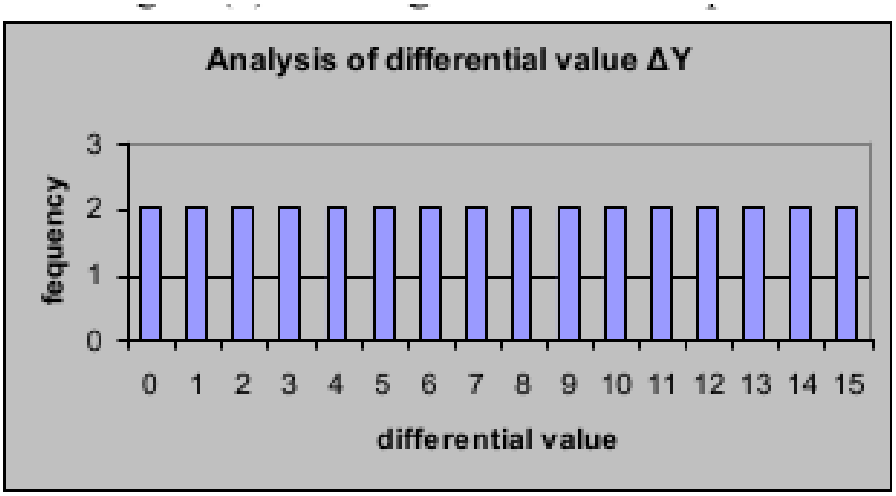

FIG. 3.4: Comportamento de uma recomendável caixa-S para o DES (MAR, 2008)

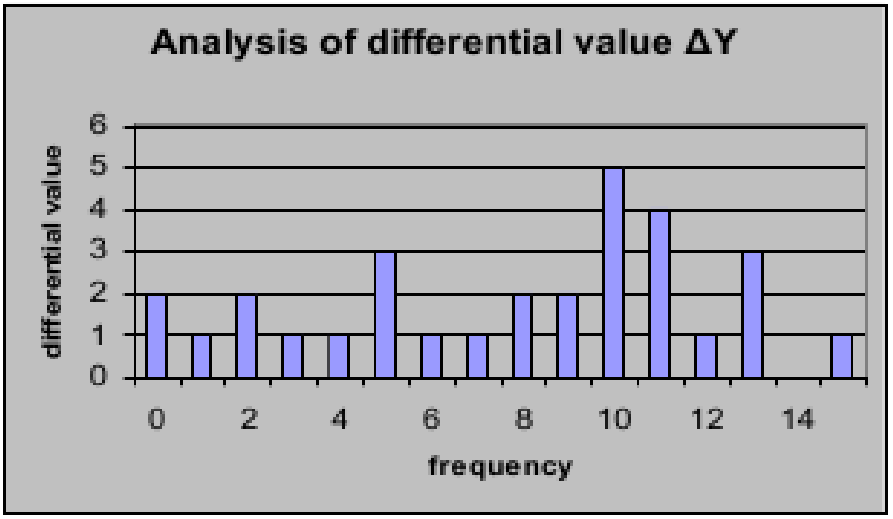

FIG. 3.5: Comportamento de uma não recomendável caixa-S para o DES (MAR, 2008)

3.2.9 ANÁLISE DOS PESOS DE HAMMING DE ACORDO COM A POSIÇÃO DO BITE

Assim como as duas métricas anteriores, esse métrica também foi definida por (MAR, 2008) e tem como objetivo analisar se frequência do índice dos vetores avalanches é caracterizada por uma distribuição uniforme.

Para a utilização dessa métricas o seguinte procedimento foi estabelecido:

a) Entrada: caixa-S de tamanho m, sendo m o número de bites .

b) Saída: Pesos de hamming de acordo com a posição do bite.

c) Algoritmo:

1) Escolher aleatoriamente um $x \in Z_2^m$ e encontra $y = S(x)$;

2) Escolher aleatoriamente outro $x' \in Z_2^m$ e encontra $y' = S(x')$;

3) Computar o valor diferencial das saídas: $\triangle y = y \oplus y'$;

4) Repetir os passos de 1 à 3 para compor a amostra de análise;

5) Analisar o peso de hamming de acordo com a posição do bites dos valores diferenciais;

A FIG 3.6 ilustra o comportamento de uma uma função que satisfaz o critério de avaliação, enquanto a FIG 3.7 ilustra o comportamento de uma função não recomendável para algoritmos criptográficos. Quando a avaliação apresentar um comportamento semelhante ao da FIG 3.6, significa que a mesma possui boas propriedades de SAC.

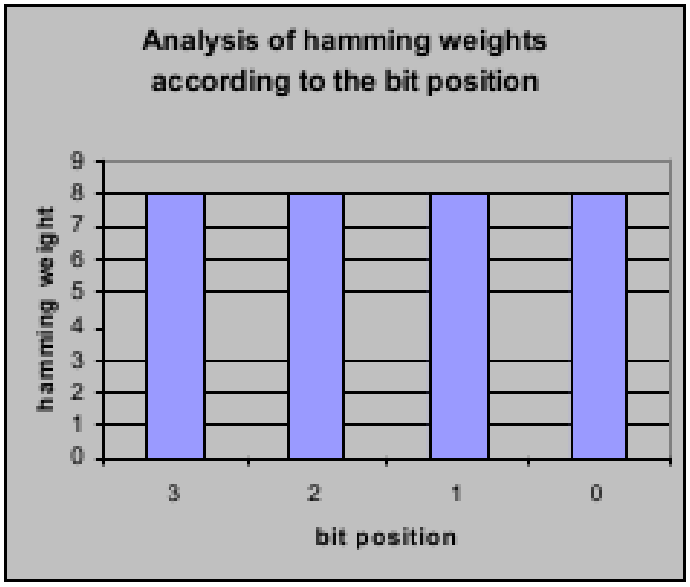

FIG. 3.6: Comportamento de uma recomendável caixa-S para o DES (MAR, 2008)

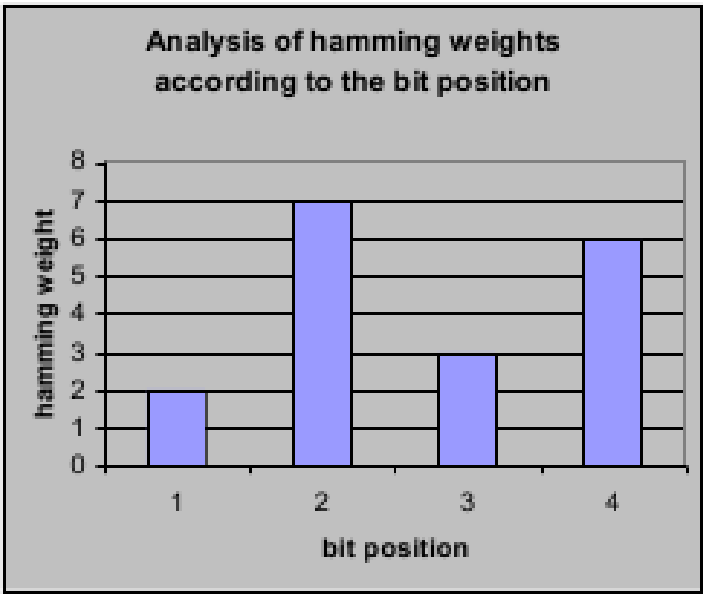

FIG. 3.7: Comportamento de uma não recomendável caixa-S para o DES (MAR, 2008)

3.3 IMPORTÂNCIA DAS MÉTRICAS PARA CRIPTOGRAFIA

O nível de segurança de um algoritmo criptográfico é um importante critério para a avaliação do algoritmo. Embora (SHANNON, 1949) tenha definido o conceito de segurança perfeita, (JUNOD, 2005) menciona que sistemas com essas características ainda são impraticáveis no mercado. Isso demonstra porque a avaliação e a seleção de métricas são tão importantes.

Nesse contexto, as métricas anteriormente apresentadas são importantes para o desenvolvimento de funções booleanas aplicadas à criptografia. Uma vez que as funções satisfaçam todos os critérios de avaliação, dão a garantia de maior segurança para os algoritmos nos quais elas serão utilizadas, haja vista que essas métricas tem como objetivo avaliar a resistência das funções levando em consideração os pontos explorados pelos ataques linear e diferencial. Além disso, essas métricas demonstram que o desenvolvimento de algoritmos criptográficos não pode ser realizado de forma totalmente aleatória, é evidente que critérios precisam ser seguidos.

4 AVALIAÇÃO DO ALGORITMO AES

Este capítulo tem como objetivo apresentar o processo de avaliação realizado no algoritmo AES, a fim de encontrar fraquezas e possíveis causas para a geração de padrões, conforme apresentadas nos trabalhos de (CARVALHO, 2006); (DILEEP, 2006); (SOUZA, 2007); (SOUZA, 2008); (NAGIREDDY, 2008); (TORRES, 2010) e (OLIVEIRA, 2011). Para isso, foi utilizada como ferramenta de análise o ataque *Differential Fault Analysis* (DFA) proposto por (GIRAUD, 2004).

4.1 DFA

Em 1997, Biham e Shamir publicaram um ataque intitulado como *Differential Fault Analysis* (DFA). O princípio básico do ataque é induzir falhas em *bytes* ou bites de dispositivos *smartcards* utilizando um meio físico como *lasers* ou eletromagnetismo. Através dessas falhas pode-se comparar a cifra original (cifra sem falha) com a cifra induzida (cifra com falha), e explorando características do criptossistema em questão pode-se até mesmo obter a chave cifradora. No artigo original, (BIHAM, 1997) aplicaram o ataque sobre o *Data Encryption Standard* (DES), nesta simulação os autores do artigo conseguiram decifrar a chave secreta, utilizando uma amostra entre 50 e 200 cifras.

Com o sucesso do modelo vários trabalhos foram desenvolvidos nessa linha de pesquisa e levando em consideração que o AES é um dos algoritmos mais populares por ser o padrão norte americano, várias extensões de ataques DFA foram propostas para o AES, como por exemplo, os trabalhos de (DUSART, 2003), (CHEN, 2003) e (GIRAUD, 2004).

Segundo (GIRAUD, 2004), o DFA é frequentemente usado para testar a segurança da criptografia de aplicações *smartcards* levando em consideração a avaliação dos dados e do comportamento das funções em cada rodada do algoritmo. Nesse contexto, optou-se por utilizar o teórico DFA ataque proposto por (GIRAUD, 2004) com o objetivo de identificar fraquezas na estrutura do algoritmo e propor modificações ao AES, a fim de aumentar a resistência do algoritmo frente ao ataque em questão, e até mesmo eliminar os padrões encontrados no modo *Electronic codebook* (ECB). A justificativa para análise do AES sobre o referido ataque é que (BIHAM, 1997), (BIEHL, 2000), (BONEH, 1997) e (BONEH, 2001) também utilizaram a mesma estrutura para atacar outros criptosistemas,

mostrando dessa forma a aceitação do ataque.

4.2 DESCRIÇÃO DO AES

Esta sessão tem por finalidade descrever as funções utilizadas no algoritmo AES, a fim de facilitar o entendimento da sessão 4.3, todavia, para maiores detalhes deste algoritmo recomenda-se a leitura da FIPS 197 (NIST, 2001) e (DAEMEN, 2002).

O algoritmo AES é composto por quatro transformações (*SubBytes, ShitRows, MixColumns e AddRoundKey*) e suas respectivas inversas além de uma expansão de chave (*KeyExpansion*). Os dados de entrada são compostos por blocos de 128 bites e chaves de 128, 192 ou 256 bites. O número de rodadas realizadas no AES é dependente do tamanho da chave secreta, para determinar esse número é estabelecido a seguinte relação:

$$Nr = Nk + Nb + 2 \tag{4.1}$$

Sendo Nr o número de rodadas, Nk o número de bites da chave dividido por 32 e Nb o número de colunas da matriz M nesse caso $Nb = 4$ pois o tamanho do bloco sempre é igual a 128 bites.

A seguir serão descritas as quatro funções que compõe as Nr rodadas do algoritmo AES e a função *KeyExpansion*. É importante ressaltar que a última rodada do AES omite a transformação MixColumns e segundo (DAEMEN, 1998) essa omissão é feita para tornar o processo de cifragem e decifragem mais similar em termos de estrutura, não aumentando ou diminuindo a segurança do algoritmo de alguma forma.

4.2.1 SUBBYTES

A transformação SubBytes é a única transformação não linear utilizada no algoritmo AES, ela opera independentemente sobre cada byte da matriz M da seguinte forma:

- Realiza o inverso multiplicativo sobre $GF(2^8)$ para cada elemento da matriz M de tal forma que $M_{i,j}\lambda \equiv 1 \ \% \ (x^8 + x^4 + x^3 + x + 1)$. Sendo λ o inverso multiplicativo de $M_{i,j}$.

- Em seguida aplica a transformação afim sobre $GF(2)$:

$$\lambda_i^{'} = \lambda_i \oplus \lambda_{(i+4)\%8} \oplus \lambda_{(i+5)\%8} \oplus \lambda_{(i+6)\%8} \oplus \lambda_{(i+7)\%8} \oplus c_i \tag{4.2}$$

para $0 \leq i < 8$ onde λ_i é o i-ésimo bite do *byte* λ e c_i é o i-ésimo *byte* da constante $c = \{01100011\}$.

4.2.2 SHIFTROWS

A transformação ShiftRows realiza uma permutação cíclica sobre as linhas da matriz M obedecendo o seguinte critério:

$$M_{i,j} = M_{i,(j+i)\%Nb} \quad \text{para} \quad 0 < i < 4 \quad \text{e} \quad 0 \leq j < Nb \tag{4.3}$$

4.2.3 MIXCOLUMNS

A transformação MixColumns trata cada coluna da matriz M como um polinômio sobre $GF(2^8)$ multiplicando cada coluna módulo $x^4 + 1$ pelo polinômio fixo $\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$.

4.2.4 ADDROUNDKEY

A transformação AddRoundKey adiciona à matriz M a respectiva chave da rodada por meio da operação *XOR*.

4.2.5 KEYEXPANSION

A função KeyExpansion tem como objetivo gerar um dicionário de chaves com $Nb * (Nr + 1)$ palavras (32 bites) utilizando como entrada a chave secreta. A Figura 4.1 ilustra o pseudocódigo da função presente em (NIST, 2001)

4.3 ATAQUE DFA TEÓRICO PROPOSTO POR GIRAUD (2004)

Nesse ataque é considerada a hipótese de que apenas um bite da matriz M seja alterado antes da última rodada do AES. A partir dessa característica Giraud demonstra como obter a chave secreta utilizada no processo. Para essa demonstração foi considerada a chave de 128 bites.

A Figura 4.2 ilustra a última rodada do algoritmo AES utilizando uma chave de 128 bites. A última rodada também pode ser representada de acordo com a equação 4.4.

$$C = ShiftRows(SubBytes(M^9)) \oplus K^{10} \tag{4.4}$$

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

FIG. 4.1: Pseudocódigo para Key Expansion (NIST, 2001)

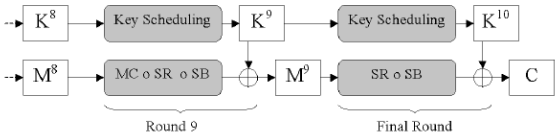

FIG. 4.2: A última rodada de um AES-128 (GIRAUD, 2004)

Sendo $SubBytes(M_j^i)$ o resultado da substituição aplicada sobre o byte M_j^i e $ShiftRows(j)$ a posição do j-ésimo byte da matriz M depois de aplicada a transformação $ShiftRows$. Então, da equação 4.4 é obtida a equação 4.5.

$$C_{ShiftRows(i)} = SubBytes(M_i^9) \oplus K_{ShiftRows(i)}^{10}, \quad \forall i \in \{0, ..., 15\} \tag{4.5}$$

Se for induzida a falha e_j sobre um bite do j-ésimo byte da matriz M^9 antes da última rodada, então pela equação 4.6 é obtida a cifra com falha induzida D.

$$D_{ShiftRows(j)} = SubBytes(M_j^9 \oplus e_j) \oplus K_{ShiftRows(j)}^{10} \tag{4.6}$$

E para todo $i \in \{0, ..., 15\}$ sendo $i \neq j$ é obtida pela equação 4.7 os restante da cifra com falha induzida D.

$$D_{ShiftRows(i)} = SubBytes(M_i^9) \oplus K_{ShiftRows(i)}^{10} \tag{4.7}$$

Então, se não existe injeção de falha sobre o i-ésimo byte M^9, é obtida das equações 4.5 e 4.7 a equação 4.8:

$$C_{ShiftRow(i)} \oplus D_{ShiftRow(i)} = 0 \tag{4.8}$$

E se existe uma injeção de falha sobre o byte M_j^9, é obtida das equações 4.5 e 4.6 a equação 4.9:

$$C_{ShiftRow(j)} \oplus D_{ShiftRow(j)} = SubByte(M_j^9) \oplus SubByte(M_j^9 \oplus e_j) \tag{4.9}$$

O primeiro passo do ataque é determinar $ShiftRow(j)$ que é a posição do único byte diferente de zero de $C \oplus D$ e então obter j. Em seguida é usado um método de contagem para encontrar o byte M_j^9. Para cada injeção e_j são encontrados os possíveis valores de M_j^9 que são verificados pela equação 4.9. Para cada ocorrência do possível valor de M_j^9, o seu método de contagem é incrementado por um. Com outra injeção e_j é esperado que a contagem do correto valor de M_j^9 seja mais frequente que os demais candidatos. Esse processo é repetido para todos os bytes até obter M^9.

Conhecendo o valor de C e de M_j^9, é possível obter o valor de K^{10} pela equação 4.4, e consequentemente aplicando a inversa *Key Scheduling* para K^{10} é possível obter a chave secreta K.

Para descobrir os possíveis bytes de M_j^9, são testados $8 * 2^8$ possíveis valores, ou seja, para todos os valores de x entre 0 e 255 e para todas a variações de e_j, é verificado se a equação 4.9 é verdadeira.

Fixando o lado esquerdo da equação 4.9 com os 255 possíveis valores e testando todos os possíveis pares (x, e_j), é observado que o número de possíveis valores para M_j^9 varia entre 2 e 14 com média 8. Supondo o pior caso, são obtidos 14 possíveis valores para cada injeção $, e_j$. Para cada conjunto existe o correto valor de M_j^9, logo, para identificar esse valor, os outros 13 elementos devem ser diferentes de cada um. Denotando A como os 13 valores obtidos na primeira injeção e B como o conjunto dos possíveis valores exceto M_j^9 obtidos na segunda injeção, pode-se identificar o correto valor com probabilidade:

$$P(A \cap B = \oslash) = P(|A \cap B| = 0) = \frac{\binom{255}{13} * \binom{255-13}{13}}{\binom{255}{13}^2} \simeq 50\% \tag{4.10}$$

Com uma terceira injeção é obtido um novo conjunto C, e a probabilidade para encontrar

o correto valor de M_j^9 passa a ser de:

$$\begin{aligned}
P_3 &= P(A \cap B \cap C = \oslash) \\
&= P(|A \cap B \cap C| = 0) \\
&= \sum_{k=0}^{min\{|A|,|B|\}} P(|A \cap B| = k, |A \cap B \cap C| = 0) \\
&= \sum_{k=0}^{13} P(|A \cap B| = k) * P(|A \cap B \cap C / |A \cap B| = k) \\
&= \sum_{k=0}^{13} \frac{\binom{255}{13} * \binom{13}{k} * \binom{255-13}{13-k}}{\binom{255}{13}^2} * \frac{\binom{255}{k} * \binom{255-k}{13}}{\binom{255}{k} * \binom{255}{13}} \\
&\simeq 97\% \qquad (4.11)
\end{aligned}$$

Nos experimentos realizados foram utilizados 10 injeções e em todos os casos foi possível descobrir a chave secreta utilizada. Os experimentos realizados foram simulados em software, ou seja, não foram feitas injeções de falhas utilizando um meio físico.

4.4 ANÁLISE E MODIFICAÇÕES REALIZADAS NO AES

(GIRAUD, 2004) conseguiu chegar a equação 4.9 devido a três características do AES: a primeira característica explorada foi a operação *AddRoundKey*, uma vez que a transformação adiciona uma chave de rodada à matriz M por meio de uma simples operação XOR. Foi possível eliminar essa transformação da equação 4.9 pois as duas chaves K^{10} se anulam; a segunda característica explorada foi a omissão da transformação *MixColumns*, como essa operação é omitida da última rodada do AES restaram apenas operações que manipulam cada *byte* da matriz M individualmente. E por fim, a última característica explorada foi a transformação *SubBytes*. Como essa transformação é de conhecimento público e é totalmente reversível, foi possível, por exaustão, descobrir, a partir da equação 4.9, a matriz M^9 e consequentemente a chave secreta utilizada no processo.

Nesse contexto, a afirmação de (DAEMEN, 1998) de que a omissão da transformação *MixColumns* na última rodada do AES não aumenta ou diminui a segurança do algoritmo passa a ser questionada. E de fato foi, (DUNKELMAN, 2010) mostram que para certos ataques aplicados sobre o AES, a ausência da transformação *MixColumns* na última rodada pode diminuir a complexidade do ataque. Partindo dessa observação, a inserção da transformação *MixColumns* é a primeira modificação que foi realizada no AES para

aumentar a resistência do algoritmo contra o ataque em estudo. A justificativa para inserir a transformação *MixColumns* é proveniente da sua própria característica. Uma vez que cada coluna da matriz M é considerada um polinômio, a equação 4.4 passaria a ser representada pela equação 4.12.

$$C = MixColumns(ShiftRows(SubBytes(M^9))) \oplus K^{10} \tag{4.12}$$

Dessa forma, considerando a mesma configuração do ataque, não seria possível obter a equação 4.9, pois cada termo M_j^9 seria dependente de outros três termos. Logo, pode-se perceber que a simples inserção da transformação *MixColumns* na última rodada do AES aumenta a resistência do algoritmo contra o teórico ataque DFA proposto por (GIRAUD, 2004). Como se não bastasse, essa análise reafirma as observações feitas por (DUNKELMAN, 2010) e questiona a afirmação feita por (DAEMEN, 1998), de que a omissão da *MixColumns* na última rodada do AES não diminuiria a segurança do algoritmo.

A segunda modificação realizada foi sobre a transformação *SubBytes*. Essa transformação, por ser a única operação não linear do algoritmo, vem sendo muito pesquisada e consequentemente surgem muitas propostas de novas transformações *SubBytes*, como por exemplo, os trabalhos de (JANADI, 2008), (KUDOU, 2009), (LIU, 2005), (LI, 2009) e (KAZLAUSKAS, 2009). O foco dessas modificações está em tornar as caixas-S dinâmicas e em alguns casos dependentes da chave secreta.

Essa característica é muito interessante, pois a caixa-S apesar de continuar sendo pública, consegue dificultar o ataque, já que para utilizar a equação 4.9 é necessário o conhecimento da caixa-S para realizar a busca por exaustão de cada *byte* da matriz M^9. Pensando nessa situação, a presente pesquisa descreve no capítulo 5 uma nova proposta de construção de caixas-S. Com essa nova função, a equação 4.9 pode ser reescrita de acordo com a equação 4.13. Sendo $SubByte'$ a nova caixa-S apresentada no capítulo 5. A partir dessa função, o algoritmo AES passa a ter mais resistência pois esta nova função é dinâmica e dependente da chave secreta, sendo que por exaustão seriam necessárias 2^{128} combinações para descobrir o valor de M^9.

$$C_{ShiftRow(j)} \oplus D_{ShiftRow(j)} = SubByte'(M_j^9) \oplus SubByte'(M_j^9 \oplus e_j) \tag{4.13}$$

Finalmente, a função de expansão de chave é a última modificação que pode ser efetuada em decorrência da análise realizada. Essa função apresenta uma falha de segurança,

por ser uma função reversível a partir de uma subchave, o que não é necessário e nem recomendável. Partindo da ideia de que o ataque fosse realizado e fosse obtida a subchave K^{10}, ainda assim não seria possível obter a chave secreta, caso a função de expansão de chave fosse irreversível. Dessa forma, nessa pesquisa também será apresentada uma nova função de expansão de chave, que seja irreversível a partir de uma subchave de rodada (vide capíitulo 6).

Quanto aos padrões encontrados no algoritmo AES utilizando o modo de operação ECB, espera-se que esses sejam eliminados após aplicadas a nova caixa-S e função de expansão de chave. Para maiores detalhes, o capítulo 7 apresentará o resultado dos experimentos, em relação ao reconhecimento de padrões realizados no algoritmo AES já modificado.

4.5 CONCLUSÕES DO CAPÍTULO

Utilizar o ataque DFA como ferramenta de análise do algoritmo AES foi de suma importância, já que esse ataque parte do princípio de que é possível analisar o algoritmo durante o processo de cifragem. Nesse contexto, foi possível perceber a influência de cada função durante o processo de cifragem, e perceber que a caixa-S é o principal ponto de ataque, por ser a única transformação não linear. Ademais, foi possível fortalecer a afirmação de (DUNKELMAN, 2010) de que a ausência da transformação *MixColumns* na última rodada do AES pode diminuir a complexidade de certos ataques. Ainda nesse contexto, a utilização do ataque evidenciou que a função de expansão de chave, por ser reversível, aumenta a insegurança do algoritmo.

5 MODELO PARA A CONSTRUÇÃO DE CAIXAS-S COM BASE EM CAMINHOS ALEATÓRIOS

Este capítulo tem como objetivo apresentar um modelo para a construção de caixas-S baseadas em caminhos aleatórios. Para demonstrar a eficiência do modelo, realizou-se a construção de uma caixa-S que foi submetida à avaliação das métricas apresentadas no capítulo 3. Uma vez que os resultados dos testes foram satisfatórios, pode-se considerar que a caixa-S possui um bom nível de aleatoriedade e não linearidade, portanto, considerada resistente aos ataques de criptoanálise linear e diferencial. Além disso, por ser construída com base no modelo apresentado neste capítulo, a caixa-S também apresenta maior resistência ao ataque DFA proposto por (GIRAUD, 2004) conforme descrito no capítulo 4 deste trabalho.

5.1 TRABALHOS RELACIONADOS

No universo da criptografia simétrica em bloco, a caixa-S é um importante componente, pois é a transformação responsável pela confusão dos bites cifrados interferindo diretamente na segurança do algoritmo criptográfico.

Segundo (CHEN, 2004), geralmente as funções caixas-S são projetadas de duas formas: ou utilizando funções matemáticas ou de forma aleatória. Nesse contexto, pode-se observar que propostas de caixas-S que utilizam funções matemáticas surgem com maior frequência, como por exemplo (LIU, 2005), (KUDOU, 2009), (ZAÏBI, 2009) e (TRAN, 2008). As estruturas matemáticas mais utilizadas para a confecção de caixas-S são geralmente funções logarítmicas; funções exponenciais; estruturas algébricas e mapeamento caótico, ou seja, funções fundamentadas em teoria do caos. Já as propostas de caixas-S projetadas de forma aleatória, geralmente utilizam estruturas como caminhos aleatórios ou heurísticas, como por exemplo (KAZMI, 2009), (CHEN, 2004) e (BURNETT, 2005).

Ademais, as caixas-S também são classificadas em estáticas ou dinâmicas. As caixas-S estáticas geralmente são representadas por tabelas de mapeamento, esse tipo de representação é utilizado pela maioria dos algoritmos de criptografia simétrica em bloco, como é o caso do AES. Já as caixas-S dinâmicas, geralmente utilizam o processo *on-the-fly*[1] e são

[1]Neste trabalho, toda vez que o termo on-the-fly aparecer, significa procedimento realizado a cada

dependentes da chave criptográfica.

A utilização de caixas-S dinâmicas agrega maior segurança para os algoritmos criptográficos, entretanto, normalmente o tempo de processamento dessas caixas-S é maior. Logo, fica a critério do projetista e da necessidade de comunicação a escolha da estrutura mais conveniente para a construção de caixas-S.

Nesse contexto, este trabalho apresenta um novo modelo para a construção de caixas-S. Seu diferencial está na utilização de caminhos aleatórios de tal forma que cada caminho dependa de um chave secreta. Outro diferencial está na abstração das funções utilizadas para a composição dos vértices, ou seja, a estrutura de caminhos aleatórios projetada, independe das escolhas das funções que comporão os vértices do grafo.

5.2 DESCRIÇÃO DO MODELO

As caixas-S são representadas por funções na forma $f : \{0,1\}^N \rightarrow \{0,1\}^M$ que representa o mapeamento de N bites de entrada para M bites de saída. Quando $N > M$, se todas as possíveis saídas são mapeadas pela função f, então diz-se que a caixa-S é sobrejetiva. Quando $N < M$, se todas as saídas são distintas, então diz-se que a caixa-S é injetiva. E quando $N = M$, se a função f for sobrejetiva e injetiva, diz-se que é bijetiva.

Sendo assim, o modelo apresentado terá como objetivo construir funções em que $N = M$ com características bijetivas. Para isso, seja um grafo direcionado $G = (V, E)$, em que $V = \{x_1, x_2, ..., x_n\}$ são os vértices, E as arestas (x_i, x_j) e n o número de vértices. Essa estrutura será utilizada para a construção do modelo de caixas-S da seguinte forma:

- Cada vértice possuirá uma transformação matemática responsável por manipular os bites.
- Um vetor de bites (chave) com tamanho maior ou igual a 128 será utilizado para determinar o caminho aleatório da transformação, ou seja, os vértices que serão utilizados para a transformação de acordo com a chave específica.
- Para a transformação, escolhe-se uma posição i da chave como ponto de partida e $x_{i \mod 128}$ será o vértice inicial para a leitura do grafo. A partir desse ponto, a variável i sempre será incrementada em uma unidade até que todas as posições do vetor sejam percorridas.

rodada do algoritmo

- Para determinar os vértices utilizados, a seguinte condição foi estabelecida: Se o bite da posição i for igual a um, então o vértice $x_{i \mod 128}$ será utilizado para a confusão dos bites.

- Na transformação inversa a mesma posição i é escolhida como ponto de partia, entretanto, $x_{(i-129) \mod 128}$ é determinado como o vértice inicial. E a partir desse ponto, a variável i é decrementada em uma unidade até que todas as posições do vetor sejam percorridas.

- Os vértices utilizados na transformação inversa são os mesmos da transformação direta, logo, a mesma regra é estabelecida: Se o bite da posição i for igual a um, então o vértice $x_{(i-129) \mod 128}$ será utilizado para a confusão dos bites.

A FIG 5.1 exemplifica o procedimento de leitura da caixa-S.

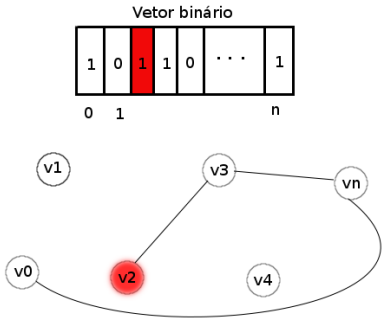

FIG. 5.1: Exemplo de leitura da caixa-S

Os algoritmos 5.1 e 5.2 representam as transformações das caixas-S direta e inversa respectivamente.

Como observado, o modelo para a construção de caixas-S é dependente de uma chave secreta, e por esse motivo as caixas-S são dinâmicas. Logo, as caixas-S baseadas nesse modelo necessitam utilizar o processo *on-the-fly*, ou seja, o processo de confusão terá que ser realizado a cada rodada sem a possibilidade de se utilizar uma tabela de mapeamento pré-processada, diferentemente do que ocorre com a maioria das caixas-S estáticas, como por exemplo a caixa-S original do AES.

Transformações *on-the-fly* implicam em vantagens e desvantagens. A desvantagem é que a transformação consumirá maior tempo de processamento se comparadas com as tabelas pré-processadas. E a vantagem é que consumirá menos memória, e pelo fato de ser dependente da chave secreta, dificulta para o inimigo conhecer qual a ordem das

5.1 Transformação direta

1: $chave[n] \leftarrow \{0,1\}_n$ # Chave com n posições contendo valores binários
2: $p \leftarrow i$ # Posição inicial para a leitura da chave
3: $byte \leftarrow y$ # Byte que sofrerá a transformação
4: **for** $j \leftarrow 1$ até n **do**
5: **if** $x_{p \ mod \ n} = 1$ **then**
6: $byte \leftarrow x_{p \ mod \ 128}(byte)$
7: **end if**
8: $p \leftarrow p + 1$
9: **end for**
10: **return** byte

5.2 Transformação inversa

1: $chave[n] \leftarrow \{0,1\}_n$ # Chave com n posições contendo valores binários
2: $p \leftarrow i$ # Posição inicial para a leitura da chave
3: $byte \leftarrow y$ # Byte que sofrerá a transformação
4: **for** $j \leftarrow 1$ até n **do**
5: **if** $x_{(p-129) \ mod \ n} = 1$ **then**
6: $byte \leftarrow x_{(p-129) \ mod \ 128}(byte)$
7: **end if**
8: $p \leftarrow p - 1$
9: **end for**
10: **return** byte

funções utilizadas na transformação. No caso específico das caixa-S que utilizam o modelo apresentado neste capítulo, o número de combinações possíveis é igual a 2^{128}, ou seja, por força bruta, no pior caso, o inimigo teria que testar 2^{128} possíveis caminhos aleatórios para descobrir a combinação correta da caixa-S dependente de uma chave secreta.

5.3 CARACTERÍSTICAS DO MODELO

A chave secreta é responsável por determinar a ordem e os vértices utilizados na transformação, logo, a probabilidade de um dado vértice x_i ser utilizado na transformação é igual a 0.5, haja vista que a chave consiste em um vetor binário. Não obstante, como a ordem

da transformações influencia, a probabilidade de um dado vértice x_i ir para um vértice x_j através aresta (x_i, x_j) é igual a $\frac{1}{2^{|j-i|}}$. Em outras palavras, $|log_2 \frac{1}{2^{|j-i|}}| - 1$ indica o número de zeros entre dois bites uns presentes no vetor da chave secreta.

Outra característica importante é o número de vértices utilizados na transformação. Para que todos os vértices sejam utilizados, é necessário que a chave secreta possua todos os bites iguais a um, todavia, geralmente as chaves secretas são produzidas por geradores de chaves aleatórias ou funções de expansão de chave, e para a aceitação de bons geradores, e funções de expansão de chaves, são empregados testes estatísticos cujo objetivo é avaliar a aleatoriedade da sequência binária resultante. Dentre os testes recomendados pelo NIST (RUKHIN, 2010) está o Teste de Frequência Monobit que visa verificar se para um determinado vetor binário o número de zeros é proporcional ao número de uns. Nesse diapasão, o Teste de Frequência Monobit é um dos requisitos para os geradores de chaves, e funções de expansão, serem considerados aleatórios, então, pode-se perceber que não serão utilizadas chaves não balanceadas para determinar um caminho aleatório no grafo e, por este motivo, o universo de 2^{128} possíveis caminhos não é totalmente explorado.

Considerando que os geradores de chaves aleatórias devem produzir chaves balanceadas mais ou menos o seu desvio padrão, o número de chaves com essa característica estaria em torno de 2^{127}, e este ainda é um número satisfatório para resistir a ataques de força bruta. Para o conhecimento deste número, a distribuição das chaves foi caracterizada levando em consideração o peso de *hamming* de acordo com a equação 5.1, sendo assim, uma distribuição binomial foi obtida em que o eixo de x representa os 128 grupos de chaves, e o eixo de y a função $f(x)$, que por sua vez representa a proporção de chaves com *hamming* igual a x.(ver FIG 5.2).

$$n_i = \frac{128!}{i! * (128 - i)!} \qquad \texttt{para} \quad 0 < i \leq 128 \tag{5.1}$$

n_i é o número de chaves pertencentes ao grupo com peso de *hamming* igual a i.

5.4 CONSTRUÇÃO DA CAIXA-S

A fim de validar a integridade e eficiência do modelo, a presente seção tem como objetivo apresentar a construção de uma caixa-S que utilize o modelo proposto neste trabalho.

Para a composição dos vértices, as funções responsáveis por manipular os bites foram dividias em cinco categorias, quais sejam: rotação cíclica; permutação entre bites; XOR

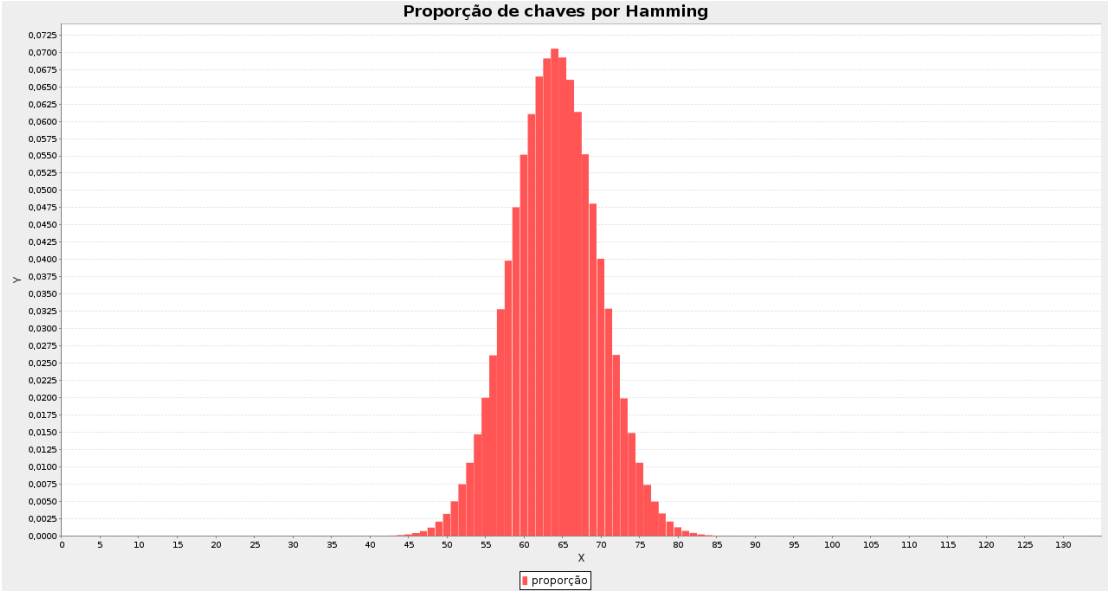

FIG. 5.2: Distribuição binomial dos grupos de chaves $\mu = 64$ e $\sigma = \sqrt{32}$

entre bites; XOR com constantes e mapeamento inverso em corpos finitos. A TAB 5.1 ilustra as funções utilizadas para a composição do grafo.

Essas categorias foram escolhidas devido à sua simplicidade, utilização difundida em algoritmos criptográficos e o baixo custo computacional. Já a escolha das funções foi realizada da seguinte forma: para a rotação cíclica foram utilizadas todas as possíveis rotações para um conjunto de oito bites. Nas categorias permutação entre bites e XOR entre bites a ordem foi estabelecida de forma aleatória. Já na categoria XOR com constantes, as constantes é que foram determinadas de forma aleatória. E na categoria inverso em $GF(2^n)$, os polinômios foram escolhidos de tal forma que fossem irredutíveis em seu corpo.

TAB. 5.1: Funções utilizadas para a composição do grafo.

Rotação Cíclica	Permutação entre bites	XOR entre bites	XOR com constantes	Inverso em $GF(2^n)$	Inverso em $GF(2^n)$
1- $x >>> 1$	15- $\{1,2,3,0,5,6,7,4\}$	31- $\{-,-,-,-,0,1,2,3\}$	47- $\{0,0,0,1,1,0,1,1\}$	64- $\{5_1 \cup 3_1\}$	81- $\{3_1 \cup 5_6\}$
2- $x >>> 2$	16- $\{1,6,5,2,0,7,3,4\}$	32- $\{-,5,6,-,0,-,-,3\}$	48- $\{0,1,0,1,1,1,0,1\}$	65- $\{5_1 \cup 3_2\}$	82- $\{3_2 \cup 5_1\}$
3- $x >>> 3$	17- $\{3,7,0,5,6,4,2,1\}$	33- $\{-,0,3,-,-,4,7,-\}$	49- $\{1,1,0,1,1,0,1,1\}$	66- $\{5_2 \cup 3_1\}$	83- $\{3_2 \cup 5_2\}$
4- $x >>> 4$	18- $\{5,3,7,1,6,0,4,2\}$	34- $\{-,-,0,1,6,7,-,-\}$	50- $\{1,1,1,1,0,0,1,1\}$	67- $\{5_2 \cup 3_2\}$	84- $\{3_2 \cup 5_3\}$
5- $x >>> 5$	19- $\{7,4,1,6,2,3,0,5\}$	35- $\{-,-,1,0,-,-,5,4\}$	51- $\{1,0,1,1,0,0,0,1\}$	68- $\{5_3 \cup 3_1\}$	85- $\{3_2 \cup 5_4\}$
6- $x >>> 6$	20- $\{1,5,6,4,3,2,7,0\}$	36- $\{1,-,3,-,5,-,7,-\}$	52- $\{1,1,0,1,0,1,0,1\}$	69- $\{5_3 \cup 3_2\}$	86- $\{3_2 \cup 5_5\}$
7- $x >>> 7$	21- $\{2,6,4,0,1,7,5,3\}$	37- $\{3,2,-,-,7,6,-,-\}$	53- $\{1,0,1,1,1,1,0,1\}$	70- $\{5_4 \cup 3_1\}$	87- $\{3_2 \cup 5_6\}$
8- $x <<< 1$	22- $\{4,0,5,7,2,3,1,6\}$	38- $\{-,0,-,2,-,4,-,6\}$	54- $\{0,0,1,1,1,1,1,1\}$	71- $\{5_4 \cup 3_2\}$	88- $\{4_1 \cup 4_1\}$
9- $x <<< 2$	23- $\{6,2,7,3,0,1,5,4\}$	39- $\{1,-,-,2,5,-,-,6\}$	55- $\{0,1,1,0,0,1,0,0\}$	72- $\{5_5 \cup 3_1\}$	89- $\{4_1 \cup 4_2\}$
10- $x <<< 3$	24- $\{4,7,3,2,5,0,1,6\}$	40- $\{6,-,-,5,2,-,-,1\}$	56- $\{1,0,0,1,0,0,1,1\}$	73- $\{5_5 \cup 3_2\}$	90- $\{4_1 \cup 4_3\}$
11- $x <<< 4$	25- $\{3,4,6,0,7,2,5,1\}$	41- $\{2,3,-,-,6,7,-,-\}$	57- $\{0,1,1,1,0,1,1,0\}$	74- $\{5_6 \cup 3_1\}$	91- $\{4_2 \cup 4_1\}$
12- $x <<< 5$	26- $\{2,0,4,7,3,1,6,5\}$	42- $\{-,-,0,1,-,-,4,5\}$	58- $\{1,1,1,0,0,1,0,1\}$	75- $\{5_6 \cup 3_2\}$	92- $\{4_2 \cup 4_2\}$
13- $x <<< 6$	27- $\{5,7,2,6,1,3,4,0\}$	43- $\{4,-,-,7,-,1,2,-\}$	59- $\{1,0,1,1,1,0,0,0\}$	76- $\{3_1 \cup 5_1\}$	93- $\{4_2 \cup 4_3\}$
14- $x <<< 7$	28- $\{6,5,0,2,7,4,1,3\}$	44- $\{4,5,6,7,-,-,-,-\}$	60- $\{0,1,0,1,0,1,0,1\}$	77- $\{3_1 \cup 5_2\}$	94- $\{4_3 \cup 4_1\}$
	29- $\{7,3,4,5,1,2,0,6\}$	45- $\{-,7,4,-,-,3,0,-\}$	61- $\{1,0,1,0,1,0,1,0\}$	78- $\{3_1 \cup 5_3\}$	95- $\{4_3 \cup 4_2\}$
	30- $\{4,2,6,7,0,3,5,1\}$	46- $\{2,3,-,-,-,-,4,5\}$	62- $\{0,0,1,1,0,0,1,1\}$	79- $\{3_1 \cup 5_4\}$	96- $\{4_3 \cup 4_3\}$
			63- $\{1,1,0,0,1,1,0,0\}$	80- $\{3_1 \cup 5_5\}$	97- $\{\ 8_1\ \}$

A função $x <<< n$ significa uma rotação cíclica de n bites para a esquerda sobre o vetor x, e $x >>> n$ uma rotação cíclica de n bites para a direita sobre o vetor x.

A permutação entre bites, como por exemplo $\{1_0, 2_1, 3_2, 0_3, 5_4, 6_5, 7_6, 4_7\}$, significa que o bite de entrada na posição x será mapeado para a posição y, essa representação é dada da seguinte forma: x_y. Como a sequência de saída sempre será em ordem crescente de 0 à 7, o subíndice y é omitida da TAB 5.1.

Para a função XOR entre bites, considerando, por exemplo, a transformação $\{-, -, -, -, 0, 1, 2, 3\}$, os bites representados pelo símbolo $-$ não sofrerão alteração, ou seja, $y_i = x_i$. Enquanto que os bites da posição que possuem números, indicam que estes bites sofrerão manipulações. Para este exemplo foram realizadas as seguintes operações: $y_4 = x_0 \oplus x_4$; $y_5 = x_1 \oplus x_5$; $y_6 = x_2 \oplus x_6$ e $y_7 = x_3 \oplus x_7$.

O procedimento que realiza a operação XOR entre constantes é similar ao anterior, o que os diferem, é que neste todos os bites serão manipulados. Considerando como hipótese a constante $\{0, 0, 0, 1, 1, 0, 1, 1\}$, o procedimento seria realizado da seguinte forma: $y_0 = x_0 \oplus 0$; $y_1 = x_1 \oplus 0$; $y_2 = x_2 \oplus 0$; $y_3 = x_3 \oplus 1$, $y_4 = x_4 \oplus 1$; $y_5 = x_5 \oplus 0$, $y_6 = x_6 \oplus 1$ e $y_7 = x_7 \oplus 1$.

Para o mapeamento inverso em corpos finitos foram utilizados os seguintes polinômios: $\{3_1 = x^3 + x + 1\}$, $\{3_2 = x^3 + x^2 + 1\}$, $\{4_1 = x^4 + x + 1\}$, $\{4_2 = x^4 + x^3 + 1\}$, $\{4_3 = x^4 + x^3 + x^2 + x + 1\}$, $\{5_1 = x^5 + x^2 + 1\}$, $\{5_2 = x^5 + x^3 + 1\}$, $\{5_3 = x^5 + x^3 + x^2 + 1\}$, $\{5_4 = x^5 + x^4 + x^2 + x + 1\}$, $\{5_5 = x^5 + x^4 + x^3 + x + 1\}$, $\{5_6 = x^5 + x^4 + x^3 + x^2 + 1\}$ e $\{8_1 = x^8 + x^4 + x^3 + x + 1\}$.

Esses polinômios foram utilizados de tal forma que a concatenação entre eles fosse igual a oito bites. Portanto, foram feitos arranjos entre polinômios de grau 4 e arranjos entre polinômios de grau 3 e 5.

5.4.1 DISTRIBUIÇÃO DAS FUNÇÕES SOBRE O GRAFO

Como observado, foram elaboradas 97 funções para a composição dos 128 vértices do grafo. Para distribuição dessas funções, os 128 vértices foram divididos em 8 grupos de 16 vértices. Dessa forma, para conseguir uma distribuição balanceada, foi considerado o sexto grupo de funções contendo 32 repetições da transformação $\{8_1\}$. Logo, o número de funções por categoria fica redistribuído conforme a TAB 5.2.

Assim sendo, as distribuições das funções por categorias foram realizadas conforme ilustrado na TAB 5.3.

O objetivo desta distribuição é manter a proporção das funções para cada grupo. As categorias com 16 funções foram distribuídas mantendo 2 funções por grupo, enquanto

TAB. 5.2: Número de funções por categoria

Categorias	Sigla	Número de funções
Rotação Cíclica	RC	14
Permutação entre bites	PB	16
XOR entre bites	XB	16
XOR com constantes	XC	17
Inverso em $GF(2^n)$	IN	33
Inverso em $GF(2^8)$	I8	32
		128

TAB. 5.3: Distribuição das categorias sobre o grafo.

$v_0 = I8$	$v_{16} = RC$	$v_{32} = XC$	$v_{48} = IN$	$v_{64} = RC$	$v_{80} = IN$	$v_{96} = RC$	$v_{112} = RC$
$v_1 = PB$	$v_{17} = I8$	$v_{33} = IN$	$v_{49} = I8$	$v_{65} = IN$	$v_{81} = I8$	$v_{97} = IN$	$v_{113} = IN$
$v_2 = XC$	$v_{18} = IN$	$v_{34} = XB$	$v_{50} = XB$	$v_{66} = PB$	$v_{82} = XC$	$v_{98} = I8$	$v_{114} = I8$
$v_3 = IN$	$v_{19} = XC$	$v_{35} = I8$	$v_{51} = RC$	$v_{67} = I8$	$v_{83} = PB$	$v_{99} = XB$	$v_{115} = PB$
$v_4 = IN$	$v_{20} = PB$	$v_{36} = RC$	$v_{52} = IN$	$v_{68} = XB$	$v_{84} = RC$	$v_{100} = IN$	$v_{116} = XC$
$v_5 = RC$	$v_{21} = XB$	$v_{37} = XC$	$v_{53} = I8$	$v_{69} = IN$	$v_{85} = IN$	$v_{101} = PB$	$v_{117} = XB$
$v_6 = XB$	$v_{22} = I8$	$v_{38} = IN$	$v_{54} = PB$	$v_{70} = XC$	$v_{86} = I8$	$v_{102} = XC$	$v_{118} = I8$
$v_7 = I8$	$v_{23} = PB$	$v_{39} = I8$	$v_{55} = IN$	$v_{71} = I8$	$v_{87} = XB$	$v_{103} = I8$	$v_{119} = IN$
$v_8 = XC$	$v_{24} = IN$	$v_{40} = IN$	$v_{56} = XC$	$v_{72} = PB$	$v_{88} = IN$	$v_{104} = RC$	$v_{120} = XB$
$v_9 = IN$	$v_{25} = RC$	$v_{41} = PB$	$v_{57} = I8$	$v_{73} = IN$	$v_{89} = RC$	$v_{105} = IN$	$v_{121} = IN$
$v_{10} = I8$	$v_{26} = I8$	$v_{42} = RC$	$v_{58} = PB$	$v_{74} = I8$	$v_{90} = I8$	$v_{106} = XC$	$v_{122} = I8$
$v_{11} = IN$	$v_{27} = XB$	$v_{43} = I8$	$v_{59} = RC$	$v_{75} = XB$	$v_{91} = XB$	$v_{107} = PB$	$v_{123} = IN$
$v_{12} = PB$	$v_{28} = IN$	$v_{44} = XB$	$v_{60} = IN$	$v_{76} = IN$	$v_{92} = PB$	$v_{108} = I8$	$v_{124} = XC$
$v_{13} = I8$	$v_{29} = XC$	$v_{45} = IN$	$v_{61} = XB$	$v_{77} = RC$	$v_{93} = XC$	$v_{109} = XB$	$v_{125} = PB$
$v_{14} = XB$	$v_{30} = IN$	$v_{46} = I8$	$v_{62} = XC$	$v_{78} = I8$	$v_{94} = IN$	$v_{110} = IN$	$v_{126} = I8$
$v_{15} = XC$	$v_{31} = I8$	$v_{47} = PB$	$v_{63} = I8$	$v_{79} = XC$	$v_{95} = I8$	$v_{111} = I8$	$v_{127} = IN$

que as categorias com 32 funções foram distribuídas mantendo 4 funções por grupos. Nesse diapasão, cada grupo possui um número específico de funções conforme ilustrado na TAB 5.4.

TAB. 5.4: Distribuição genérica.

Categoria	N^o de funções por grupo
RC	2
PB	2
XB	2
XC	2
IN	4
$I8$	4
Total	16

Não obstante, a categoria RC possui apenas 14 funções, por este motivo foi acrescentada mais uma função nas categorias XC e IN. Assim, o primeiro e o último grupo possuem um comportamento diferente do apresentado na TAB 5.4, pois estes dois grupos possuem apenas uma função da categoria RC, na qual as duas funções extras são responsáveis em compensar essa carência. Logo, o primeiro grupo possui 3 funções da categoria XC e o último grupo 5 funções da categoria IN.

A TAB 5.5 ilustra o mapeamento das funções para cada vértice, tendo como referência as TABs 5.1 e 5.3.

5.4.2 LEITURA DO GRAFO

Como mencionado, a caixa-S proposta é dependente da chave secreta, chave esta que tem a função de determinar os vértices que participarão do processo de transformação dos

TAB. 5.5: Distribuição das funções sobre o grafo.

$v_0 = f_{97}$	$v_{16} = f_7$	$v_{32} = f_{61}$	$v_{48} = f_{95}$	$v_{64} = f_{10}$	$v_{80} = f_{84}$	$v_{96} = f_2$	$v_{112} = f_6$
$v_1 = f_{15}$	$v_{17} = f_{97}$	$v_{33} = f_{96}$	$v_{49} = f_{97}$	$v_{65} = f_{75}$	$v_{81} = f_{97}$	$v_{97} = f_{79}$	$v_{113} = f_{80}$
$v_2 = f_{47}$	$v_{18} = f_{78}$	$v_{34} = f_{35}$	$v_{50} = f_{37}$	$v_{66} = f_{22}$	$v_{82} = f_{63}$	$v_{98} = f_{97}$	$v_{114} = f_{97}$
$v_3 = f_{64}$	$v_{19} = f_{62}$	$v_{35} = f_{97}$	$v_{51} = f_4$	$v_{67} = f_{97}$	$v_{83} = f_{23}$	$v_{99} = f_{43}$	$v_{115} = f_{25}$
$v_4 = f_{67}$	$v_{20} = f_{21}$	$v_{36} = f_9$	$v_{52} = f_{70}$	$v_{68} = f_{39}$	$v_{84} = f_3$	$v_{100} = f_{88}$	$v_{116} = f_{48}$
$v_5 = f_8$	$v_{21} = f_{33}$	$v_{37} = f_{51}$	$v_{53} = f_{97}$	$v_{69} = f_{90}$	$v_{85} = f_{94}$	$v_{101} = f_{18}$	$v_{117} = f_{45}$
$v_6 = f_{31}$	$v_{22} = f_{97}$	$v_{38} = f_{77}$	$v_{54} = f_{27}$	$v_{70} = f_{49}$	$v_{86} = f_{97}$	$v_{102} = f_{55}$	$v_{118} = f_{97}$
$v_7 = f_{97}$	$v_{23} = f_{19}$	$v_{39} = f_{97}$	$v_{55} = f_{65}$	$v_{71} = f_{97}$	$v_{87} = f_{41}$	$v_{103} = f_{97}$	$v_{119} = f_{92}$
$v_8 = f_{52}$	$v_{24} = f_{93}$	$v_{40} = f_{74}$	$v_{56} = f_{53}$	$v_{72} = f_{28}$	$v_{88} = f_{66}$	$v_{104} = f_{13}$	$v_{120} = f_{46}$
$v_9 = f_{81}$	$v_{25} = f_{11}$	$v_{41} = f_{17}$	$v_{57} = f_{97}$	$v_{73} = f_{86}$	$v_{89} = f_1$	$v_{105} = f_{68}$	$v_{121} = f_{82}$
$v_{10} = f_{97}$	$v_{26} = f_{97}$	$v_{42} = f_{12}$	$v_{58} = f_{20}$	$v_{74} = f_{97}$	$v_{90} = f_{97}$	$v_{106} = f_{54}$	$v_{122} = f_{97}$
$v_{11} = f_{91}$	$v_{27} = f_{34}$	$v_{43} = f_{97}$	$v_{59} = f_5$	$v_{75} = f_{40}$	$v_{91} = f_{42}$	$v_{107} = f_{16}$	$v_{123} = f_{83}$
$v_{12} = f_{24}$	$v_{28} = f_{72}$	$v_{44} = f_{36}$	$v_{60} = f_{89}$	$v_{76} = f_{76}$	$v_{92} = f_{26}$	$v_{108} = f_{97}$	$v_{124} = f_{60}$
$v_{13} = f_{97}$	$v_{29} = f_{59}$	$v_{45} = f_{87}$	$v_{61} = f_{38}$	$v_{77} = f_{14}$	$v_{93} = f_{57}$	$v_{109} = f_{44}$	$v_{125} = f_{29}$
$v_{14} = f_{32}$	$v_{30} = f_{71}$	$v_{46} = f_{97}$	$v_{62} = f_{50}$	$v_{78} = f_{97}$	$v_{94} = f_{73}$	$v_{110} = f_{85}$	$v_{126} = f_{97}$
$v_{15} = f_{56}$	$v_{31} = f_{97}$	$v_{47} = f_{30}$	$v_{63} = f_{97}$	$v_{79} = f_{58}$	$v_{95} = f_{97}$	$v_{111} = f_{97}$	$v_{127} = f_{69}$

bites.

Considerando uma chave representada por um vetor de 128 bites, as posições do vetor que contém bites iguais a um, indicarão os vértices que participarão do processo de transformação dos bites. Como exemplo considere a seguinte chave (vide TAB 5.6):

TAB. 5.6: Exemplo de chave de 128 bites

1_0	1_{16}	1_{32}	1_{48}	0_{64}	0_{80}	0_{96}	1_{112}
0_1	0_{17}	1_{33}	0_{49}	1_{65}	0_{81}	1_{97}	1_{113}
0_2	1_{18}	0_{34}	0_{50}	0_{66}	1_{82}	0_{98}	0_{114}
1_3	1_{19}	1_{35}	1_{51}	0_{67}	1_{83}	1_{99}	1_{115}
1_4	1_{20}	1_{36}	0_{52}	0_{68}	0_{84}	0_{100}	0_{116}
0_5	0_{21}	1_{37}	1_{53}	0_{69}	0_{85}	1_{101}	1_{117}
0_6	0_{22}	1_{38}	1_{54}	0_{70}	1_{86}	0_{102}	0_{118}
0_7	1_{23}	1_{39}	1_{55}	0_{71}	1_{87}	1_{103}	1_{119}
1_8	0_{24}	0_{40}	1_{56}	0_{72}	1_{88}	0_{104}	0_{120}
1_9	0_{25}	0_{41}	1_{57}	0_{73}	0_{89}	0_{105}	0_{121}
1_{10}	1_{26}	0_{42}	1_{58}	1_{74}	0_{90}	0_{106}	0_{122}
1_{11}	1_{27}	1_{43}	1_{59}	0_{75}	0_{91}	0_{107}	0_{123}
0_{12}	0_{28}	0_{44}	0_{60}	0_{76}	0_{92}	1_{108}	0_{124}
0_{13}	1_{29}	1_{45}	0_{61}	1_{77}	0_{93}	1_{109}	1_{125}
0_{14}	1_{30}	0_{46}	1_{62}	0_{78}	1_{94}	0_{110}	1_{126}
1_{15}	0_{31}	1_{47}	0_{63}	0_{79}	0_{95}	0_{111}	0_{127}

Assim, os vértices utilizados para a transformação dos bites estão ilustrados em azul na TAB 5.7.

5.4.3 COMENTÁRIOS DA CAIXA-S

Para construir caixas-S baseadas no modelo apresentado, dois pontos são fundamentais. O primeiro é a escolha das função utilizadas para a composição dos vértices. E o segundo é a distribuição dessas funções sobre os vértices.

TAB. 5.7: Funções utilizadas considerando a chave da TAB 5.6.

$v_0 = f_{97}$	$v_{16} = f_7$	$v_{32} = f_{61}$	$v_{48} = f_{95}$	$v_{64} = f_{10}$	$v_{80} = f_{84}$	$v_{96} = f_2$	$v_{112} = f_6$
$v_1 = f_{15}$	$v_{17} = f_{97}$	$v_{33} = f_{96}$	$v_{49} = f_{97}$	$v_{65} = f_{75}$	$v_{81} = f_{97}$	$v_{97} = f_{79}$	$v_{113} = f_{80}$
$v_2 = f_{47}$	$v_{18} = f_{78}$	$v_{34} = f_{35}$	$v_{50} = f_{37}$	$v_{66} = f_{22}$	$v_{82} = f_{63}$	$v_{98} = f_{97}$	$v_{114} = f_{97}$
$v_3 = f_{64}$	$v_{19} = f_{62}$	$v_{35} = f_{97}$	$v_{51} = f_4$	$v_{67} = f_{97}$	$v_{83} = f_{23}$	$v_{99} = f_{43}$	$v_{115} = f_{25}$
$v_4 = f_{67}$	$v_{20} = f_{21}$	$v_{36} = f_9$	$v_{52} = f_{70}$	$v_{68} = f_{39}$	$v_{84} = f_3$	$v_{100} = f_{88}$	$v_{116} = f_{48}$
$v_5 = f_8$	$v_{21} = f_{33}$	$v_{37} = f_{51}$	$v_{53} = f_{97}$	$v_{69} = f_{90}$	$v_{85} = f_{94}$	$v_{101} = f_{18}$	$v_{117} = f_{45}$
$v_6 = f_{31}$	$v_{22} = f_{97}$	$v_{38} = f_{77}$	$v_{54} = f_{27}$	$v_{70} = f_{49}$	$v_{86} = f_{97}$	$v_{102} = f_{55}$	$v_{118} = f_{97}$
$v_7 = f_{97}$	$v_{23} = f_{19}$	$v_{39} = f_{97}$	$v_{55} = f_{65}$	$v_{71} = f_{97}$	$v_{87} = f_{41}$	$v_{103} = f_{97}$	$v_{119} = f_{92}$
$v_8 = f_{52}$	$v_{24} = f_{93}$	$v_{40} = f_{74}$	$v_{56} = f_{53}$	$v_{72} = f_{28}$	$v_{88} = f_{66}$	$v_{104} = f_{13}$	$v_{120} = f_{46}$
$v_9 = f_{81}$	$v_{25} = f_{11}$	$v_{41} = f_{17}$	$v_{57} = f_{97}$	$v_{73} = f_{86}$	$v_{89} = f_1$	$v_{105} = f_{68}$	$v_{121} = f_{82}$
$v_{10} = f_{97}$	$v_{26} = f_{97}$	$v_{42} = f_{12}$	$v_{58} = f_{20}$	$v_{74} = f_{97}$	$v_{90} = f_{97}$	$v_{106} = f_{54}$	$v_{122} = f_{97}$
$v_{11} = f_{91}$	$v_{27} = f_{34}$	$v_{43} = f_{97}$	$v_{59} = f_5$	$v_{75} = f_{40}$	$v_{91} = f_{42}$	$v_{107} = f_{16}$	$v_{123} = f_{83}$
$v_{12} = f_{24}$	$v_{28} = f_{72}$	$v_{44} = f_{36}$	$v_{60} = f_{89}$	$v_{76} = f_{76}$	$v_{92} = f_{26}$	$v_{108} = f_{97}$	$v_{124} = f_{60}$
$v_{13} = f_{97}$	$v_{29} = f_{59}$	$v_{45} = f_{87}$	$v_{61} = f_{38}$	$v_{77} = f_{14}$	$v_{93} = f_{57}$	$v_{109} = f_{44}$	$v_{125} = f_{29}$
$v_{14} = f_{32}$	$v_{30} = f_{71}$	$v_{46} = f_{97}$	$v_{62} = f_{50}$	$v_{78} = f_{97}$	$v_{94} = f_{73}$	$v_{110} = f_{85}$	$v_{126} = f_{97}$
$v_{15} = f_{56}$	$v_{31} = f_{97}$	$v_{47} = f_{30}$	$v_{63} = f_{97}$	$v_{79} = f_{58}$	$v_{95} = f_{97}$	$v_{111} = f_{97}$	$v_{127} = f_{69}$

Quanto à escolha das funções, é importante garantir uma boa variedade, pois quanto maior o número de funções mais difícil será para o inimigo mapear os possíveis caminhos aleatórios equivalentes. Caso contrário, se o número de funções for reduzido, a probabilidade de possíveis caminhos aleatórios equivalentes será alta e desta forma o custo computacional para o ataque de força bruta será reduzido.

Quanto à distribuição das funções, é importante garantir que as funções estejam bem distribuídas sobre os vértices de tal forma que a chave consiga selecionar os vértices que não se anulem. Caso essa distribuição não seja bem planejada, funções que trabalhem de forma semelhante ou até mesmo que se anulem podem gerar pontos fixos.

Nesse contexto, pode-se perceber dois níveis de segurança. O primeiro nível de segurança está relacionado ao modelo da caixa-S, enquanto o inimigo não conseguir reduzir e encontrar possíveis caminhos aleatórios equivalentes de tal forma que seja viável em tempo computacional, a segurança da caixa-S está garantida. Já o segundo nível de segurança está relacionado à escolha e à distribuição das funções, uma vez que o inimigo consiga mapear a caixa-S utilizada, é importante garantir que as funções gerem transformações com alta aleatoriedade e não linearidade, garantindo dessa forma a resistência da caixa-S contra ataques diferencias e lineares.

5.5 AVALIAÇÃO DA CAIXA-S

Esta seção tem como objetivo avaliar o segundo nível de segurança da caixa-S, ou seja, supondo que um inimigo consiga mapear a caixa-S utilizada, ainda assim qual seria a resistência da caixa-S?

Para responder esta pergunta, a caixa-S construída na seção 5.4 será submetida à

avaliação das métricas apresentadas no capítulo 3.

5.5.1 COLETA DA AMOSTRA DE CHAVES

O universo de chaves utilizado neste trabalho é igual a 2^{128}, portanto, não há recurso computacional viável para a análise das caixas-S de todas as possíveis chaves. Logo, houve a necessidade de se trabalhar com uma amostra.

Foram estabelecidas duas condições para a seleção da amostra de chaves. A primeira condição era que o gerador de chaves tinha que passar nos testes estatísticos propostos pelo NIST (RUKHIN, 2010). E a segunda, que a distribuição da amostra fosse caracterizada conforme a distribuição do universo de chaves apresentada na FIG 5.2.

O gerador utilizado para avaliação foi *SecureRandom*[2] disponível no pacote *security* da linguagem de programação Java. Ele foi escolhido pois o mesmo foi desenvolvido com base na FIPS 140-2, passando maior credibilidade para sua utilização.

No processo de avaliação, o primeiro requisito foi satisfeito em aproximadamente 70% dos casos. A TAB 5.8 ilustra os parâmetros de configuração utilizados para a avaliação da amostra de chaves. Em que n representa o número de bites do vetor de entrada. Os demais parâmetros possuem significados diferentes em cada teste, então, recomenda-se para um melhor entendimento de cada parâmetro e até mesmo do objetivo e sistematização de cada teste, que consulte a referência (RUKHIN, 2001).

Para atender o segundo requisito, foi necessário coletar uma amostra de 10^6 chaves. A FIG 5.3 ilustra a distribuição obtida.

5.5.2 PROCESSO DE AVALIAÇÃO

As métricas apresentadas no capítulo 3 foram utilizadas no processo de avaliação da caixa-S, e como já mencionado, o objetivo dessa avaliação é verificar o segundo nível de segurança da caixa-S, ou seja, partindo da hipótese de que o inimigo conheça a função de transformação, as métricas tem como objetivo verificar a resistência dessa função em relação aos ataques linear e diferencial.

[2] http://download.oracle.com/javase/6/docs/api/java/security/SecureRandom.html

TAB. 5.8: Configuração utilizada no conjunto de testes

Testes	Dados de entrada
1 The Frequency (Monobit) Test	$n = 128$
2 Frequency Test within a Block	$n = 128$, $M = 20$
3 The Runs Test	$n = 128$
4 Tests for the Longest-Run-of-Ones in a Block	$n = 128$
5 The Binary Matrix Rank Test	$n = 10^6$, $M = 8$, $Q = 8$
6 The Discrete Fourier Transform (Spectral) Test	$n = 128$
7 The Non-overlapping Template Matching Test	$n = 10^6$, $N = 8$, $B =$ "101010101"
8 The Overlapping Template Matching Test	$n = 10^6$, $N = 8$, $B =$ "101010101"
9 Maurer's "Universal Statistical"Test	$n = 10^6$, $L = 10$, $Q = 2048$
10 The Linear Complexity Test	$n = 10^6$, $M = 5000$
11 The Serial Test	$n = 10^6$, $m = 16$
12 The Approximate Entropy Test	$n = 10^6$, $m = 16$
13 The Cumulative Sums (Cusums) Test	$n = 128$
14 The Random Excursions Test	$n = 10^6$
15 The Random Excursions Variant Test	$n = 10^6$

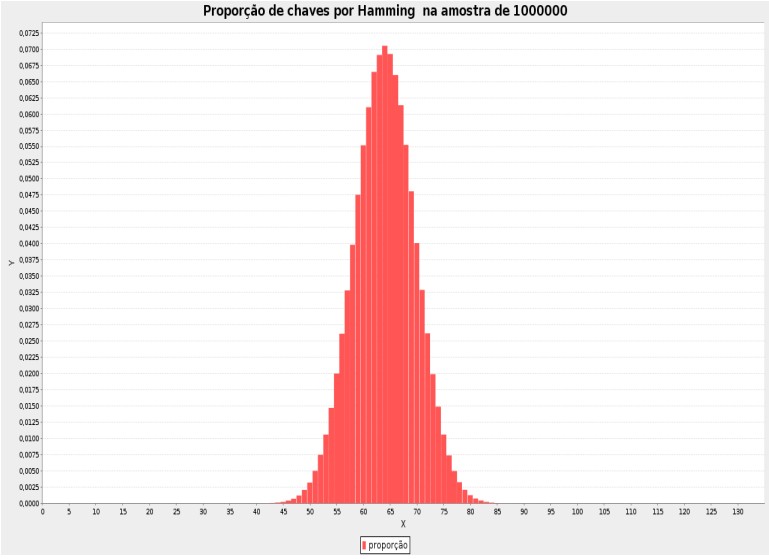

FIG. 5.3: Distribuição binomial da amostra 10^6 chaves $\mu = 64$ e $\sigma = \sqrt{32}$

5.5.2.1 AVALIAÇÃO DA NÃO LINEARIDADE

Essa métrica como apresentada no capítulo 3 foi especificada para funções booleanas na forma $f(x) : V_2^n \rightarrow V_2$, entretanto, a função de transformação da caixa-S é da forma $f(x) : V_2^n \rightarrow V_2^n$, logo, para avaliar essa função, é necessário considerar a menor distância de um dos vetores para o conjunto das funções afins. Dessa forma, o processo de avaliação é similar ao apresentado no capítulo 3, sendo a única diferença o número de vetores avaliados

para cada entrada da função.

Na avaliação da caixa-S foram utilizadas as 10^6 chaves da amostra. Para cada chave foi verificado a não linearidade da função, a FIG 5.4 ilustra a não linearidade das funções, o eixo x representa a não linearidade e o eixo y o número de funções com a não linearidade x.

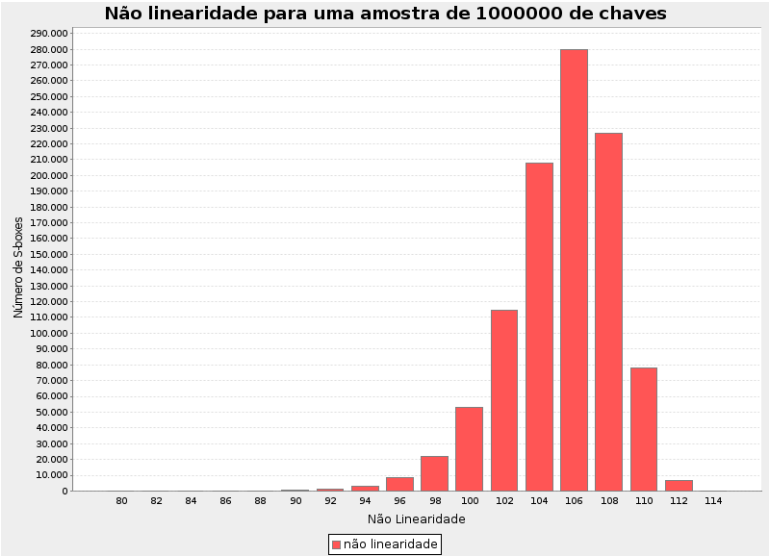

FIG. 5.4: Não linearidade das caixas-S da amostra de chaves coletadas

A caixa-S original do AES possui uma não linearidade igual a 112. Realizando uma comparação, pode-se perceber que a maioria das caixas-S analisadas possuem não linearidade igual a 106. Sendo assim, a não linearidade da composição de funções resultantes dos diversos caminhos aleatórios pode ser considerada satisfatória.

Quanto aos demais valores, pode-se observar que certas combinações de funções atingem uma não linearidade maior do que a obtida pela caixa-S original do AES, contudo, o inverso também é evidente, existem certas composições que resultam em caixas-S com não linearidade igual a 80 por exemplo. Entretanto, essa fraqueza pode ser suprida pela escolha das funções que compõe os vértices. Pois dependendo das funções utilizadas para a composição dos vértices, pode ser que os valores obtidos sejam melhores do que os apresentados pelas funções escolhidas neste trabalho, uma vez que o critério de escolha das funções booleanas que compõe o grafo, não foi o principal foco deste trabalho.

5.5.2.2 CRITÉRIO DE ESTREITO AVALANCHE (SAC)

Como mencionado no capítulo 3, esta métrica é considerada uma combinação das métricas de completude e efeito avalanche, logo, apenas o SAC foi utilizada para medir a aleatoriedade dos bites de saída em função de pequenas variações dos bites de entrada.

Para a realização do teste foram consideradas todas as possíveis 256 entradas de oito bites, e cada entrada foi comparada com outras oito entradas em que apenas a posição i foi complementada. Então, para cada entrada x e x_i foram obtidas as saídas $y = f(x)$ e $y' = f(x_i)$, a partir dessa saída foi obtido o vetor avalanche $\Delta_y = y \oplus y_i$. Para cada delta foi verificado o nível de aleatoriedade, ou seja, a proporção de zeros e uns. Após obter todos os vetores avalanches, verificou-se que na média o nível de aleatoriedade é próximo de 0.5, demonstrando que as caixas-S possuem um bom nível de aleatoriedade. A FIG 5.5 ilustra as medidas de SAC obtidas pela amostra de chaves.

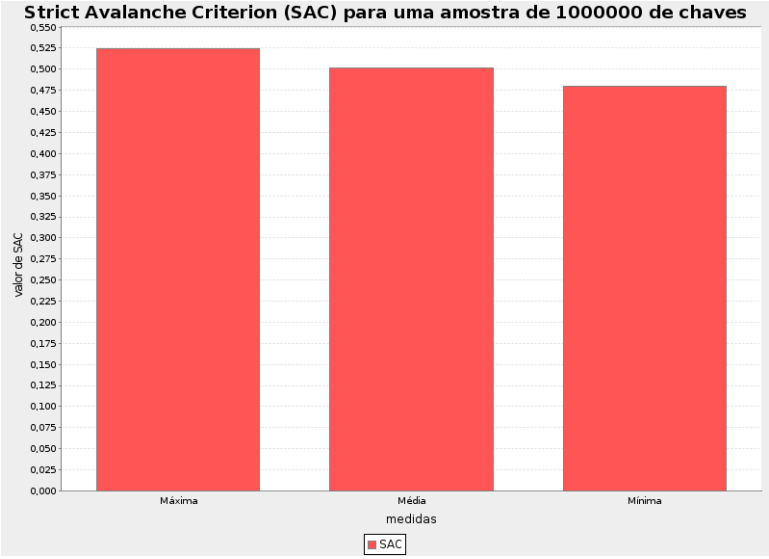

FIG. 5.5: Medidas de SAC das caixas-S

5.5.2.3 AVALIAÇÃO DA CORRELAÇÃO

O objetivo desta métrica é avaliar a independência entre os vetores avalanches. Para medir a correlação, o mesmo processo de coleta do SAC foi utilizado. Foram consideradas todas as possíveis 256 entradas de oito bites, e cada entrada foi comparada com outras oito entradas em que apenas a posição i foi complementada. Então, para cada entrada x e x_i foram obtidas as saídas $y = f(x)$ e $y' = f(x_i)$, a partir dessa saída foi obtido o vetor

avalanche $\Delta_y = y \oplus y_i$. A partir dai, foi verificado a correlação entre todos os pares de vetores, totalizando 32896 comparações.

O valor médio da correlação entre os vetores para toda a amostra de caixas-S analisadas foi de 0.3. Essa medida demonstra um bom nível de independência entre os vetores, uma vez que o valor zero indica a total independência entre os vetores. Conseguir produzir vetores independentes significa obter maior resistência contra ataques que visam mapear diferentes saídas por dedução de mapeamentos previamente conhecidos, ou seja, quanto menor a correlação entre vetores avalanches mais difícil será para o inimigo realizar ataques diferenciais.

5.5.2.4 AVALIAÇÃO DA ANÁLISE DA FREQUÊNCIA DE VÁRIOS PESOS DE HAMMING

Esta métrica é uma variação do SAC e por este motivo também avalia a aleatoriedade dos bites de saída. Para verificar essa característica, é observado se a distribuição dos pesos de hamming dos bites de saída produz uma distribuição binomial.

Da mesma forma que nas métricas anteriores, para a avaliação dessa métricas, foram consideradas todas as possíveis 256 entradas de oito bites, e cada entrada foi comparada com outras oito entradas em que apenas a posição i foi complementada. Então, para cada entrada x e x_i foram obtidas as saídas $y = f(x)$ e $y' = f(x_i)$, a partir dessa saída foi obtido o vetor avalanche $\Delta_y = y \oplus y_i$. Sendo assim, foi calculado o peso de hamming de cada vetor conforme ilustrado na FIG 5.6.

A FIG 5.7 ilustra a avaliação dessa métrica aplicada na caixa-S original do AES. Se realizada uma comparação, pode-se perceber que, na média, as caixas-S avaliadas possuem características semelhantes à da caixa-S original.

O resultado dessa métrica sendo satisfatório, reforça a convicção de que as caixas-S possuem um bom nível de aleatoriedades mesmo quando submetidas a pequenas variações nos bites de entrada.

5.5.2.5 AVALIAÇÃO DA ANÁLISE DA FREQUÊNCIA DE VÁRIOS VALORES DIFERENCIAIS

Esta métrica tem como objetivo analisar a uniformidade dos vetores avalanches, ou seja, verificar se a frequência dos vetores avalanches é caracterizada por uma distribuição uniforme. No processo de coleta de dados, foram consideradas todas as possíveis 256 entradas

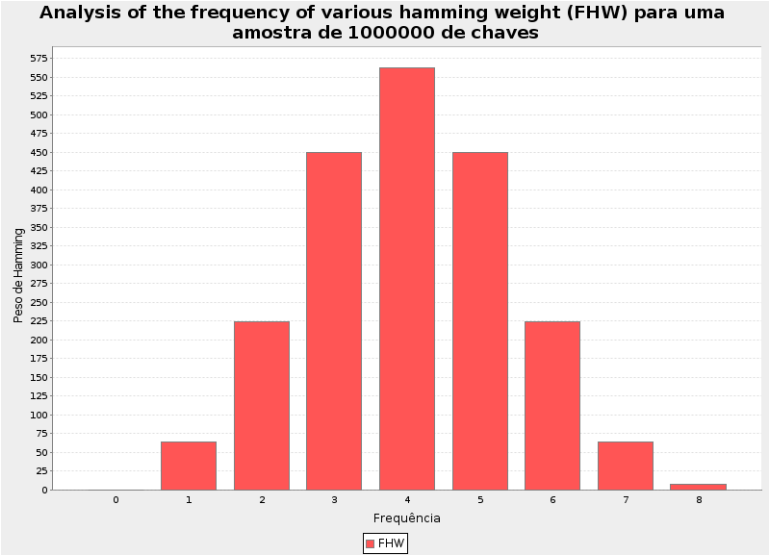

FIG. 5.6: Distribuição de hamming da amostra de caixas-S

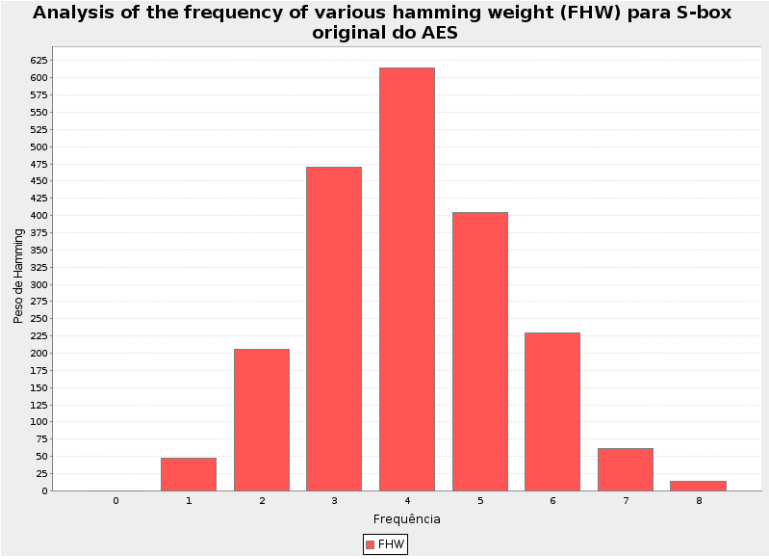

FIG. 5.7: Distribuição de hamming da amostra de caixas-S

de oito bites, e cada entrada foi comparada com outras oito entradas em que apenas a posição i foi complementada. Então, para cada entrada x e x_i foram obtidas as saídas $y = f(x)$ e $y' = f(x_i)$, a partir dessa saída foi obtido o vetor avalanche $\Delta_y = y \oplus y_i$. Para cada delta foi realizada a contagem da frequência. A FIG 5.8 ilustra a distribuição da frequência dos vetores avalanches.

Pela observação da FIG 5.8, pode-se observar que a amostra da caixa-S atende ao requisito esperado, logo, pode-se concluir que as caixas-S avaliadas possuem um bom nível de aleatoriedade e também uma boa distribuição quanto a frequência de seus vetores

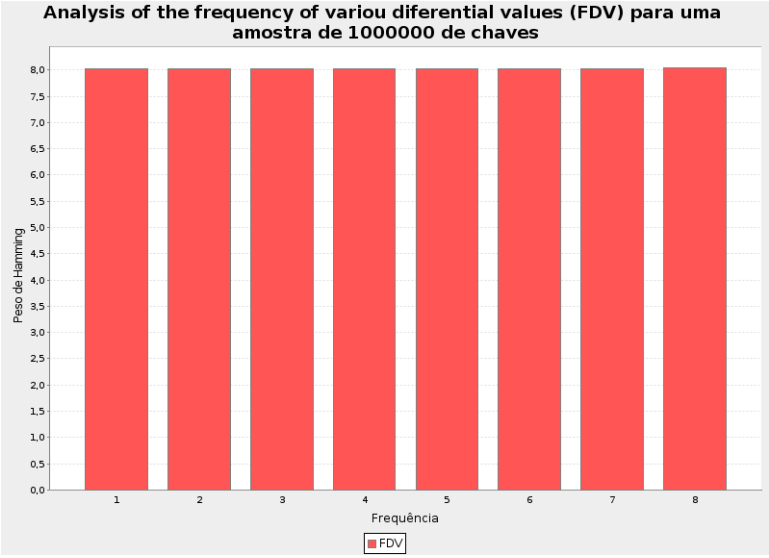

FIG. 5.8: Distribuição da frequência dos vetores avalanches.

avalanches. De acordo com (MAR, 2008), quando a avaliação dessa métrica é satisfatória, significa que as caixas-S possuem boas propriedades de completude.

5.5.2.6 AVALIAÇÃO DA ANÁLISE DOS PESOS DE HAMMING DE ACORDO COM A POSIÇÃO DO BITE

Esta métrica tem com objetivo verificar se os índices dos vetores avalanches geram uma distribuição uniforme. Caso essa característica seja observada, fica caracterizado que as caixas-S possuem um bom nível de segurança, caso contrário, se a os valores do eixo X apresentarem variações no eixo Y, significa que as caixas-S podem apresentar fraquezas e indícios de padrões que poderão ser explorados na elaboração de ataques. Caso o valor Y seja muito baixo para um determinado valor de X, significa que a posição X dos bites dos vetores avalanche são iguais com muita frequência, e caso o valor Y seja muito alto, significa que os bites são complementares na maioria dos casos. Dessa forma, em um ataque diferencial por exemplo, para uma determinada posição X seria mais fácil prever o valor esperado.

Portanto, para avaliar essa característica, foram consideradas todas as possíveis 256 entradas de oito bites, e cada entrada foi comparada com outras oito entradas em que apenas a posição i foi complementada. Então, para cada entrada x e x_i foram obtidas as saídas $y = f(x)$ e $y' = f(x_i)$, a partir dessa saída foi obtido o vetor avalanche $\Delta_y = y \oplus y_i$.

Considerando o conjunto de todos os deltas, foi construída uma matriz M^{nxm}, em que

n indica o número de deltas, ou seja, o número de linhas da matriz, e m o número de colunas. A partir da matriz M^{nxm}, foram computadas as frequências de cada coluna, e verificado se essas frequências eram caracterizadas por uma distribuição uniforme. A FIG 5.9 ilustra a distribuição da amostra de caixas-S avaliadas, e se realizada uma comparação com a caixas-S original do AES (vide FIG 5.10), pode-se observar que a distribuição da caixa-S do AES possui algumas variações em Y.

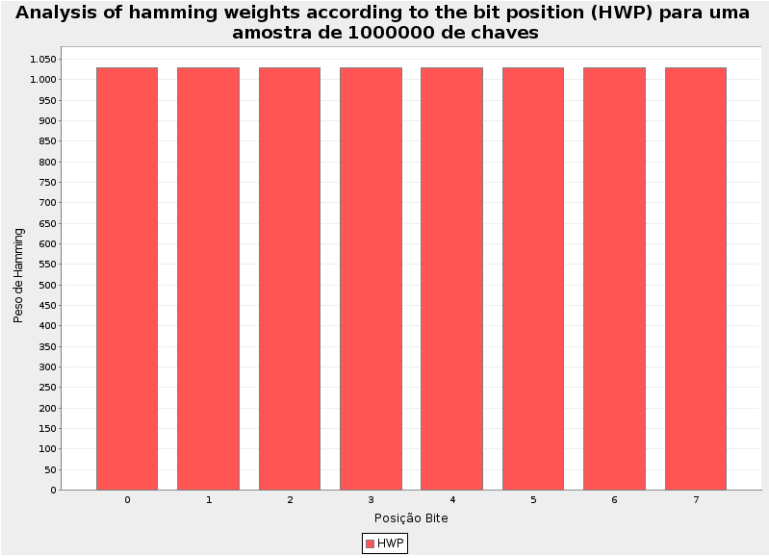

FIG. 5.9: Distribuição da frequência dos vetores avalanches por índice dos bites.

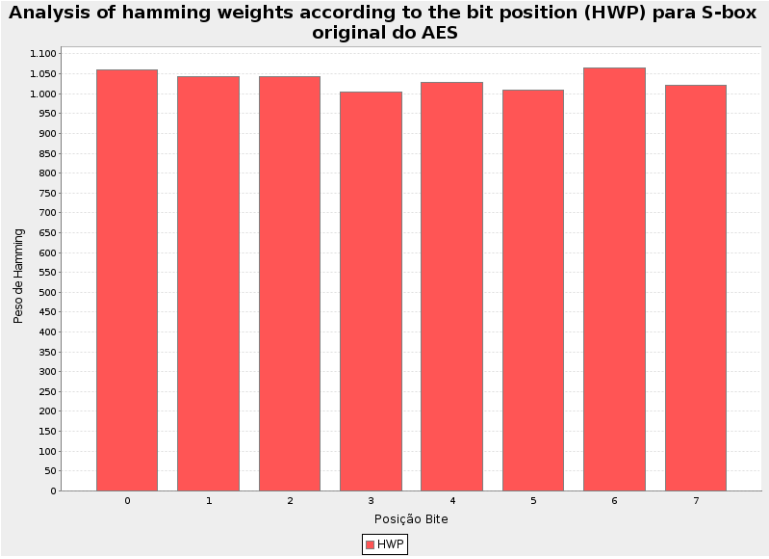

FIG. 5.10: Distribuição da frequência dos vetores avalanches por índice dos bites (AES).

5.6 CONCLUSÕES DO CAPÍTULO

Este capítulo apresentou a primeira contribuição da dissertação, que foi a apresentação de um novo modelo para a construção de caixas-S dinâmicas. Sua arquitetura está fundamentada no conceito de caminhos aleatórios, sendo a chave secreta a responsável por definir o caminho aleatório a ser percorrido e desta forma determinando o conjunto de funções utilizadas no processo de transformação.

Quanto ao nível de segurança, a nova proposta de construção de caixas-S é composta por dois níveis. O primeiro nível de segurança está atrelada à chave secreta utilizada, e sua complexidade de quebra por força bruta é da ordem $O(2^n)$, sendo n o tamanho da chave utilizada. No caso específico da avaliação realizada neste trabalho, a complexidade foi de $O(2^{128})$. Já o segundo nível de segurança está associado às funções utilizadas para a composição dos vértices. Para demonstrar a importância dessa escolha, foi apresentado o processo de distribuição das funções, e em seguida realizou-se a avaliação de segurança de uma amostra de caixas-S com o objetivo de verificar se o segundo nível de segurança era satisfatórios. Para isso, utilizou-se as métricas apresentadas no capítulo 3 e verificou-se mediante a avaliação realizada, que as funções escolhidas foram satisfatórias. Contudo, vale ressaltar que outras funções podem ser utilizadas na tentativa de obter melhores resultados quando submetidas às métricas de avaliação, como por exemplo as *bent functions* propostas por (ROTHAUS, 1976).

6 AVALIAÇÃO E PROPOSTA DE UMA NOVA FUNÇÃO DE EXPANSÃO DE CHAVE PARA O ALGORITMO AES

Este capítulo tem como objetivo apresentar uma nova função de expansão de chave para o algoritmo AES, com o objetivo de torná-lo um algoritmo mais seguro levando em consideração os critérios de classificação apresentados por (CARTER, 1999). Além disso, foi realizada uma análise comparativa com as cinco funções de expansão de chave dos algoritmos finalistas do concurso do AES levando em consideração critérios como quantidade de subchaves com peso de hamming balanceado; propagação ou correção de erros quando as subchaves fossem concatenadas; aleatoriedade das chaves quando submetidas ao testes do NIST e a verificação do SAC.

6.1 FUNÇÃO DE EXPANSÃO DE CHAVE

As funções de expansão de chave vêm sendo muito utilizadas nos algoritmos de criptografia simétricos, ou seja, algoritmos em que a mesma chave secreta é utilizada para cifrar e decifrar a mensagem. Essas funções surgiram com o propósito de aumentar a segurança do algoritmo sem aumentar o espeço de memória necessário para o armazenamento de chaves, ou seja, considerando que um algoritmo requer um registrador de n bites para armazenar a chave secreta, sendo n o tamanho da chave, todas as subchaves devem ser armazenadas no mesmo registrador, só que em momentos diferentes. Logo, cada subchave deverá ser processada em uma rodada do algoritmo de cifragem e armazenada no registrador. Dessa forma, será utilizada no processo de cifragem uma chave K de tamanho $n * r$, sendo r o número de rodadas do algoritmo.

Não obstante, se o algoritmo criptográfico utilizar uma função de expansão de chave, em que todas as subchves são geradas e armazenadas antes de iniciar o processo de cifragem, esse algoritmo não estará usufruindo da melhor característica desta função, que é otimizar o espaço de armazenamento de chaves. Sendo assim, este algoritmo também estaria utilizando uma chave k de tamanho $n * r$, entretanto, ele seria equivalente a um algoritmo que utilizasse direto uma chave secreta de tamanho $n * r$ sem a necessidade de expansão. Logo, se o uso da expansão de chaves não for *on-the-fly*, utilizar uma única chave seria melhor, pois estaria ocupando a mesma quantidade de memória requerida pelo

segundo procedimento, e economizando o processamento das subchaves.

Nesse contexto, a função de expansão de chave foi uma forma encontrada para a utilização de chaves maiores com a otimização do espaço de armazenamento. Pelo conhecimento que se tem, o primeiro algoritmo que utilizou esse procedimento foi o algoritmo Lucifer eleito em 1977 o padrão DES com algumas modificações, versão mais atual FIPS 46-3 (NIST, 1999). Como nessa época o recuro computacional era escasso, utilizar o procedimento de expansão de chave era uma boa forma de manter a segurança do algoritmo sem ter problemas com os recursos computacionais.

Entretanto, algoritmos mais novos como o Serpent, Towfish, MARS, RC6 e o próprio AES, atual padrão norte americano, ainda utilizam esse procedimento para a geração de subchaves. Sendo assim, observou-se que a utilização de funções de expansão de chave agregam outros benefícios, como por exemplo a geração de subchaves consideradas fortes a partir de uma chave secreta fraca. Entende-se como chave forte, uma chave em que a proporção de zeros e uns é equilibrada, ou seja, o peso de hamming da chave está próximo de $\frac{n}{2}$, já as chaves fracas são justamente o inverso, são chaves em que o número de zeros e uns é desequilibrado, sendo o peso de hamming muito maior ou menor do que $\frac{n}{2}$.

Ademais, esse capítulo analisará as funções de expansão de chave dos cinco algoritmos finalistas do concurso do AES, juntamente com a nova função que será proposta. Para parâmetro de análise será utilizado a metodologia de classificação proposta por (CARTER, 1999).

6.2 METODOLOGIA DE CLASSIFICAÇÃO DE FUNÇÕES DE EXPANSÃO DE CHAVES PROPOSTAS POR CARTER, DAWSON E NIELSEN (1999)

(CARTER, 1999) publicaram na segunda conferência do concurso do AES, um artigo com o título *Key Schedule Classification of the AES Candidates*. O objetivo desse artigo foi avaliar os cinco algoritmos finalistas do concurso, levando em consideração apenas a segurança de suas respectivas funções de expansão de chave. Para isso, foram estabelecidas duas categorias de classificação. Na primeira categoria se enquadram as funções em que a partir de uma subchave é possível ter conhecimento de alguma outra subchave ou até mesmo da chave secreta utilizada no algoritmo. Já a segunda categoria é o inverso, ou seja, a partir de uma subchave não é possível ou é muito difícil obter o conhecimento de alguma outra subchave ou da chave secreta utilizada no algoritmo.

A seguir são apresentadas as subcategorias e suas descrições:

- Categoria 1(A): Todos os bites da chave secreta são utilizados em cada rodada. Então, o conhecimento de uma subchave permitirá o conhecimento de todos os bites da chave secreta e de todas as outras subchves.
- Categoria 1(B): O conhecimento de uma subchave permite descobrir alguns bites da chave secreta ou de outras subchaves.
- Categoria 1(C): O conhecimento de uma subchave permitirá o conhecimento de outras subchaves ou da chave secreta depois de algumas simples operações de inversão.
- Categoria 2(A): Nem todos os bites da chave secreta são utilizados para criar as subchaves. Nessa categoria, certas chaves secretas produzem ao menos duas chaves idênticas na rodada. Em outras palavras, a entropia das subchaves não é maximizada.
- Categoria 2(B): Todos os bites da chave secreta são utilizados para produzir as subchaves, maximizando a entropia das subchaves.
- Categoria 2(C): Cada subchave é gerada independentemente, e o tamanho da chave secreta e igual a soma do tamanho de todas as subchaves.

Para determinar a classificação de cada algoritmo por categoria e em seguida a subcategoria, os autores elaboraram as seguintes perguntas:

- O conhecimento de todos os bites de uma subchave, possibilita o conhecimento de alguma outra subchave ou da chave secreta?
- Todas as subchaves dependem de todos os bites da chave secreta?

Se a resposta da primeira pergunta for positiva, significa que a função avaliada pertence a primeira categoria. Já a resposta da segunda pergunta, caso também seja positiva, determinará a subcategoria da função caso ela pertença a segunda categoria, ou seja, a segunda pergunta só é realizada caso a primeira seja negativa.

A TAB 6.2 ilustra a classificação realizada pelos autores, ressaltando que os algoritmos em negrito são os cinco finalistas do concurso do AES.

A partir dessa classificação é possível perceber que dentre os cinco finalistas apenas o algoritmo Rijmdael foi classificado na categoria 1C, ou seja, sua função de expansão

TAB. 6.1: Classificação dos algoritmos de acordo com suas funções de expansão de chave. Fonte: (CARTER, 1999).

Type 1A	*Type 1B*	*Type 1C*	*Type 2A*	*Type 2B*	*Type 2C*
	MAGENTA	CRYPTON DEAL **Rijmdael (AES)** SAFER+	DFC	CAST-256 E2 FROG HPC LOKI97 **MARS** **RC6** **Serpent** **Twofish**	

de chave proporciona a partir de uma subchave o conhecimento de outras subchaves e até mesmo da chave secreta. Sendo assim, pode-se levantar a hipótese de que o critério de segurança das funções de expansão de chave não foi um dos fatores decisivos para a escolha do algoritmo vencedor, pois o Rijmdael foi escolhido o novo padrão AES.

Uma das possibilidades para que esse critério não fosse crucial para a escolha do novo algoritmo AES, seria a de que o grau de complexidade necessário para descobrir alguma subchave que permitisse o conhecimento da chave secreta fosse inviável computacionalmente. Entretanto, observou-se que já existem estudos que exploram justamente essa característica, os vários tipos de ataques DFA são um exemplo disso. Portanto, é evidente que o atual AES fica em desvantagem com relação a segurança de sua função de expansão de chave em relação aos outros quatro algoritmos finalistas do concurso.

Nesse contexto, este trabalho tem como objetivo realizar modificações na função de expansão de chave do algoritmo AES sem alterar de forma significativa a estrutura do algoritmo original, mas que seja o suficiente para que o algoritmo seja inserido na categoria 2B, categoria esta que é considerada mais segura do que a categoria 1C.

6.3 DESENVOLVIMENTO DA NOVA FUNÇÃO DE EXPANSÃO DE CHAVE

Antes de realizar as modificações necessárias, realizou-se a análise no algoritmo de expansão de chave do algoritmo AES a fim de verificar o ponto crítico do algoritmo que permite o conhecimento da chave secreta. Para isso, foi observado a função de expansão de chave conforme apresentada na FIG 4.1 e a partir dela elaborou-se o algoritmo 6.3 que corresponde ao inverso da função. Este algoritmo teve como finalidade demonstrar que

realmente é possível obter a chave secreta a partir da última subchave de rodada conforme argumentado por (GIRAUD, 2004).

6.3 Função de expansão de chave inversa do algoritmo AES

1: $w[n] \leftarrow \{0,1\}_n$ # Conjunto de Subchaves em palavras de 32 bites
2: $Nk \leftarrow (n/4) - 7$ # Tamanho das Subchaves em palavras de 32 bites
3: **for** $i \leftarrow n-1$ até Nk **do**
4: **if** $i\%Nk = 0$ **then**
5: $w[i-Nk] \leftarrow w[i] \oplus subWord(rotWord(w[i-1])) \oplus rcon[(i/Nk)-1]$
6: **else if** $(Nk > 6) \wedge (i\%Nk = 4)$ **then**
7: $w[i-Nk] \leftarrow w[i] \oplus subWord(w[i-1])$
8: **else**
9: $w[i-Nk] \leftarrow w[i] \oplus w[i-1]$
10: **end if**
11: $i \leftarrow i-1$
12: **end for**
13: **return** w

Partindo do princípio que o ataque DFA apresentado no capítulo 4 é bem sucedido, pode-se então utilizar a última subchave como dado de entrada do algoritmo 6.3 e descobrir a chave secreta utilizada no processo de cifragem. Isso só é possível pois o principal componente utilizado para a transformação dos bites é o mesmo que o utilizado no algoritmo de cifragem, no caso a transformação caixa-S. Como essa transformação é representada por uma tabela pré-processada, fica evidente que a tabela inversa pode ser obtida facilmente. Desta forma, por ser a única transformação não linear, e ser facilmente reversível, é evidente que a caixa-S é o principal componente da função de expansão de chave que a classifica na categoria 1C.

Sendo assim, uma alternativa para solucionar esse problema seria substituir a caixa-S estática pela caixa-S dinâmica apresentada no capítulo 5. Dessa forma, mesmo que a última subchave fosse obtida mediante ao ataque DFA, ainda assim não seria possível obter a chave secreta pois a função de expansão de chave não permitiria, haja vista que agora a caixa-S é dinâmica, ou seja, utiliza a chave secreta para determinar os vértices utilizados na transformação dos bites. Logo, não seria fácil descobrir a chave secreta a partir de uma subchave, para conseguir isso, por força bruta, seria necessário realizar no

pior caso 2^{128} combinações de caminhos aleatórios.

O algoritmo 6.4 ilustra a nova função de expansão de chave com a caixa-S dinâmica. Realizadas as modificações, considera-se que o objetivo foi alcançado, pois a arquitetura da função não foi alterada drasticamente, mas o suficiente para que fosse inserida na categoria 2B, categoria esta que engloba as funções que não permitam ou que dificultem bastante o conhecimento de alguma subchave ou da chave secreta por meio de outra subchave, além disso, essas funções devem utilizar todos os bites da chave secreta para produzir as subchaves.

6.4 Função de expansão de chave com caixa-S dinâmica

1: $w[Nb*(Nr+1)] \leftarrow \{0,1\}_n$ # Conjunto de Subchaves em palavras de 32 bites
2: $k[4*Nk]$ # Chave em bytes
3: $v[32*Nk] \leftarrow \{0,1\}$ # Vetor utilizado para de terminar o caminho aleatório do grafo
4: $c \leftarrow 0xb7e15163$ # Contante utilizada para modificar a chave secreta
5: $i \leftarrow 0$
6: **while** $i < Nk$ **do**
7: $w[i] \leftarrow word(k[4*i], k[4*i+1], k[4*i+2], k[4*i+3])$
8: $v[i] \leftarrow word(k[4*i], k[4*i+1], k[4*i+2], k[4*i+3]) \oplus c$
9: $i \leftarrow i+1$
10: **end while**
11: $i \leftarrow Nk$
12: **while** $i < Nb*(Nr+1)$ **do**
13: $temp \leftarrow w[i-1]$
14: **if** $i\%Nk = 0$ **then**
15: $temp \leftarrow subWord^{dinamica}(rotWord(temp), v) \oplus rcon[(i/Nk)]$
16: **else if** $(Nk > 6) \wedge (i\%Nk = 4)$ **then**
17: $temp \leftarrow subWord^{dinamica}(temp, v)$
18: **end if**
19: $w[i] \leftarrow w[i-Nk] \oplus temp$
20: $i \leftarrow i+1$
21: **end while**

Nesse diapasão, após essas modificações foi considerado que a nova função pertencia a categoria 2B, o que demonstra sua maior segurança, entretanto, foi necessário realizar

uma nova análise desta função com o objetivo de verificar a sua eficiência na produção de subchaves, para isso, as seis funções de expansão de chave em estudo foram submetidas aos testes estatísticos de aleatoriedade do NIST (RUKHIN, 2001). Além disso, também foi verificado o SAC das funções.

6.4 ANÁLISE COMPARATIVA DA PRODUÇÃO DE SUBCHAVES DAS FUNÇÕES

Para realizar a análise das funções de expansão de chave, foram geradas cem mil chaves de cento e vinte e oito bites, utilizando o gerador *SecureRandom* disponível no pacote *security* da linguagem de programação Java. A partir dessa amostra, foi gerado um conjunto de um milhão de subchaves de cento e vinte e oito bites por função, totalizando uma amostra de seis milhões de subchaves.

6.4.1 AVALIAÇÃO DOS PESOS DE HAMMING

A primeira medida de avaliação utilizada foi a classificação das chaves por peso de *hamming*, como as chaves utilizadas são de cento e vinte e oito bites, existem cento e vinte e nove possíveis grupos de *hamming* para classificar o universo das chaves $\{0, ..., 128\}$. Sabendo que a distribuição das chaves com base nessa classificação pode ser representada por uma distribuição binomial sendo $\mu = 64$ e $\sigma = 5,656854249$, verificou-se a proporção de subchaves que estariam presentes no intervalo $\mu - \sigma$ a $\mu + \sigma$. Essa análise teve como objetivo verificar a quantidade de chaves que podem ser consideradas aleatórias, pois chaves balanceadas possuem maior chance de serem aprovadas nos testes estatísticos de aleatoriedade do NIST.

A TAB 6.2 ilustra a proporção de subchaves que possuem hamming no intervalo especificado. Os resultados demonstram que as seis funções de expansão de chave produzem subchaves com pesos de hamming parecidos. Diante desses valores, espera-se que a quantidade de chaves aprovadas pelos testes de aleatoriedade do NIST sejam similares aos valores obtidos, pois caso a proporção de chaves aprovadas seja maior, significa que os testes estariam aprovando chaves desbalanceadas, ou seja, chaves com uma grande diferença de zeros e uns.

TAB. 6.2: Proporção de chaves dentro do intervalo de hamming 59 a 69

algoritmo	quantidade de chaves avaliadas	proporção de chaves no intervalo
AES Modificado	10^6	0.668914
AES	10^6	0.668795
Twofish	10^6	0.669064
Serpent	10^6	0.66738
MARS	10^6	0.656115
RC6	10^6	0.669419

6.4.2 AVALIAÇÃO DA PROPAGAÇÃO OU CORREÇÃO DE ERROS

Outra análise realizada foi a verificação do fator de correção ou propagação de erros quando realizada a concatenação de subchaves. Sendo assim, as amostras de chaves foram reagrupadas de tal forma que gerassem um novo conjunto de cento e vinte e oito chaves com um milhão de bites, esse procedimento foi realizado na amostra de cada algoritmo. Os valores de μ e σ foram calculados para os novos vetores, sendo $\mu = 5 * 10^5$ e $\sigma = 500$. Sendo assim, verificou-se a proporção de subchaves que estariam presentes no intervalo $\mu - \sigma$ a $\mu + \sigma$, ou seja, a quantidade de chaves com zeros e uns balanceados.

Essa análise teve como objetivo verificar se a diferença entre os pesos de hamming dos vetores de cento e vinte e oito bites, podem ser corrigidos quando estes forem concatenados formando um vetor de um milhão de bites. Se a proporção de chaves aceitas no intervalo especificado, for maior do que a proporção de chaves aceitas no intervalo especificado para os vetores de cento e vinte e oito bites, significa que houve um fator de correção. Caso contrário, se a proporção de chaves aceitas for menor do que a proporção de aceitação das chaves de cento e vinte e oito bites, significa que o erro dos pesos de hamming foi propagado ao longo da concatenação.

A TAB 6.3 ilustra a proporção de chaves que possuem o peso de hamming no intervalo especificado. Para esta análise, é importante ressaltar que se for observado a propagação de erros, significa que é mais conveniente usar uma outra função de expansão de chave ou utilizar uma chave secreta cujo tamanho seja igual a soma de todas as subchaves, eliminando a necessidade de um gerador de subchaves. Pensando nessa segunda hipótese, foi verificado a proporção de aceitação de vetores com um milhão de bites do gerador *SecureRandom* de tal forma que não fosse necessário a concatenação. Foram geradas cento e vinte e oito chaves de um milhão de bites, e o número de chaves aceitas foi de oitenta e cinto, ou seja a proporção de chaves aceitas foi de 0.6640625. Isso significa que se

os geradores de subchaves, quando concatenam suas chaves, obtiverem uma proporção de aceitação similar a de um gerador de número aleatório que não necessita de concatenação, as subchaves produzidas podem ser consideradas tão boas quão uma única chave secreta de tamanho igual a soma de todas as subchaves. Entretanto, isso não significa que esse é o único procedimento necessário para assegurar que os geradores são bons ou ruins, pois além dessa avaliação, ainda é necessário verificar sua aleatoriedade com os testes estatísticos do NIST e também o efeito avalanche.

TAB. 6.3: Proporção de chaves dentro do intervalo de hamming 499500 a 500500

algoritmo	quantidade de chaves avaliadas	quantidade de chaves no intervalo	proporção de chaves no intervalo
AES Modificado	128	87	0.6796875
AES	128	83	0.6484375
Twofish	128	84	0.65625
Serpent	128	96	0.75
MARS	128	0	0.0
RC6	128	94	0.734375

Em relação aos resultados apresentados na TAB 6.3, é importante destacar que a função de expansão de chave do algoritmo MARS propagou o erro dos pesos de hamming das chaves de cento e vinte e oito bites de tal forma que todas as chaves concatenadas foram rejeitadas quando avaliadas no intervalo para vetores de um milhão de bites. Essa característica indica que para o algoritmo MARS seria interessante, ou reestruturar o algoritmo de expansão de chave, ou substituí-lo por outro, ou até mesmo utilizar uma única chave secreta de tamanho igual a soma de todas as subchaves. Não obstante, para essa última alternativa, o algoritmo necessitaria de mais espaço de memória, pois as chaves não seriam geradas a cada rodada.

6.4.3 AVALIAÇÃO ESTATÍSTICA DE ALEATORIEDADE

A próxima avaliação diz respeito a análise estatística de aleatoriedade das sequências binárias produzidas pelas funções em estudo. Para isso, utilizou-se o conjunto de testes estatísticos proposto pelo NIST (RUKHIN, 2001), cujo objetivo é verificar a existência de desvios de padrões de aleatoriedade em sequências binárias. Quinze testes de hipótese foram elaborados para essa verificação, a hipótese de nulidade H_0, afirma que a sequência é aleatória, já a hipótese alternativa H_1, afirma que a sequência não é aleatória. Para cada teste aplicado, a decisão é avaliada sobre a rejeição ou aceitação da hipótese de nulidade.

Os resultados dos quinze testes são comparados com o valor crítico α. Se o resultado de pelo menos um teste exceder o valor crítico, significa que a hipótese nulidade deve

ser rejeitada, ou seja, a sequência binária não é considerada aleatória. Caso contrário, a hipótese nulidade é não rejeitada, sendo a sequencia binária considerada aleatória.

A TAB 6.4 ilustra os possíveis resultados que podem ocorrer durante o processo de interpretação de um teste de hipóteses. A probabilidade de uma sequência aleatória ser rejeitada (erro tipo I) é geralmente denominada como nível de significância do teste, e é representada pela letra α. Já a probabilidade de uma sequência não aleatória ser aceita (erro tipo II) é representada pela letra β.

TAB. 6.4: Possíveis decisões no teste de hipóteses

Hipóteses	Aceitar H_0	Rejeitar H_0
Sequência aleatória H_0	nenhum erro	erro tipo 1
Sequência não aleatória H_1	erro tipo 2	nenhum erro

(RUKHIN, 2001) sugeriram que o valor de α fosse igual a $\alpha = 0.01$. Assim, quando o o valor-p que corresponde ao resultado do teste for $valor-p \geq 0.01$, significa que a sequência deve ser considerada aleatória com um grau de confiança de 99%. Já se $valor - p < 0.01$, significa que a sequência não deve ser considerada aleatória com um grau de confiança de 99%.

Nesse contexto, a TAB 6.5 ilustra a proporção de chaves que foram aprovadas em todos os quinze testes, ou seja, os vetores que não excederam o nível de significância em nenhum dos testes. Os resultados demonstram que a proporção de chaves aprovadas está coerente com a classificação de hamming realizada nos vetores de cento e vinte e oito bites como também nos vetores de um milhão de bites. Isso significa que existe uma coerência entre os resultados dos testes estatísticos com os vetores cujo peso de hamming esta próximo do balanceamento ideal.

Contudo, analisando os resultados, pode-se observar que a proporção de vetores aprovados pelas funções do algoritmo AES e Serpent é um pouco maior do que a proporção observada tanto na classificação das subchaves de cento e vinte e oito bites, quanto nas de um milhão de bites, esse comportamento pode insinuar a existência de vetores com o erro do tipo 2, ou seja, vetores que foram considerados aleatórios quando na verdade não deveriam. Outra hipótese para esse comportamento seria a de que os testes considerem como vetor aleatório uma faixa um pouco maior do que a especificada na analise de hamming, e observe outros comportamentos, como por exemplo a periodicidade de zeros e uns consecutivos.

A segunda hipótese parece ser a mais convincente, caso contrário, seria suficiente

realizar apenas a análise dos pesos de hamming, o que definitivamente não é verdade, pois uma chave de cento e vinte e oito bites com sessenta e quatro zeros consecutivos terá um hamming balanceado, entretanto, certamente esta chave será reprovada por um dos testes estatísticos.

Outro resultado interessante foi a proporção de aceitação do algoritmo MARS, que ficou bem abaixo dos demais. Contudo, esse resultado pode ser explicado com a análise de concatenação dos vetores, quando os vetores foram concatenados houve uma propagação muito grande dos erros de hamming a tal ponto que nenhum vetor de um milhão de bites ficasse dentro da faixa de aceitação. Dessa forma, esse fator pode ter influenciado na avaliação quando submetidos aos teste estatísticos.

Ademais, a função que está sendo proposta e as funções dos algoritmos Twofish e RC6, obtiveram um resultado convincente com as outras análises.

TAB. 6.5: Proporção de chaves aprovadas nos testes de aleatoriedade do NIST

algoritmo	proporção de chaves aprovadas
AES Modificado	0.6640625
AES	0.7578125
Twofish	0.6640625
Serpent	0.7265625
MARS	0.3671875
RC6	0.6953125

A TAB 6.6 ilustra os parâmetros de configuração utilizados para a avaliação das amostras. Em que n representa o número de bites do vetor de entrada. Os demais parâmetros possuem significados diferentes em cada teste, então, recomenda-se para um melhor entendimento de cada parâmetro e até mesmo do objetivo e sistematização de cada teste, que consulte a referência (RUKHIN, 2001).

6.4.4 AVALIAÇÃO DO EFEITO AVALANCHE E COMPLETUDE DOS VETORES

Por fim, a última avaliação tem como objetivo verificar o comportamento das funções de expansão de chave quando os dados de entrada sofrem pequenas variações. Para realizar esse procedimento, uma chave secreta de cento e vinte oito bites foi escolhida aleatoriamente e utilizada como dado de entra em cada função, gerando seus respectivos conjuntos de subchaves. A partir daí foram realizadas n iterações, sendo n tamanho da chave, e para cada iteração, o bite i era complementado. Dessa forma, a nova chave era utilizada como dado de entrada das funções, gerando um novo conjunto de subchaves.

TAB. 6.6: Configuração utilizada no conjunto de testes

Testes	Dados de entrada
1 The Frequency (Monobit) Test	$n = 128$
2 Frequency Test within a Block	$n = 128$, $M = 20$
3 The Runs Test	$n = 128$
4 Tests for the Longest-Run-of-Ones in a Block	$n = 128$
5 The Binary Matrix Rank Test	$n = 10^6$, $M = 8$, $Q = 8$
6 The Discrete Fourier Transform (Spectral) Test	$n = 128$
7 The Non-overlapping Template Matching Test	$n = 10^6$, $N = 8$, $B =$ "101010101"
8 The Overlapping Template Matching Test	$n = 10^6$, $N = 8$, $B =$ "101010101"
9 Maurer's "Universal Statistical"Test	$n = 10^6$, $L = 10$, $Q = 2048$
10 The Linear Complexity Test	$n = 10^6$, $M = 5000$
11 The Serial Test	$n = 10^6$, $m = 16$
12 The Approximate Entropy Test	$n = 10^6$, $m = 16$
13 The Cumulative Sums (Cusums) Test	$n = 128$
14 The Random Excursions Test	$n = 10^6$
15 The Random Excursions Variant Test	$n = 10^6$

As subchaves de cada conjunto eram então comparadas duas a duas utilizando a distância de hamming. Se por meio dessa comparação, após todas a iterações, não for encontrada nenhuma distância de hamming igual a zero, significa que a função satisfaz a métrica de completude. E se os vetores que obtiverem distância de hamming próximo de $n/2$, significa que esses vetores satisfazem o efeito avalanche. Os vetores que satisfizerem as duas condições, estarão satisfazendo a propriedade de SAC (*Strict avalanche criterion*).

Nassa mesma linha de pensamento, essa avaliação tem como objetivo verificar a proporção de subchaves que satisfazem o SAC quando escolhida uma chave secreta qualquer. Se a avaliação indicar que uma boa proporção das Subchaves satisfazem o SAC, significa que a função geradora pode oferecer resistência contra ataques diferencias.

A TAB 6.7 ilustra a proporção de subchaves que satisfazem o SAC. Os resultados mostram que todas as funções apresentaram proporções parecidas, o que a priori indica que todas as funções apresentam o mesmo nível de aceitação de SAC. Não obstante, para essa análise não é suficiente apenas avaliar a quantidade de subchaves que satisfazem o critério de aceitação, também é importante verificar o comportamento das subchaves que estão fora do critério de aceitação.

Para essa avaliação foram analisadas as FIG 6.1; FIG 6.2; FIG 6.3; FIG 6.4; FIG 6.5 e FIG 6.6. As distâncias de hamming entre cinquenta e nove a sessenta e nove correspondem as subchaves que satisfazem o critério de SAC. Logo para as distâncias que estão fora dessa

TAB. 6.7: Proporção de subchaves que satisfazem o SAC

Algoritmo	Quantidade de subchaves avaliadas	Quantidade de subchaves aceitas	Proporção de subchaves que satisfazem o SAC
AES modificado	12800	8640	0.675
AES	12800	7555	0.590234375
Twofish	12800	8531	0.666484375
Serpent	12800	8039	0.628046875
MARS	12800	8400	0.65625
RC6	12800	8608	0.6725

faixa, é importante observar o tamanho da cauda de cada função, pois quanto maior for a cauda para a esquerda, significa que o número de bites diferentes entre duas subchaves é menor, ou seja, as chaves são mais semelhantes, já se a cauda se estender para a direita, significa que o número de bites diferentes entre duas subchaves é maior, e quanto maior for esse valor mais complementares são as chaves com essa característica.

Portanto, quanto menor forem as extremidades da distribuição, melhor é a aceitação global da função geradora, pois apesar de existirem distâncias de hamming fora do intervalo de aceitação, se a cauda for curta, significa que as chave estão próximas da região crítica de aceitação. Entretanto, se o comportamento for o oposto, significa que a função de expansão de chave produz subchaves muito parecidas quando as chaves secretas são parecidas.

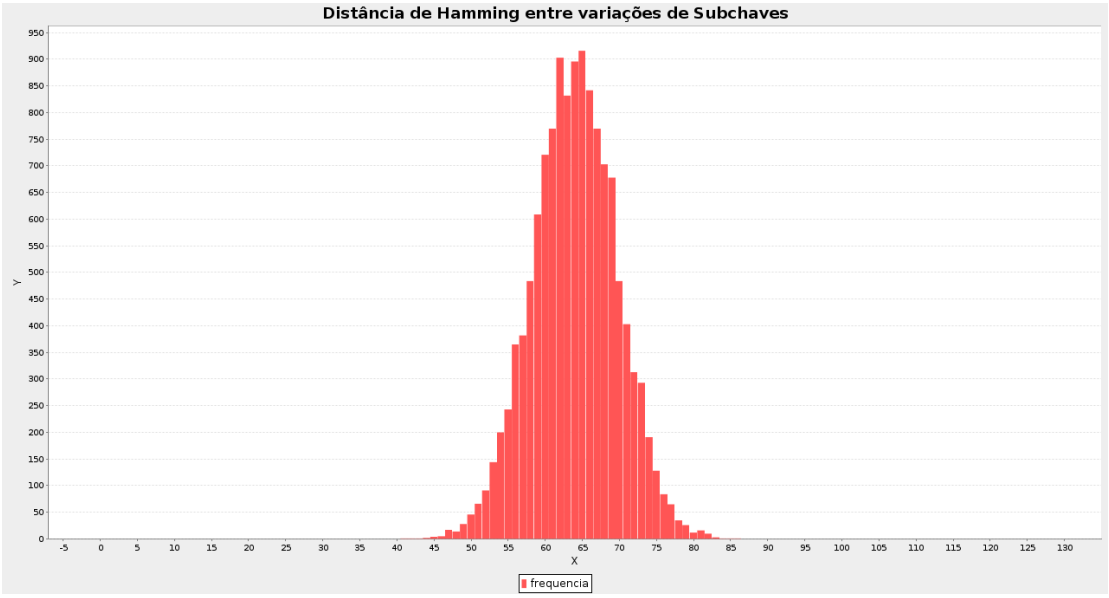

FIG. 6.1: SAC AES Modificado

Analisando a FIG 6.1 que corresponde a distribuição da função que está sendo proposta nesse trabalho, pode-se observar que a distribuição possui uma cauda curta. Sendo a menor e a maior diferença entre as subchaves próximas de quarenta e cinco e oitenta e cinco respectivamente. Isso significa que apesar de existirem subchaves que não satisfaçam

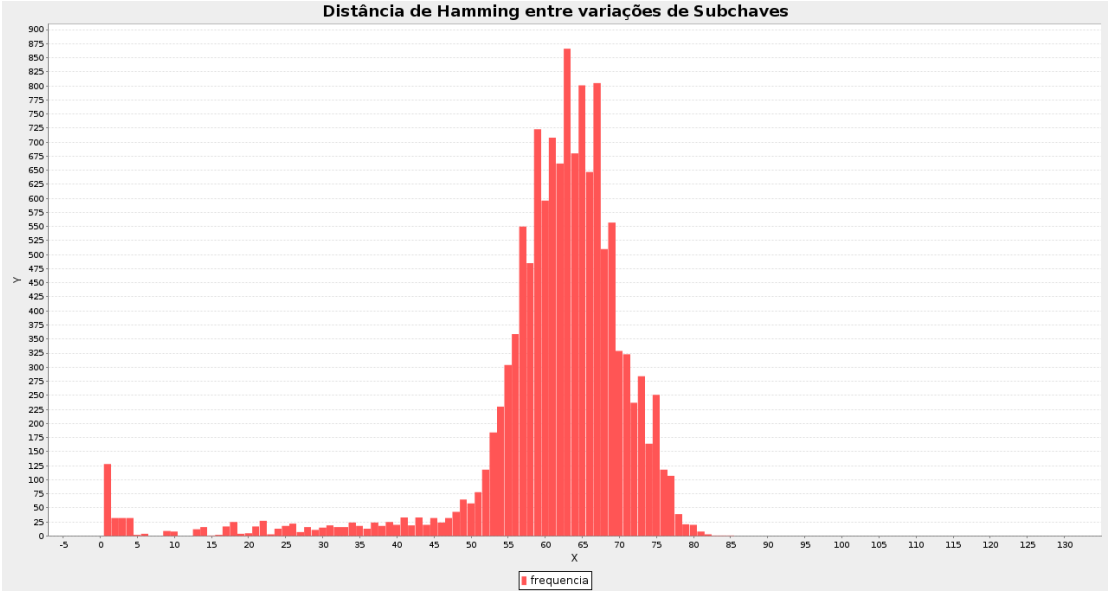

FIG. 6.2: SAC AES

o SAC, estas não estariam tão distante disso. Diferentemente do que foi observado na FIG 6.2 que corresponde a distribuição da função do AES, nesse caso, a cauda a esquerda é bastante acentuada, demonstrando que apesar das duas funções, função proposta e função AES, apresentarem uma proporção parecida de subchaves que satisfazem o SAC, não significa que essas funções são parecidas, pois como observado, das subchaves que estão fora do intervalo de aceitação, as subchaves do AES são muito mais semelhantes do que as subchaves da função que está sendo proposta.

Sendo assim, esse comportamento indica que a função de expansão de chave original do AES está muito mais sujeita a ataques diferencias se comparada com a função que está sendo proposta nesse trabalho.

Assim como a função do AES, a função de expansão de chave do algoritmo Serpent ilustrada na FIG 6.4, também apresenta um comportamento de cauda acentuada a esquerda. Demonstrando que essa função também gera subchaves muito parecidas quando se utiliza chaves secretas parecidas.

Outrossim, as outras funções correspondentes aos algoritmos Twofish, MARS e RC6, apresentam um comportamento semelhante ao da função que está sendo proposta nesse trabalho, ou seja, possuem caudas curtas indicando que os valores fora do intervalo de aceitação estão próximos da região crítica.

Esse comportamento demonstra a importância da avaliação SAC tanto para os vetores que estão dentro da região de aceitação, quanto para os que estão fora. Pois mesmo que as

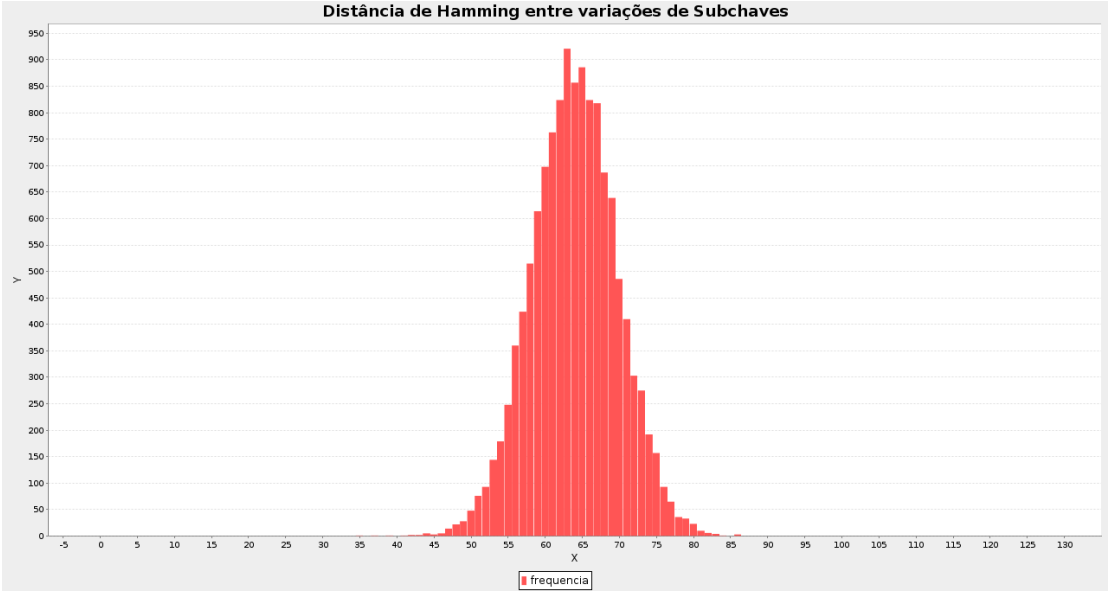

FIG. 6.3: SAC Twofish

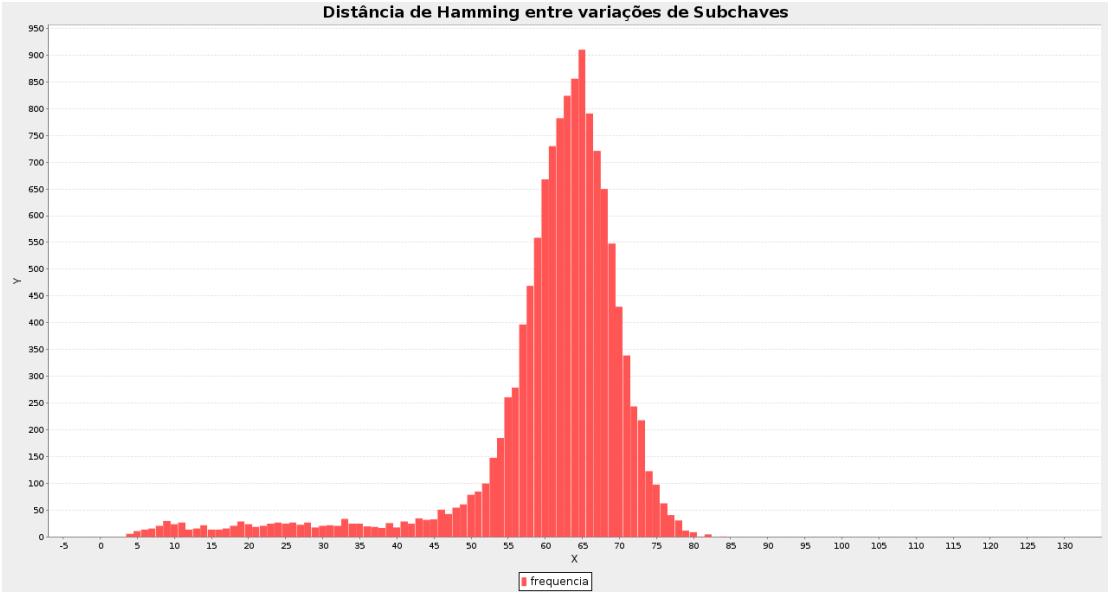

FIG. 6.4: SAC Serpent

funções apresentem proporções de aceitações parecidas, as caudas são fundamentais para distinguir as funções que podem apresentar maior resistência aos ataques diferencias.

6.5 CONCLUSÕES DO CAPÍTULO

Esse capítulo apresentou a avaliação da segurança das funções de expansão de chave dos cinco algoritmos finalistas do concurso do AES. Para isso, foi utilizado o critério de

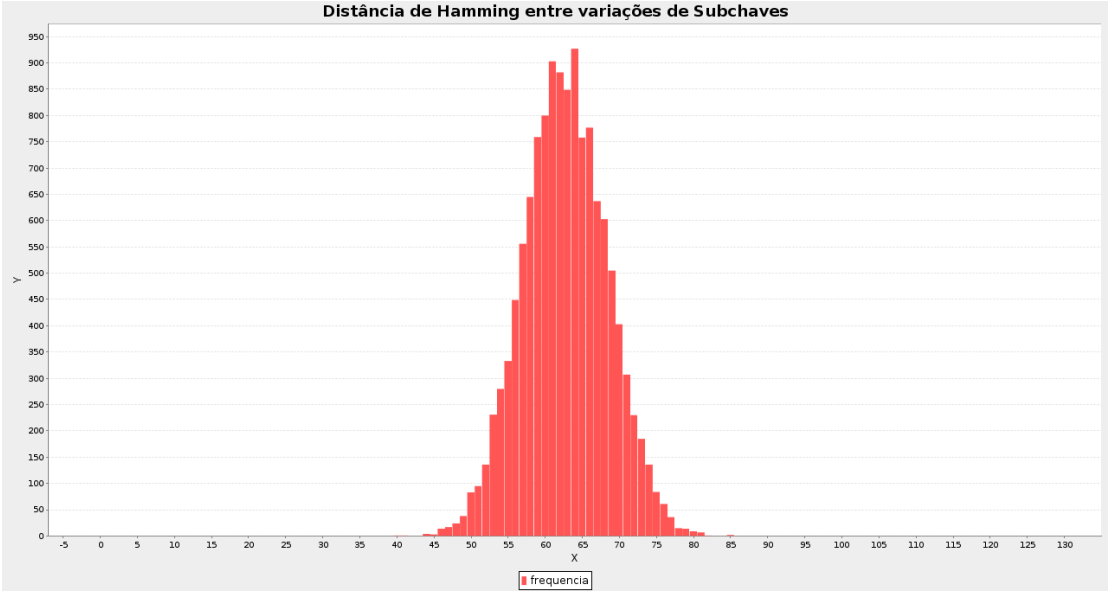

FIG. 6.5: SAC MARS

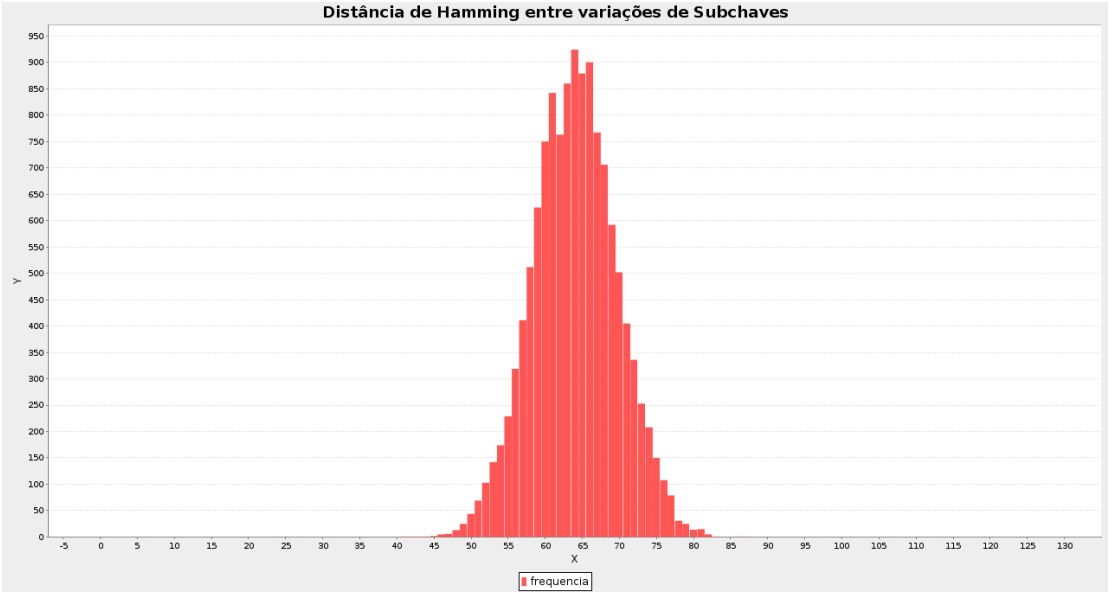

FIG. 6.6: SAC RC6

classificação proposto por (CARTER, 1999), através desse critério foi observado que a única função que não estava no grupo das funções mais seguras era a função do algoritmo AES. Por esse motivo o optou-se por analisar a função de expansão de chave do algoritmo AES e verificar qual o ponto crítico da função que permite o conhecimento da chave secreta a partir de uma subchave. Uma vez que o ponto crítico foi identificado, foi realizada a modificação no algoritmo de tal forma que a estrutura não fosse modificada drasticamente e fosse suficiente para que essa nova função fosse inserida no mesmo grupo

das outras funções consideradas mais seguras de acordo com o critério de classificação proposto por (CARTER, 1999).

Após essa modificação foi necessário avaliar a nova função no que diz respeito ao nível de aleatoriedade das subchaves produzidas. Nesse sentido foi realizada uma avaliação comparativa entre as funções de expansão de chave dos cinco algoritmos finalistas do AES e a nova função. Foram observadas critérios como a quantidade de subchaves com peso de hamming balanceado; a propagação ou correção de erros quando as subchaves fossem concatenadas; a aleatoriedade das chaves quando submetidas ao testes do NIST e a verificação do SAC.

Em relação aos resultados, foi observado que a função proposta, que corresponde a melhoria na transformação não linear do algoritmo de expansão de chave do AES, e as funções dos algoritmos Twofish e RC6 apresentaram resultados satisfatórios em todos os critérios de avaliação. Já a função do algoritmo MARS apresentou um resultado insatisfatório em relação a propagação de erros e na avaliação de aleatoriedade do NIST. As funções dos algoritmos AES e Serpent, apesar de apresentarem os melhores resultados nos testes de aleatoriedade do NIST, apresentaram resultados comprometedores quando submetidos ao critério de avaliação do SAC.

Esses resultados comprovam a importância do conjunto de critérios de avaliação, pois apesar de algumas funções apresentarem bons resultados em certos critérios, podem apresentar fraquezas em outros.

7 DISPERSÃO DE PADRÕES COM O MODO DE OPERAÇÃO ECB

Este capítulo tem como objetivo apresentar os resultados da avaliação do algoritmo AES modificado no que diz respeito a dispersão de padrões dos criptogramas. Para isso, foram realizados experimentos nos quais foi observada a transmissão dos padrões do texto claro para as respectivas cifras dos cinco algoritmos finalistas do concurso do AES, e ao mesmo tempo também foi observado que para o mesmo conjunto textos em claro, as cifras correspondentes ao algoritmo AES modificado diminuíram os padrões encontrados. Para realizar o agrupamento foi elaborado um novo algoritmo baseado em teoria dos grafos que tem como objetivo agrupar os criptogramas pertencentes a subgrafos conexos. E para a avaliação dos grupos foram utilizadas as métricas de revocação, precisão, grau de conectividade dos vértices e entropia da amostra.

7.1 RECONHECIMENTO DE PADRÕES EM CRIPTOGRAMAS

O estudo de reconhecimento de padrões em criptogramas vem sendo explorado pelo Grupo de Segurança da Informação do Instituto Militar de Engenharia (GSI/IME) desde 2006. (CARVALHO, 2006) iniciou essa linha de pesquisa explorando algoritmos de criptografia polialfabética e os algoritmos de cifra de bloco DES e AES. A principal contribuição desse trabalho foi a consolidação da fase de pré-processamento dos dados, que consiste na determinação do tamanho da palavra, construção do dicionário de palavras e a construção da matriz de similaridade. (SOUZA, 2007) deu continuidade na pesquisa contribuindo com uma análise empírica da avaliação dos coeficientes de similaridade e distância como: Co-seno, Dice, Jaccard, Overlap, Simple-matching, distância Euclidiana, distância Manhattan, distância Canberra e distância Bray-Curtis; e de métodos de agrupamentos hierárquicos aglomerativos, como: single-link, complete-link e group-average link. Confirmando o melhor desempenho da medida de similaridade co-seno e do método single-link dentro do contexto proposto. Além disso, (SOUZA, 2007) contribuiu com a aplicação de redes neurais para o processo de agrupamento dos criptogramas, essa foi a primeira tentativa do GSI/IME na identificação de algoritmos criptográficos a partir somente do criptograma, ou seja, sem a necessidade de informar o número de grupos desejáveis. (OLIVEIRA, 2011) é o trabalho antecessor ao presente trabalho, sua principal

contribuição foi o desenvolvimento de um novo algoritmo de agrupamento de criptogramas que não necessitasse do conhecimento do número de grupos a serem formados, e que melhorasse os resultados quando comparados com os resultados do algoritmo de redes neurais desenvolvido por (SOUZA, 2007). Para tal, (OLIVEIRA, 2011) desenvolveu um algoritmo de reconhecimento de padrões de criptogramas utilizando algoritmos genéticos. Além disso, (OLIVEIRA, 2011) propôs um método de classificação de criptogramas utilizando o conceito de Template Matching, cujo objetivo é classificar isoladamente criptogramas, a priori desconhecidos.

A FIG 7.1 ilustra o processo de reconhecimento de padrões em criptogramas desenvolvidos pelo GSI/IME. A contribuição desse trabalho para o processo foi o desenvolvimento de um novo algoritmo de agrupamento de criptogramas utilizando teoria dos grafos, sua vantagem em relação aos demais está no custo computacional, uma vez que o novo algoritmo necessita de menos tempo computacional para apresentar os mesmos resultados das outras metodologias utilizadas. Outra contribuição para o processo foi a utilização de dois novos métodos de avaliação, que são o grau de conectividade dos documentos e a entropia da amostra. O primeiro método foi desenvolvido em função da estrutura utilizada pelo novo algoritmo de agrupamento, como o algoritmo é fundamentado em teoria dos grafos, foi possível na etapa de avaliação observar a grau de conectividade dos vértices, que representa a relação entre os documentos agrupados. O segundo método de avaliação proposto foi a verificação da entropia da amostra, ou seja, verificar o grau de incerteza que a mostra de texto cifrado oferece quando comparada com a amostra dos textos em claro.

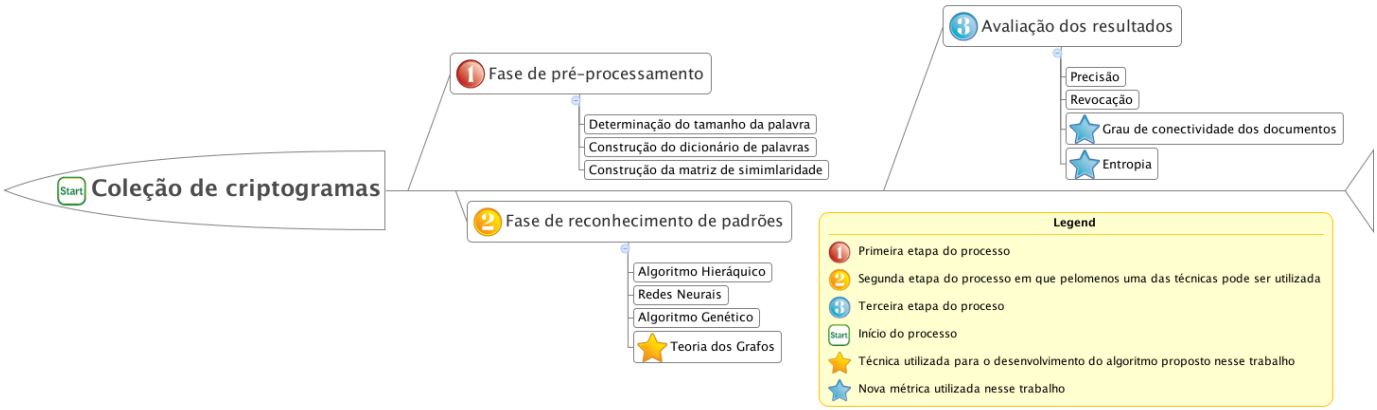

FIG. 7.1: Processo de reconhecimento de padrões em criptogramas GSI/IME

7.2 PROPOSTA DO NOVO ALGORITMO DE RECONHECIMENTO DE PADRÕES

Como já mencionado anteriormente, uma das contribuições desse trabalho para o processo de reconhecimento de padrões de criptogramas desenvolvido pelo GSI/IME foi o desenvolvimento de um novo algoritmo de agrupamento de criptogramas que fosse mais eficiente em relação ao tempo de processamento e que apresentasse resultados tão satisfatórios quanto aos demais algoritmos já projetados. Para isso, utilizou-se a teoria dos grafos para a fundamentação do algoritmo.

Dado um grafo não direcionado $G(V, E)$ em que $V = x_1, x_2, ..., x_n$ são os vértices e E as arestas (x_i, x_j), cujo peso determina a similaridade entre os vértices aos quais ela está conectada, o objetivo é encontrar o número de subgrafos conexos pertencentes à G. Um grafo é conexo quando existir um passeio entre quaisquer pares de seus vértices, um passeio é uma sequência $x_1, ..., x_k$ de vértices tais que $(x_i, x_{i+1}) \in E(G)$ para $1 \leq i \leq k-1$, $k > 1$. A estrutura de dados utilizada para a representação do Grafo foi a matriz de similaridade obtida na fase de pré-processamento.

Dessa forma, o primeiro passo do algoritmo consiste em construir a lista de adjacência dos vértices e em seguida escolher um vértice qualquer e realizar uma busca em largura para encontrar todos os vértices conexos ao vértice escolhido. Para cada vértice encontrado, repete-se o processo anterior até que todos os vértices conexos sejam visitados. Se todos os vértices já foram visitados então fim do processo de agrupamento, senão, escolhe-se um novo vértice ainda não visitado para encontrar uma nova partição.

Em relação a complexidade do algoritmo, ele pode ser expressado na ordem de $O(n^2)$. Outra característica interessante está na simplicidade do algoritmo e na forma de determinação dos grupos. Como os grupos são formados pelo número de subgrafos conexos, a quantidade de grupos não é um dado necessário para a execução do algoritmo. Essa característica também representa uma vantagem em relação aos outros algoritmos desenvolvidos pelo GSI/IME, pois o algoritmo genético apesar de encontrar o correto número de grupos por meio da função *Calinski-Harabasz*, ainda assim são necessárias uma série de outras configurações, como por exemplo o número de gerações, tamanho da população, número de cortes e etc...

Para verificar a eficiência dos resultados obtidos em relação aos outros algoritmos desenvolvidos pelo GSI/IME, foram realizadas duas publicações que demonstram a mesma eficiência dos resultados em relação a qualidade dos grupos. O primeiro trabalho publi-

cado foi (TORRES, 2010), nesse trabalho foram comparados os resultados da qualidade dos grupos formados pelo algoritmo genético e pelo algoritmo em grafos. Os resultados demonstram a eficiência dos dois métodos para o reconhecimento de criptogramas cifrados pelos algoritmos finalistas do concurso do AES. No segundo trabalho (TORRES, 2011), foram comparados os resultados dos agrupamentos realizados pelos algoritmos desenvolvidos em (CARVALHO, 2006), (SOUZA, 2007), (OLIVEIRA, 2011) e o proposto nesse trabalho. Os resultados mais uma vez demonstraram a eficiência dos algoritmos e suas particularidades.

7.3 MÉTRICAS DE AVALIAÇÃO

Para os experimentos realizados nesse trabalho, foram utilizadas quatro métricas de avaliação, dais quais duas também fazem parte da contribuição dessa pesquisa. As duas métricas já utilizadas nos trabalhos anteriores são a precisão e revocação. A precisão temo como objetivo avaliar a eficiência dos grupos em relação a sua homogeneidade, ou seja, verificar se um dado grupo é formado apenas por elementos de mesma característica, no caso dos criptogramas, verificar se os grupos formados contém apenas criptogramas cifrados com a mesma chave ou com o mesmo algoritmo. Já a revocação, tem como objetivo avaliar a eficiência dos grupos verificando a sua integridade, ou seja, verifica se não existe nenhum outro grupo que contenha algum criptograma cifrado com a mesma chave ou com o mesmo algoritmo iguais aos criptogramas do grupo em avaliação.

Portanto, é possível perceber que nem sempre que um determinado grupo obtenha precisão máxima, necessariamente ela também obterá a revocação máxima, e o inverso também é verdadeiro. Para ilustrar esse comportamento, considere o seguinte exemplo: dez documentos foram cifrados com o algoritmo α utilizando três chaves diferentes k_1, k_2 e k_3, totalizando uma amostra de trinta criptogramas. No processo de agrupamento foram encontrados quatro grupos conforme ilustrado na TAB 7.1. Os resultados mostram que para os dois primeiros grupos, apesar de a precisão ser igual a um, a revocação é diferente. Já para o terceiro grupo, tanto a precisão quanto a revocação foram iguais a um, e no caso do quarto grupo, apesar da revocação ter sido igual a um, a precisão foi de 0,9.

Sendo assim, a equação 7.1 e equação 7.2 são utilizadas para calcular a precisão e revocação respectivamente.

TAB. 7.1: Exemplo de precisão e revocação

Grupo 1	Grupo 2	Grupo 3	Grupo 4
documento 1_{k_1}	documento 7_{k_1}	documento 1_{k_2}	documento 1_{k_3}
documento 2_{k_1}	documento 8_{k_1}	documento 2_{k_2}	documento 2_{k_3}
documento 3_{k_1}	documento 9_{k_1}	documento 3_{k_2}	documento 3_{k_3}
documento 4_{k_1}		documento 4_{k_2}	documento 4_{k_3}
documento 5_{k_1}		documento 5_{k_2}	documento 5_{k_3}
documento 6_{k_1}		documento 6_{k_2}	documento 6_{k_3}
		documento 7_{k_2}	documento 7_{k_3}
		documento 8_{k_2}	documento 8_{k_3}
		documento 9_{k_2}	documento 9_{k_3}
		documento 10_{k_2}	documento 10_{k_3}
			documento 10_{k_1}
precisão = 1; revocação = 0.6	precisão = 1; revocação = 0.3	precisão = 1; revocação = 1	precisão = 0,909; revocação 1

$$\text{precisão} = \frac{\text{número de criptogramas homogêneos recuperados}}{\text{número de criptogramas recuperados}} = \frac{CH}{CR} \tag{7.1}$$

$$\text{revocação} = \frac{\text{número de criptogramas homogêneos recuperados}}{\text{criptogramas homogêneos da amostra}} = \frac{CH}{CI} \tag{7.2}$$

7.3.1 GRAU DE CONECTIVIDADES DOS VÉRTICES

A terceira métrica de avaliação, que corresponde a uma das novas métricas propostas nesse trabalho, diz respeito ao grau de conectividade dos documentos agrupados. Em consequência ao desenvolvimento do novo algoritmo de reconhecimento de padrões realizado nesse trabalho, foi possível observar características em relação aos agrupamentos dos criptogramas que não eram observadas quando eram utilizados os outros algoritmos de agrupamento desenvolvidos pelo GSI/IME. Uma dessas características diz respeito ao grau de conectividade entre os documentos agrupados. Através da observação das conectividades dos vértices dos subgrafos encontrados, foi possível perceber que existem diferenças entre grupos que quando avaliados apenas pelas métricas de precisão e revocação aparentam ser iguais.

Entende-se como grau de conectividade de um vértice v, $grau(v)$, o número de arestas incidentes a v. Se $grau(v) = 0$, então o vértice v é chamado de vértice isolado. Se $grau(v) = n - 1$, então o vértice v é chamado de vértice universal. Quando todos os n vértices de um grafo são vértices universais, sendo $n \geq 1$, o grafo é chamado de grafo completo.

Nesse contexto, essa métrica de avaliação tem como objetivo encontrar o máximo possível de vértices isolados, pois isso significaria que o documento representado por esse vértice não possui relacionamento algum com nenhum outro documento presente na amostra.

Por outro lado, quando existe um grande número de vértices que possuem altas conectividades, sendo o máximo um vértice universal, significa que os documentos avaliados possuem alta dependência entre si.

A fim de exemplificar a importância dessa métrica, considere o seguinte exemplo hipotético: cinco documentos foram cifrados com o algoritmo α utilizando duas chaves diferentes k_1 e k_2, totalizando uma amostra de dez criptogramas. No processo de agrupamento foram encontrados dois grupos de criptogramas conforme ilustrado na FIG 7.2.

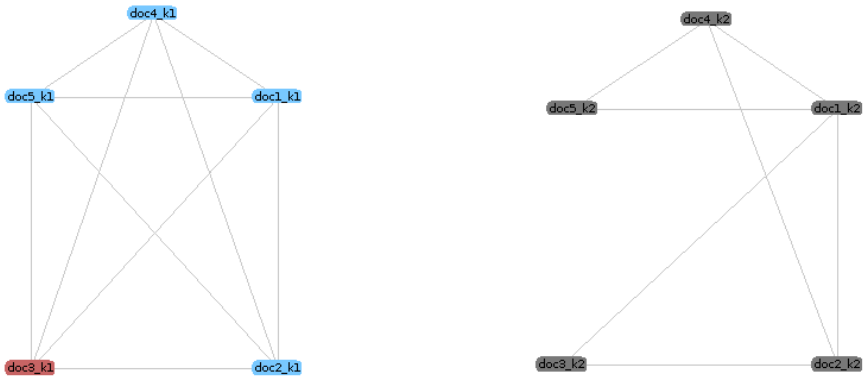

FIG. 7.2: Exemplo do grau de conectividade dos vértices

Em relação a esse exemplo, pode-se perceber a importância dessa métrica de avaliação, pois apesar de os dois grupos ilustrados apresentarem revocação e precisão igual a um, a conectividade de cada grupo é diferente. Os criptogramas cifrados com a chave k_1 apresentam conectividade máxima, portanto, cada vértice desse grupo é considerado como vértice universal, demonstrando dessa forma, um forte relacionamento entre os documentos desse grupo. Já os criptogramas cifrados com a chave k_2 apresentam uma conectividade mais fraca, sendo que o único vértice universal desse grupo é representado pelo documento $doc1\_k2$. Dessa forma, fica evidente a importância dessa métrica como avaliação complementar as métricas de revocação e precisão.

7.3.2 ENTROPIA AMOSTRAL

A quarta métrica de avaliação, que corresponde a uma das novas métricas propostas nesse trabalho, diz respeito a avaliação da entropia das amostras. Essa métrica tem por finalidade avaliar o grau de incerteza dos criptogramas quando comparados com os seus correspondentes textos em claro.

Segundo (DENNING, 1982), a entropia tem como objetivo medir a quantidade de informação contida em uma mensagem em relação a média de bites necessários para sua codificação. Em outras palavras, a entropia mede a incerteza do número de bites a serem deduzidos quando uma mensagem é distorcida em função de perdas em um canal de comunicação ou quando criptografadas. Sendo assim, a entropia é uma função de distribuição de probabilidade sobre o conjunto de todas as possíveis mensagens. Seja $X_1, ..., X_n$, n possíveis mensagens com probabilidade de ocorrência $p(X_1), ..., p(X_n)$, onde $\sum_{i=i}^{n} p(X_i) = 1$. A entropia da mensagem é definida pelo peso médio conforme definido na equação 7.3.

$$H(X) = \sum_{i=1}^{n} p(X_i) log_2 \frac{1}{p(X_i)} \tag{7.3}$$

Nesse contexto, considerando que os criptogramas em avaliação trabalham com mensagens em bloco de cento e vinte e oito bites, a modelagem para essas mensagens pode ser definida da seguinte forma: sendo $n = 128$ o tamanho do bloco, o número de possíveis mensagens é igual a 2^n. Sendo $X_1, ..., X_{2^n}$ as possíveis mensagens com distribuição de probabilidade uniforme $p(X_i) = \frac{1}{2^n}$ para $i = 1, ..., 2^n$. A entropia para essa distribuição é igual a $H(X) = 128$, pois: $H(X) = 2^n \left[\left(\frac{1}{2^n} \right) log_2 2^n \right]$.

Portanto, considerando que os algoritmos criptográficos apresentam as condições ideais, cada bloco teria a mesma probabilidade de ser utilizado, garantido a incerteza máxima da informação. Contudo, a entropia amostral, tem como objetivo a partir da análise da amostra redistribuir a probabilidade de cada mensagem. Para isso, é realizada a análise da distribuição dos pesos de hamming da amostra e então recalculadas as probabilidades conforma a equação 7.4.

$$p(X_i) = \frac{y}{tC_k^n} = \frac{y}{t\frac{n!}{k!(n-k)!}} \tag{7.4}$$

Tal que y é o número de blocos com peso de hamming k; t o total de blocos na amostra e C_k^n o número de blocos com peso de hamming igual a k em relação ao universo de 2^n. Para exemplificar considere o seguinte exemplo: Foram utilizados como texto em claro um conjunto de duzentos e cinco blocos, cuja distribuição dos pesos de hamming é ilustrada na FIG 7.3. Para a construção da amostra das cifras foi utilizado o algoritmo AES e a respectiva distribuição de hamming é ilustrada na FIG 7.4.

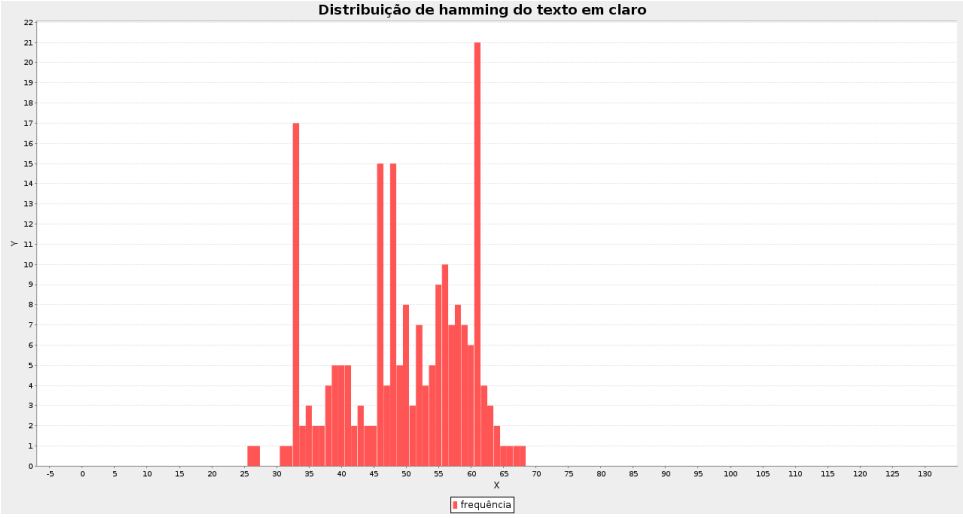

FIG. 7.3: Exemplo de distribuição de hamming do texto em claro

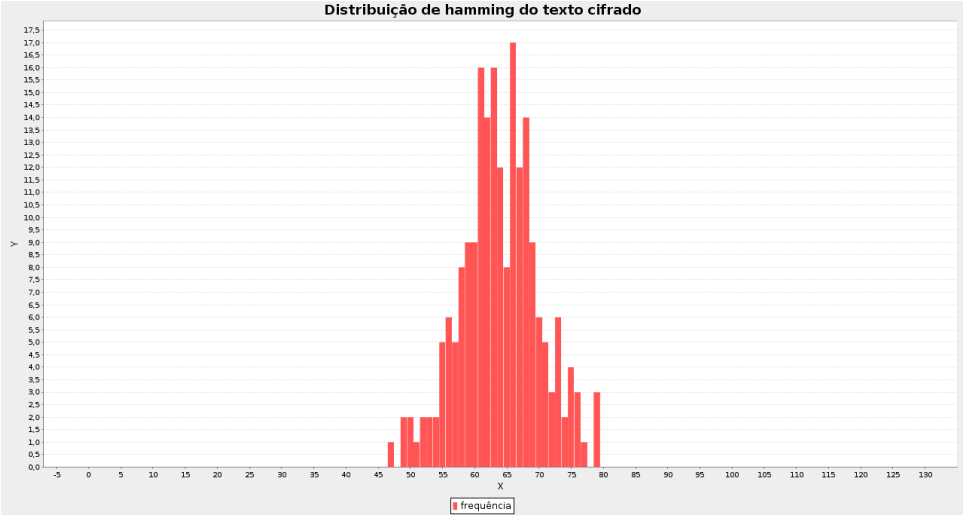

FIG. 7.4: Exemplo de distribuição de hamming do texto cifrado

Analisando as duas distribuições de hamming, a do texto em claro e a do texto cifrado, pode-se perceber que quando os textos em claro são submetidos a um algoritmo criptográfico, os blocos de saída tendem a ser mais equilibrados, ou seja, os blocos resultantes tendem a ficar nos grupos de hamming cujo número de blocos com essa característica é maior, aumentando dessa forma a incerteza das cifras. Em relação a entropia, a amostra dos textos em claro obteve $H(X) = 122.1$, e a amostra dos textos cifrados obteve $H(X) = 127.89$. Demonstrando o maior grau de incerteza dos criptogramas, a tal ponto que o valor da entropia da amostra dos criptogramas ficou próxima da entropia máxima.

7.4 EXPERIMENTOS

Esse seção tem como objetivo analisar o grau de dispersão dos criptogramas gerados pelo algoritmo AES modificado, que corresponde ao AES com as contribuições apresentadas nos capítulos 5 e 6, e realizar um estudo comparativo com os resultados obtidos pelos algoritmo AES, Twofish, Serpent, MARS e RC6. Para tal, foram realizados três conjuntos de experimentos. No primeiro conjunto foi utilizada uma amostra de trinta textos retirados da a Bíblia inglesa *(www.o-bible.com)*, cada texto possui 10240 bytes, totalizando uma amostra de 307200 bytes. Esse primeiro conjunto de experimento corresponde ao maior conjunto de dados utilizados nos trabalhos de (CARVALHO, 2006), (SOUZA, 2007) e (OLIVEIRA, 2011). O objetivo desse experimento foi verificar a dispersão dos criptogramas para a maior amostra utilizada nos trabalhos anteriores.

No segundo conjunto de experimento foi utilizada uma amostra de vinte e nove textos retirados do repositório *Open American National Corpus* (OANC) *americannationalcorpus.org/*, cada texto possui 40816 bytes, totalizando uma amostra de 1183664 bytes. O objetivo foi verificar o nível de dispersão dos criptogramas para uma amostra de dados maiores e com mais redundâncias. Entende-se como redundância o número de blocos repetidos na amostra.

No terceiro conjunto de experimento foi utilizada uma amostra de trinta e um textos retirados do repositório OANC, só que diferentemente das amostras do segundo experimento, a composição da atual amostra foi formada por documentos com assuntos diferentes, influenciando dessa forma para uma menor redundância da amostra. O tamanho de cada texto é de 61424 bytes, totalizando uma amostra de 1904144 bytes. O objetivo desse experimento foi demonstrar a influência dos parâmetros tamanho da amostra e redundância de blocos em relação o desempenho dos resultados de dispersão obtidos.

Uma característica que deve ser ressaltada, é que diferentemente dos trabalhos anteriores, como (CARVALHO, 2006), (SOUZA, 2007) e (OLIVEIRA, 2011), que construíram seus conjuntos de testes com o objetivo de encontrar padrões em criptogramas com a menor quantidade de informações possível, os experimentos realizados nesse trabalho é justamente o oposto, como o objetivo é verificar a dispersão dos padrões, então é mais conveniente que se utilize como dado de entrada texto com alta similaridade, pois uma vez que o algoritmo proposto consiga dispersa padrões nesse cenário de avaliação, significa que ele também conseguirá obter resultados satisfatórios para o cenário de experimentos

utilizados nos trabalhos citados.

Para a realização dos experimentos foi utilizado o processo de reconhecimento de padrões conforme apresentado na FIG 7.1. Todavia, na segunda etapa do processo, apenas o algoritmo de agrupamentos de criptogramas proposto nesse trabalho foi utilizado. Portanto, a FIG 7.1 pode ser adaptada para a FIG 7.5.

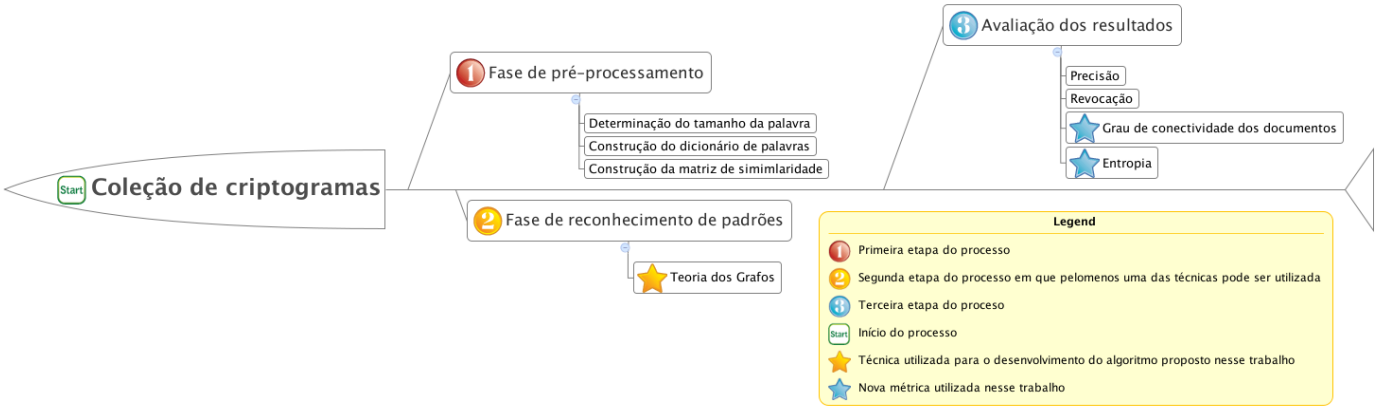

FIG. 7.5: Processo de reconhecimento de padrões utilizado nos experimentos

7.4.1 PRIMEIRO CONJUNTO DE EXPERIMENTO

A fim de realizar uma análise comparativa entre os resultados dos agrupamentos obtidos pelos criptogramas dos algoritmos em estudo, a amostra de texto em claro também foi submetida ao processo de reconhecimento de padrões. Dessa forma, foi possível comparar os resultados das métricas de avaliação e verificar a influência do texto em claro no processo de cifragem dos criptogramas.

Para a análise dos algoritmos foram utilizadas cinco chaves aleatórias geradas pelo *SecureRandom*, gerando um amostra de cento e cinquenta criptogramas para cada algoritmo. Quanto aos resultados, os criptogramas dos cinco algoritmos finalistas do concurso do AES foram agrupados corretamente gerando um grupo por chave utilizada, logo, as medidas de precisão e revocação de cada grupo foi igual a um. A FIG 7.6 ilustra os grupos encontrados para o algoritmo AES. Quanto aos grupos dos outros quatro algoritmo finalistas, o comportamento é exatamente o mesmo do grupos formados pelo algoritmo AES.

A justificativa para tal comportamento vem da análise do texto em claro. Como o agrupamento dos textos em claro foi único, ou seja, a precisão e a revocação foi igual a

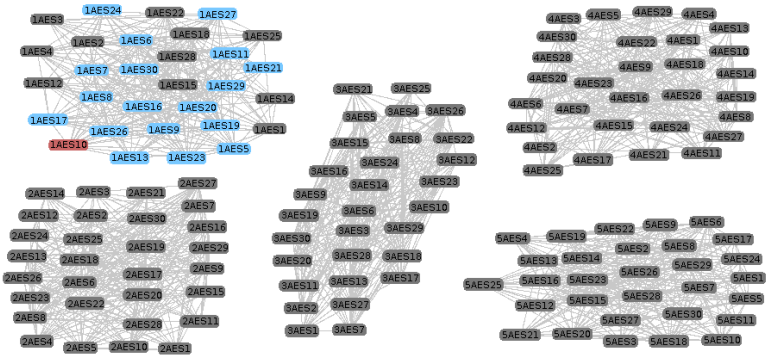

FIG. 7.6: Grupos encontrados dos criptogramas do AES

um, e como o modo de operação utilizado para a cifragem dos algoritmos foi o modo ECB, os blocos iguais da amostra do texto em claro serão transferidos para as cifras, portanto, a redundância existente na amostra de texto em claro é integralmente transferida para a amostra de cifras.

FIG. 7.7: Grupos encontrados dos criptogramas do AES modificado

Não obstante, esse comportamento não foi observado quando os textos em claro foram cifrados pelo algoritmo AES modificado, utilizando também o modo ECB. Nesse caso, o número de grupos encontrados conforme ilustrado na FIG 7.7, foi muito maior do que o esperado. Como foram utilizadas cinco chaves e o agrupamento dos textos em claro foi de apenas um grupo, assim como aconteceu para os outros algoritmos, o número de grupos esperados caso o padrão encontrado nos textos em claro fossem transferidos para as cifras, seria igual a cinco. Entretanto o número de grupos encontrados foi igual a cento e trinta, demonstrando que o algoritmo AES após as modificações realizadas nesse trabalho, é capaz de dispersar os padrões existentes nos textos em claro, mesmo quando o modo de operação utilizado seja o ECB.

A TAB 7.2 ilustra as medidas de precisão e revocação dos grupos resultantes do algoritmo AES modificado.

TAB. 7.2: Precisão e revocação dos grupos resultantes do algoritmo AES modificado

Precisão	Revocação	Número de grupos
1	0,033333333	120
1	0,066666667	5
1	0,133333333	5

Os resultados mostram que o algoritmo AES modificado conseguiu dispersar os padrões existentes nos textos em claro, entretanto pelo fato de a precisão sempre ser igual a um, ou seja, nenhum criptograma com chaves diferentes ficou no mesmo grupo, demonstrando que ao menos para essa amostra as chaves influenciaram em formações de subgrupos sem interseções.

Quanto à análise do grau de conectividade dos documentos, essa pode ser observada na TAB 7.3.

TAB. 7.3: Grau de conectividade dos vértices para textos de 10240 bytes

Documento	Texto Claro	AES Modificado	AES	Twofish	Serpent	MARS	RC6
Texto1	15	0	15	15	15	15	15
Texto2	14	0	14	14	14	14	14
Texto3	20	2	20	20	20	20	20
Texto4	13	0	13	13	13	13	13
Texto5	22	1	22	22	22	22	22
Texto6	20	1	20	20	20	20	20
Texto7	19	0	19	19	19	19	19
Texto8	20	1	20	20	20	20	20
Texto9	19	0	19	19	19	19	19
Texto10	13	0	13	13	13	13	13
Texto11	11	0	11	11	11	11	11
Texto12	17	0	17	17	17	17	17
Texto13	20	0	20	20	20	20	20
Texto14	24	0	24	24	24	24	24
Texto15	21	0	21	21	21	21	21
Texto16	19	2	19	19	19	19	19
Texto17	25	0	25	25	25	25	25
Texto18	18	0	18	18	18	18	18
Texto19	14	0	14	14	14	14	14
Texto20	14	0	14	14	14	14	14
Texto21	11	0	11	11	11	11	11
Texto22	15	0	15	15	15	15	15
Texto23	22	0	22	22	22	22	22
Texto24	17	0	17	17	17	17	17
Texto25	20	1	20	20	20	20	20
Texto26	18	0	18	18	18	18	18
Texto27	23	0	23	23	23	23	23
Texto28	12	0	12	12	12	12	12
Texto29	20	0	20	20	20	20	20
Texto30	14	0	14	14	14	14	14

Em relação a essa análise foi possível observar que a conectividade existente nos documentos da amostra dos textos em claro é transferida integralmente para os criptogramas cifrados pelos algoritmos AES, Twofish, Serpent, MARS e RC6. Os resultados contidos na tabela ilustram a conectividade dos documentos para apenas uma das chaves utilizadas, pois o comportamento observado foi idêntico para as outras chaves. Portanto, para esse

experimento, fica evidente que a chave não influência no grau de relacionamento dos criptogramas. Quanto a análise da conectividade dos criptogramas gerados pelo algoritmo AES modificado, é possível perceber que o grau de conectividade dos criptogramas quando comparados com a mostra de texto em claro, foi bastante reduzida. Isso significa que o algoritmo conseguiu diminuir os padrões existentes nas mensagens cifradas.

A última análise realizada diz respeito à entropia da amostra. Essa análise tem como objetivo verificar o grau de incerteza produzida pelos criptogramas em relação a amostra utilizada. Para tal, foi calculado a entropia da amostra dos textos em claro em função da distribuição de hamming e comparada com os resultados obtidos para as amostras de cada algoritmo e de cada chave.

A FIG 7.8 ilustra a distribuição de hamming correspondente a amostra de textos em claro. Já a FIG 7.9 corresponde a distribuição de hamming dos criptogramas cifrados pelo algoritmo AES utilizando a chave 1.

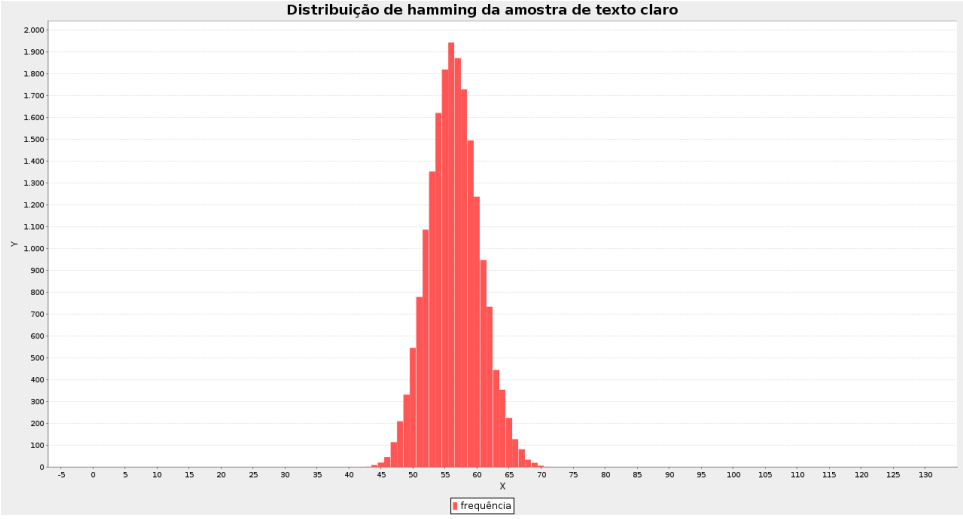

FIG. 7.8: Distribuição de hamming da amostra de texto em claro

Em relação a essas duas distribuições, pode-se perceber que os pesos de hamming correspondentes ao texto em claro estão mais deslocados para a esquerda, sendo seu ponto máximo o hamming cinquenta e seis. Quando a amostra é cifrada, a distribuição sofre um deslocamento para a direita e um alargamento nas extremidades, isso significa que os blocos cifrados são melhores distribuídos e seus pesos de hamming se concentram próximos da faixa de melhor balanceamento, que corresponde aos grupos de maior combinações C_k^n. Portanto, é possível observar que devida a essa redistribuição, o grau de incerteza dos criptogramas tende a aumentar. As demais distribuições correspondentes aos outros

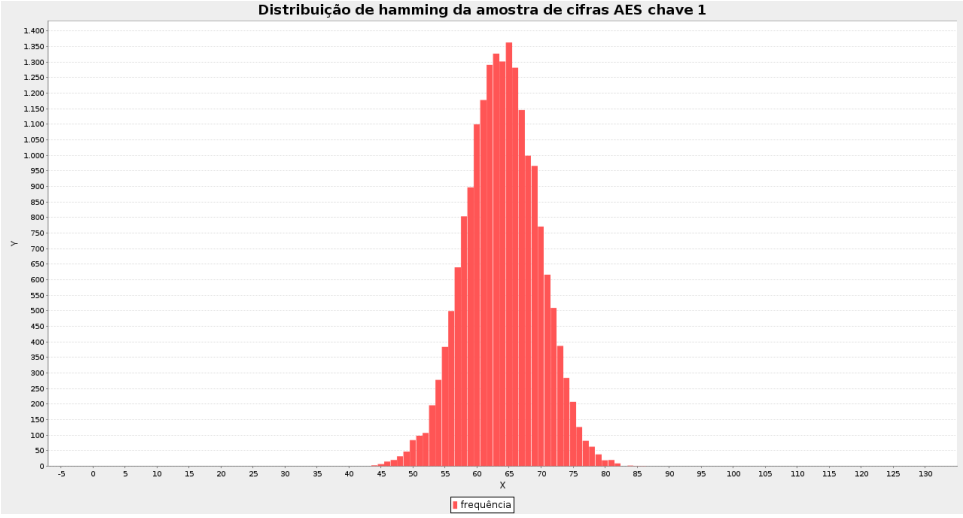

FIG. 7.9: Distribuição de hamming da amostra de criptogramas AES chave 1

algoritmo e chaves, apresentaram um comportamento semelhante ao da FIG 7.9, essa semelhança pode ser observada em função da entropia calculada.

A TAB 7.4 ilustra a entropia calculada para cada chave e para cada algoritmo utilizado nesse experimento.

Os resultados mostram que a variação da entropia entre os algoritmos e as chaves utilizadas é muito pequena, demonstrando o ótimo nível de grau de incerteza ainda que existam padrões. Entretanto pode-se perceber uma leve diferença entre a entropia do texto em claro e as demais. Esse comportamento se dá justamente pela observação do deslocamento dos pesos de hamming para o centro e o alargamento das caudas. Em relação ao comportamento das entropias referentes aos algoritmos e as chaves, pode-se perceber que os valores calculados ficaram muito próximos da entropia máxima, que é igual a cento e vinte e oito, a variação entre um valor e outro, geralmente acontece na terceira ou quarta casa decimal. Essa característica justifica o porque do comportamento gráfico ser tão semelhante para todos os algoritmo e as chaves. Contudo, como os blocos de cada algoritmo e de cada chave utilizada não apresentam interseções, isso significa que apesar de os blocos ocuparem as mesmas faixas de pesos de hamming, ainda assim eles não se misturam. Entende-se que a chave secreta seja responsável por esse comportamento, entretanto, como o universo é finito, 2^n, se o conjunto de entrada fosse maior que o universo, certamente essas interseções iriam ocorrer.

Com os resultados do primeiro experimento foi possível constatar que o algoritmo AES após realizadas as modificações sugeridas nos capítulos 5 e 6, conseguiu dispersar os

TAB. 7.4: Entropia calculada

Amostra	H(x)
Texto em claro	126.571428
AES modificado chave 1	127.998375
AES modificado chave 2	127.997447
AES modificado chave 3	127.998650
AES modificado chave 4	127.997687
AES modificado chave 5	127.997654
AES chave 1	127.997649
AES chave 2	127.998117
AES chave 3	127.998301
AES chave 4	127.998256
AES chave 5	127.998083
Twofish chave 1	127.998225
Twofish chave 2	127.998794
Twofish chave 3	127.998501
Twofish chave 4	127.997585
Twofish chave 5	127.998361
Serpent chave 1	127.998250
Serpent chave 2	127.998556
Serpent chave 3	127.998090
Serpent chave 4	127.997932
Serpent chave 5	127.998028
MARS chave 1	127.997332
MARS chave 2	127.998073
MARS chave 3	127.997657
MARS chave 4	127.998179
MARS chave 5	127.998428
RC6 chave 1	127.998155
RC6 chave 2	127.998472
RC6 chave 3	127.998689
RC6 chave 4	127.997753
RC6 chave 5	127.998464

padrões encontrados no texto em claro a tal ponto que o agrupamento dos criptogramas não conseguisse dar informações suficientes para a identificação do algoritmo utilizado no processo de cifragem, mesmo que o modo de operação utilizado fosse o modo ECB.

7.4.2 SEGUNDO CONJUNTO DE EXPERIMENTO

O segundo conjunto de experimento teve como objetivo avaliar o desempenho do algoritmo AES modificado quando os textos em claro apresentam um maior nível de redundância.

Para isso foram utilizados vinte e nove textos do repositório OANC totalizando uma amostra de 1183664 bytes. O mesmo processo aplicado no primeiro experimento foi utilizado, portanto, a FIG 7.10 ilustra os grupos encontrados para o algoritmo AES. Assim como no primeiro experimento, os grupos dos algoritmos Twofish, Serpent, MARS e RC6, apresentão o mesmo comportamento do algoritmo AES, logo, FIG 7.10 também os representam.

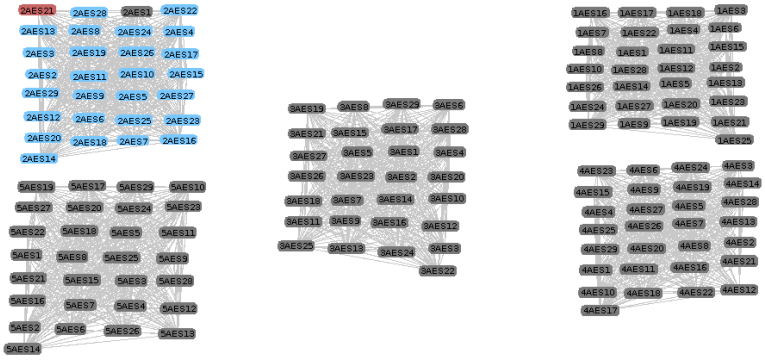

FIG. 7.10: Grupos encontrados dos criptogramas do AES

A justificativa para tal comportamento é a mesma da argumentada no primeiro experimento já que a única variável que foi alterada do primeiro para o segundo experimento foi o conjunto de textos em claro. Portanto, pode-se perceber que apesar dos agrupamentos referentes aos cinco algoritmo finalistas do concurso do AES apresentarem revocação e precisão iguais a um, existe uma diferença significativa que pode ser observada a partir da análise do comportamento da FIG 7.6 e FIG 7.10. Os grupos encontrados no segundo experimento (vide FIG 7.10) apresentam uma conectividade muito maior do que os grupos apresentados no primeiro experimento (vide FIG 7.6). Esse comportamento demonstra a importância da métrica grau de conectividade dos vértices, haja vista que essa métrica indica que os grupos encontrados no segundo experimento são muito mais similares em relação aos grupos do primeiro experimento, apesar de as medidas de revocação e precisão indicarem que eles são aparentemente iguais. Sendo assim, fica evidente a importância da métrica grau de conectividade dos vértices como uma medida de avaliação complementar as métricas de revocação e precisão.

Com relação aos grupos encontrados para os criptogramas gerados pelo algoritmo AES modificado, esses mais uma vez apresentaram uma dispersão satisfatória. A FIG 7.11 ilustra os grupos encontrados para esse algoritmo.

Foram encontrados trinta grupos e as respectivas medidas de precisão e revocação

FIG. 7.11: Grupos encontrados para o algoritmo AES modificado

encontram-se na TAB 7.5.

TAB. 7.5: Precisão e revocação dos grupos resultantes do algoritmo AES modificado

Precisão	Revocação	Número de grupos
1	0,034482759	20
1	0,068965517	5
1	0,793103448	5

Observando os resultados de precisão e revocação a presentados na TAB 7.5, uma análise natural seria imaginar que pelo menos um grupo de cada chave apresenta uma revocação a ponto de ser considerada um padrão identificável. De fato o número de criptogramas aglomerados em um mesmo grupo foi um tanto expressivo, entretanto, se observada as conectividades, pode-se perceber que elas foram significativamente enfraquecidas. Para uma análise mais criteriosa a TAB 7.6 ilustra o grau de conectividade de cada documento.

Em relação a conectividade de cada documento, pode-se observar que quase todos os textos em claro são caracterizados como vértices universais. Essa característica demonstra a maior redundância dos textos em claro do segundo experimento em relação aos textos utilizados no primeiro experimento. Logo, isso explica o porque do relacionamento entre os criptogramas dos algoritmos AES, Twofish, Serpent, MARS e RC6, apresentarem uma conexão muito mais forte no segundo experimento. Não obstante, em relação aos criptogramas gerados pelo algoritmo AES modificado, pode-se perceber que apesar de certos grupos apresentarem um número razoável de criptogramas, o conexão entre eles foi enfraquecida a tal ponto que a maior conexão existente entre as cifras foi de grau quatro, e para os textos em claro a menor conexão é de grau vinte e sete. Essa comportamento comprova mais uma vez a eficiência das modificações sugeridas nos capítulos 5 e 6.

TAB. 7.6: Grau de conectividade dos vértices para textos de 40816 bytes

Documento	Texto Claro	AES Modificado	AES	Twofish	Serpent	MARS	RC6
Texto1	28	3	28	28	28	28	28
Texto2	28	1	28	28	28	28	28
Texto3	28	2	28	28	28	28	28
Texto4	27	0	27	27	27	27	27
Texto5	28	1	28	28	28	28	28
Texto6	28	0	28	28	28	28	28
Texto7	28	1	28	28	28	28	28
Texto8	28	3	28	28	28	28	28
Texto9	28	3	28	28	28	28	28
Texto10	28	1	28	28	28	28	28
Texto11	27	3	27	27	27	27	27
Texto12	28	1	28	28	28	28	28
Texto13	28	3	28	28	28	28	28
Texto14	28	3	28	28	28	28	28
Texto15	28	1	28	28	28	28	28
Texto16	28	1	28	28	28	28	28
Texto17	28	0	28	28	28	28	28
Texto18	28	4	28	28	28	28	28
Texto19	28	0	28	28	28	28	28
Texto20	27	1	27	27	27	27	27
Texto21	28	3	28	28	28	28	28
Texto22	28	1	28	28	28	28	28
Texto23	28	1	28	28	28	28	28
Texto24	28	2	28	28	28	28	28
Texto25	28	2	28	28	28	28	28
Texto26	28	1	28	28	28	28	28
Texto27	28	3	28	28	28	28	28
Texto28	28	1	28	28	28	28	28
Texto29	27	2	27	27	27	27	27

Por fim a análise da entropia amostral apresentou resultados semelhantes aos obtidos no primeiro experimento, essa característica permite pensar que os padrões encontrados nos textos em claro não tem grandes influências no processo de transformação dos blocos para os textos cifrados, no que se refere ao grau de incerteza da entropia calculada. A FIG 7.12 e FIG 7.13, referem-se a distribuição de hamming da amostra dos textos em claro e das cifras geradas pelo algoritmo AES modificado respectivamente.

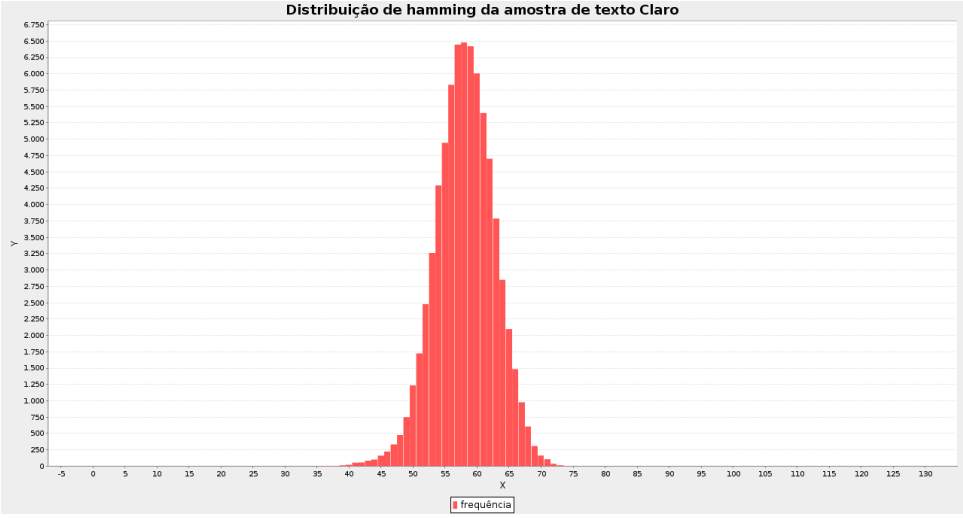

FIG. 7.12: Distribuição de hamming da amostra de texto em claro

Como pode ser observado, novamente durante o processo de cifragem houve uma redistribuição dos blocos cifrados de uma forma que houvesse um deslocamento da dis-

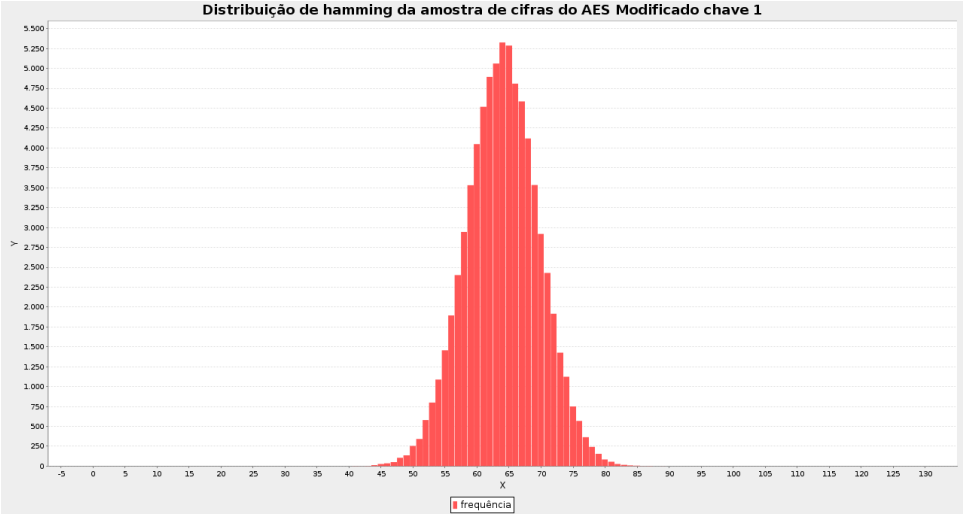

FIG. 7.13: Distribuição de hamming da amostra de criptogramas AES modificado

tribuição para o centro e um leve alargamento da cauda. Esse comportamento também foi observado para todos os outros algoritmos e chaves utilizadas, sendo essa característica expressada numericamente na TAB 7.7 que apresenta a entropia calculada para cada chave e algoritmo avaliado.

Analisando os resultados pode-se perceber que a entropia do conjunto de textos em claro foi maior do que a entropia dos textos em claro utilizados no primeiro experimento. Nesse contexto, acredita-se que em relação ao grau de incerteza, todos os algoritmos avaliados conseguem um bom desempenho e, partindo da hipótese de que o objetivo dos algoritmos criptográficos é garantir a confidencialidade da mensagem, pelo menos quanto a grau de incerteza da informação todos os algoritmos apresentaram resultados satisfatórios, contudo, quanto a análise de padrões, mesmo sem saber do que se trata a informação, apenas o algoritmo AES modificado conseguir apresentar bons resultados.

7.4.3 TERCEIRO CONJUNTO DE EXPERIMENTO

O terceiro e último experimento teve como objetivo avaliar a influência do tamanho da amostra e nível de redundância. Para isso foram utilizados textos do repositório OANC cujo conteúdo dos textos tratassem de assuntos diferentes, sendo assim, foram utilizados trinta e um textos tratando de três assuntos diferentes, totalizando uma amostra de 1904144 bytes. A finalidade de utilizar textos de assuntos diferentes é verificar se mesmo usando uma amostra maior do que a utilizada no segundo experimento, porém com um grau de redundância menor entre os textos, se os resultados dos agrupamentos seriam

TAB. 7.7: Entropia calculada

Amostra	H(x)
Texto em claro	127.158751
AES modificado chave 1	127.999549
AES modificado chave 2	127.999262
AES modificado chave 3	127.999527
AES modificado chave 4	127.999525
AES modificado chave 5	127.999577
AES chave 1	127.999603
AES chave 2	127.999602
AES chave 3	127.999514
AES chave 4	127.999541
AES chave 5	127.999471
Twofish chave 1	127.999594
Twofish chave 2	127.999334
Twofish chave 3	127.999264
Twofish chave 4	127.999351
Twofish chave 5	127.999396
Serpent chave 1	127.999540
Serpent chave 2	127.999574
Serpent chave 3	127.999547
Serpent chave 4	127.999402
Serpent chave 5	127.999649
MARS chave 1	127.999395
MARS chave 2	127.999454
MARS chave 3	127.999543
MARS chave 4	127.999536
MARS chave 5	127.999533
RC6 chave 1	127.999516
RC6 chave 2	127.999422
RC6 chave 3	127.999513
RC6 chave 4	127.999449
RC6 chave 5	127.999393

melhores ou piores, ou seja, verificar qual a maior influência, o grau de redundância ou o tamanho da amostra.

Para a avaliação foram realizadas as mesmas etapas dos experimentos anteriores, portanto, a FIG 7.14 e FIG 7.15 ilustram os grupos encontrados para o algoritmo AES e o para o algoritmo AES modificado respectivamente.

Os resultados obtidos pelos criptogramas gerados pelos algoritmos Twofish, Serpent, MARS e RC6 também apresentam o mesmo comportamento ilustrado na FIG 7.14. Sendo

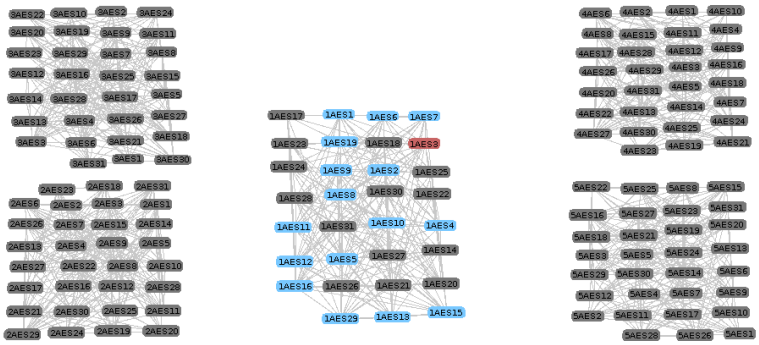

FIG. 7.14: Grupos encontrados para o algoritmo AES

assim, fazendo uma análise comparativa com os resultados obtidos no segundo experimento, pode-se perceber que os grupos do terceiro experimento também apresentaram revocação e precisão iguais a um. Entretanto, o grau de conectividade entre os criptogramas do segundo experimento é muito maior. Demonstrando dessa forma que o parâmetro redundância tem maior influência nos resultados do que o tamanho da amostra propriamente dito. Logicamente, que se esses dois parâmetros trabalharem em conjunto, ou seja, se for construída uma amostra com uma grande quantidade de textos e esses textos apresentarem bastante redundância, as chances de reconhecimento dos criptogramas é maior.

Essa característica também influenciou nos resultados obtidos pelo algoritmo AES modificado, em que a FIG 7.15 ilustra um maior número de vértices isolados.

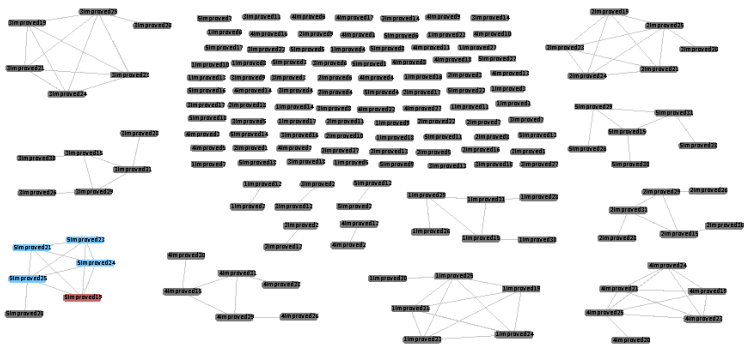

FIG. 7.15: Grupos encontrados para o algoritmo AES melhorado

Em números foram encontrados cem grupos com medidas de precisão e revocação conforme ilustrada na TAB 7.8.

Nesse experimento é ressaltada a importância da métrica grau de conectividade dos

TAB. 7.8: Precisão e revocação dos grupos resultantes do algoritmo AES modificado

Precisão	Revocação	Número de grupos
1	0,032258065	85
1	0,064516129	5
1	0,193548387	10

documentos, pois apesar de a medida de revocação informar que existem dez grupos com revocação igual a $0,193548387$, quando esses grupos são analisados em relação a conectividade, pode-se perceber que cinco desses grupos apresentam um comportamento de conectividade fraca, e os outros cinco apresentam um comportamento de grupos um pouco mais conectados.

A TAB 7.9 ilustra a conectividade dos documentos para cada algoritmo.

TAB. 7.9: Grau de conectividade dos vértices para textos de 61424 bytes

Documento	Texto Claro	AES Modificado	AES	Twofish	Serpent	MARS	RC6
Texto1	12	0	12	12	12	12	12
Texto2	12	1	12	12	12	12	12
Texto3	16	0	16	16	16	16	16
Texto4	11	0	11	11	11	11	11
Texto5	18	0	18	18	18	18	18
Texto6	13	0	13	13	13	13	13
Texto7	10	0	10	10	10	10	10
Texto8	16	0	16	16	16	16	16
Texto9	19	0	19	19	19	19	19
Texto10	12	0	12	12	12	12	12
Texto11	12	0	12	12	12	12	12
Texto12	15	1	15	15	15	15	15
Texto13	14	0	14	14	14	14	14
Texto14	28	0	14	14	14	14	14
Texto15	14	3	14	14	14	14	14
Texto16	17	0	17	17	17	17	17
Texto17	8	0	8	8	8	8	8
Texto18	12	0	12	12	12	12	12
Texto19	12	4	12	12	12	12	12
Texto20	12	1	12	12	12	12	12
Texto21	12	4	12	12	12	12	12
Texto22	9	0	9	9	9	9	9
Texto23	9	4	9	9	9	9	9
Texto24	14	4	14	14	14	14	14
Texto25	12	5	12	12	12	12	12
Texto26	15	1	15	15	15	15	15
Texto27	15	0	15	15	15	15	15
Texto28	16	1	16	16	16	16	16
Texto29	16	3	16	16	16	16	16
Texto30	13	1	13	13	13	13	13
Texto31	14	3	14	14	14	14	14

Analisando os resultados pode-se perceber que o número de vértices isolados gerados pelo algoritmo AES modificado foi maior quando comparado com os seus respectivos vértices do segundo experimento. Esse comportamento é justamente o reflexo da redundância dos textos, pois como no segundo experimento a maioria dos vértices correspondentes aos documentos dos textos em claro eram vértices universais, o grau de dispersão necessário para gerar vértices isolados teria que ser máximo. Entretanto, como no terceiro experimento, o grau de conectividade dos textos em claro eram menores, foi possível obter um maior número de vértices isolados. Demonstrando dessa forma a influência direta do grau de redundância dos textos para o desempenho da dispersão dos criptogramas gerados pelo

algoritmo AES modificado.

Em relação a entropia, foi observado o mesmo comportamento dos experimentos anteriores. Havendo uma melhor distribuição do blocos cifrados em comparação com a distribuição dos textos em claro. AS FIG 7.16 e FIG 7.17 apresentam a distribuição de hamming dos textos em claro e das cifras geradas pelo algoritmo AES modificado respectivamente.

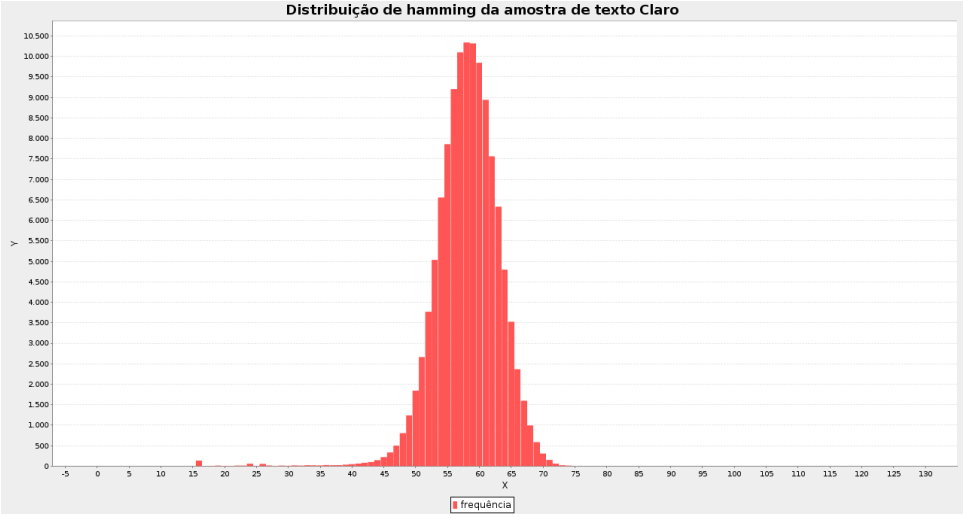

FIG. 7.16: Distribuição de hamming da amostra de texto em claro

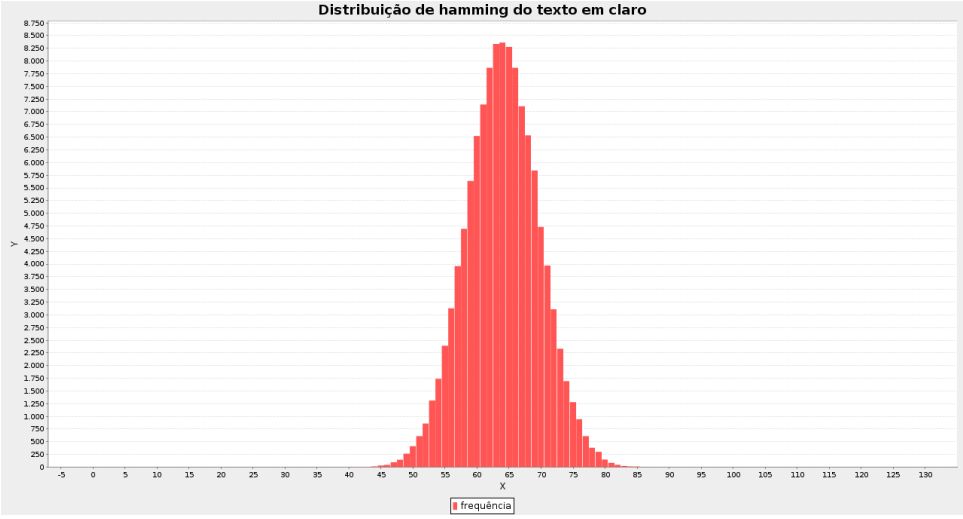

FIG. 7.17: Distribuição de hamming da amostra de criptogramas AES modificado

Os resultados da entropia calculada para cada algoritmo e chave pode ser observado na TAB 7.10.

TAB. 7.10: Entropia calculada

Amostra	H(x)
Texto em claro	127.136004
AES modificado chave 1	127.999673
AES modificado chave 2	127.999720
AES modificado chave 3	127.999689
AES modificado chave 4	127.999624
AES modificado chave 5	127.999674
AES chave 1	127.999788
AES chave 2	127.999727
AES chave 3	127.999681
AES chave 4	127.999674
AES chave 5	127.999766
Twofish chave 1	127.999818
Twofish chave 2	127.999672
Twofish chave 3	127.999669
Twofish chave 4	127.999518
Twofish chave 5	127.999648
Serpent chave 1	127.999730
Serpent chave 2	127.999612
Serpent chave 3	127.999731
Serpent chave 4	127.999744
Serpent chave 5	127.999725
MARS chave 1	127.999715
MARS chave 2	127.999612
MARS chave 3	127.999596
MARS chave 4	127.999726
MARS chave 5	127.999681
RC6 chave 1	127.999672
RC6 chave 2	127.999681
RC6 chave 3	127.999574
RC6 chave 4	127.999597
RC6 chave 5	127.999624

Analisando os resultados pode-se perceber que os valores também ficaram bem próximos dos resultados obtidos nos experimentos anteriores, entretanto, nesse experimento os resultados foram um pouco melhores. Portanto, os valores de entropia obtidos no terceiro experimento indica que o grau de incerteza dos criptogramas tende a ficar melhor quando o tamanho da amostra utilizada é maior, logo, as condições ideais para a melhor incerteza dos criptogramas é combinar amostras maiores com um baixo grau de redundância. Além disso, o comportamento gráfico também foi semelhante, ou seja, todas as combinação de

algoritmos e chaves utilizadas apresentaram uma distribuição de pesos de hamming com a média centrada em $n/2$ e uma cauda variando entre quarenta e cinco a oitenta e cinco.

7.5 CONCLUSÕES DO CAPÍTULO

Esse capítulo apresentou as contribuições desse trabalho no que se refere ao processo de reconhecimento de padrões em criptogramas. Como contribuição foi apresentado o algoritmo de agrupamento que utiliza teoria dos grafos, sendo sua principal destaque em relação aos demais algoritmo já propostos pelo grupo GSI/IME o tempo de processamento necessário para apresentar os mesmos resultados referentes a qualidade dos grupos encontrados. Além disso, ainda nas contribuições para o processo de reconhecimento de padrões, também foram apresentadas duas novas métricas de avaliação que são: grau de conectividade dos documentos e entropia amostral. A primeira tem como objetivo ser uma medida de avaliação complementar as medidas de precisão e revocação, com a finalidade de mostrar comportamentos diferentes quando em certos casos essas duas medidas acusam que dois grupos possam ser iguais. O grau de conectividade pode apresentar certas características que podem diferenciar tais grupos. Quanto a entropia amostral, essa métrica tem por finalidade quantificar o grau de incerteza dos criptogramas levando em consideração a distribuição de hamming observada.

Ademais, esse capítulo também apresentou a avaliação do algoritmo AES levando em consideração as modificações sugeridas nos capítulos 5 e 6. Em relação aos resultados foi possível observar através dos três conjuntos de experimentos realizados, que o algoritmo apresentou resultados satisfatórios. Nos experimentos também foi realizada uma análise comparativa com as cinco cifras finalistas do concurso do AES, e os resultados demonstraram que a única cifra dentre as comparadas, que conseguiu dispersar os padrões existentes nos textos em claro utilizando o modo de operação ECB, foi o algoritmo AES com as modificações propostas. Portanto, considera-se que o objetivo principal desse trabalho foi alcançado, objetivo este que era o de conseguir realizar modificações no algoritmo AES a fim de evitar que as cifras fossem reconhecidas mesmo que o modo de operação utilizado fosse o ECB.

8 CONCLUSÃO

Este trabalho apresentou uma nova metodologia de construção de caixas-S com a finalidade de garantir a maior segurança aos algoritmo que por ventura vierem a utilizá-la. Para avaliar sua eficiência em relação ao grau de incerteza dos valores obtidos a partir dos valores de entrada, foi utilizado um conjunto de métricas de avaliação conforme descritas no capítulo 3, o resultado desta avaliação demonstrou que a nova caixa-S apresenta bons níveis de aleatoriedade e não linearidade. Para avaliar a eficiência da caixa-S no que se refere ao nível de dispersão de padrões em criptogramas, a caixa-S foi acoplada ao algoritmo AES e conforme apresentado no capítulo 7, foram realizados testes que demonstraram que com a utilização da caixa-S, o algoritmo AES modificado foi capaz de impedir a transmissão de assinaturas mesmo utilizando o modo de operação ECB.

Outra contribuição deste trabalho diz respeito às modificações realizadas na função de expansão de chave do algoritmo AES com o objetivo de melhorar a segurança da função segundo a metodologia de classificação proposto por (CARTER, 1999), realizadas as modificações, a função possou a a fazer parte da categoria de funções de expansão de chaves que impossibilite ou dificulte bastante o conhecimento da chave secreta a partir de uma subchave.

Essas duas contribuições surgiram após uma análise do comportamento do algoritmo AES em relação ao teórico ataque DFA proposto por (GIRAUD, 2004). A partir da análise foi possível observar que as duas principais funções que permitiam o sucesso do ataque eram justamente a caixa-S e a função de expansão de chave do algoritmo. Portanto, a partir daí foram elaboradas as duas novas funções descritas anteriormente.

Outra contribuição proveniente da análise do algoritmo AES por meio do ataque teórico DFA foi a constatação da afirmação de (DUNKELMAN, 2010) de que ausência da transformação *MixColumns* na última rodada pode diminuir a complexidade do ataque.

Além disso, este trabalho também apresentou um novo algoritmo de reconhecimento de padrões em criptogramas que foi utilizado para os testes realizados no capítulo 7. Quanto aos testes, foi realizada uma análise comparativa entre os cinco algoritmo finalistas do concurso do AES, e os resultados mostraram que todos os algoritmo apresentaram o

mesmo resultado em relação a transmissão dos padrões existentes nos textos em claro para os textos cifrados. Esse comportamento deu indícios de que os cinco algoritmos avaliados não são capazes de eliminar as assinaturas provenientes dos textos em claro, ou que de fato foi observado através das métricas de revocação, precisão e grau de conectividade. Contudo, a métrica de entropia amostral também avaliou outra característica interessante dos algoritmos, os resultados dessa métrica mostraram que apesar de os padrões serem transmitidos para a cifras, o grau de incerteza dos criptogramas é quase perto do ideal. Esse comportamento foi observado para todos os cinco algoritmos e chaves utilizadas.

As métricas grau de conectividade dos documentos e entropia amostral também fazem parte da contribuição deste trabalho, a utilização dessas métricas possibilitaram a análise e observação de comportamentos que não eram observados quando utilizadas apenas as métricas de revocação e precisão.

Com relação as chaves, como o grau de incerteza dos criptogramas medidos a partir da métrica entropia amostral eram muito semelhantes, então, graficamente o comportamento da distribuição de hamming dos blocos cifrados produzidos pelas combinações de algoritmos e chaves demonstraram estarem todos na mesma faixa de distribuição. Porém, sem haver nenhuma intersecção entre blocos cifrados por chaves ou algoritmos diferentes. Esse comportamento dá indícios de que dentro do universo de blocos de 2^{128} existem, a priori, partições que são formadas a partir da combinação de chave e algoritmo utilizado. Entretanto, sabendo que o universo é finito, certamente existem possíveis combinações de textos em claro que podem a partir da combinação de algoritmo e chaves utilizadas, forçarem a interseção entre esses grupos. O fato intrigante é que empiricamente partindo de amostras de textos em claros com significado, esse comportamento ainda não foi observado. Todavia, para demonstrar que esse comportamento existe, elaborou-se o seguinte exemplo:

Dois blocos de textos em claro são cifrados utilizando os algoritmos A e B e a mesma chave k. Como resultado são gerados quatro blocos cifrados, formando dois grupos de criptogramas. Cada grupo contém apenas criptogramas gerados por um dos algoritmos; não há mistura, no mesmo grupo, de criptogramas gerados por algoritmos diferentes.

Então, os dois blocos cifrados pelo algoritmo A são decifrados utilizando o algoritmo B e a chave k, gerando mais dois blocos de texto em claro, porém aparentemente sem significado algum.

Esses dois novos blocos são cifrados pelos algoritmos A e B, ambos utilizando a chave

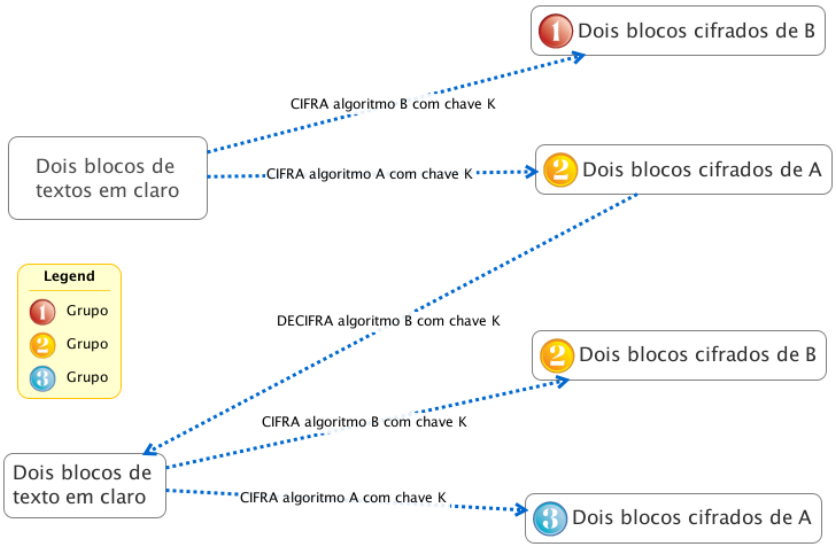

FIG. 8.1: Exemplo de intersecção de blocos por algoritmos

k.

Agora, quando essas cifras são submetidas ao reconhecimento de padrões, são encontrados três grupos, sendo que o segundo grupo conforme ilustrado na 8.1 apresenta cifras tanto do algoritmo A como do algoritmo B. Demonstrando dessa forma a interseção de blocos.

8.1 TRABALHOS FUTUROS

Os resultados desta dissertação sugerem como trabalhos futuros:

a) Desenvolvimento de novas funções de transformação para composição dos vértices da caixa-S.

b) Acoplagem da caixa-S aos outros algoritmos finalistas do concurso do AES para verificar o desempenho da dispersão de padrões desses algoritmos quando estiverem utilizando a caixa-S.

c) Avaliação da resistência da caixa-S em relação a outros ataques DFAs e até mesmo aos ataques clássicos como o diferencial e o linear.

d) Desenvolvimento de um novo algoritmo de reconhecimento de padrões que consiga agrupar os criptogramas dispersos em função das funções desenvolvidas neste trabalho.

e) Construção de experimentos para verificar a existência de interseções de criptogramas de algoritmos ou chaves diferentes a partir de amostras de textos com significados.

f) Estudo minucioso da caixa-S para encontrar caminhos aleatórios equivalentes.

9 REFERÊNCIAS BIBLIOGRÁFICAS

BIEHL, I., MEYER, B. e MÜLLER, V. **Differential Fault Attacks on Elliptic Curve Cryptosystems**. Em *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '00, págs. 131–146, London, UK, 2000. Springer-Verlag. ISBN 3-540-67907-3. URL http://portal.acm.org/citation.cfm?id=646765.704124.

BIHAM, E. e SHAMIR, A. **Differential Cryptanalysis of DES-like Cryptosystems**. Em *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '90, págs. 2–21, London, UK, UK, 1991. Springer-Verlag. ISBN 3-540-54508-5. URL http://dl.acm.org/citation.cfm?id=646755.705229.

BIHAM, E. e SHAMIR, A. **Differential Fault Analysis of Secret Key Cryptosystems**. Em *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, págs. 513–525. Springer, 1997. ISBN 3-540-63384-7. URL http://dx.doi.org/10.1007/BFb0052259.

BONEH, D., DEMILLO, R. A. e LIPTON, R. J. **On the importance of checking cryptographic protocols for faults**. Em *Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'97, págs. 37–51, Berlin, Heidelberg, 1997. Springer-Verlag. ISBN 3-540-62975-0. URL http://portal.acm.org/citation.cfm?id=1754542.1754548.

BONEH, D., DEMILLO, R. A. e LIPTON, R. J. **On the Importance of Eliminating Errors in Cryptographic Computations**. *J. Cryptology*, 14(2):101–119, 2001. URL http://dx.doi.org/10.1007/s001450010016.

BURNETT, L. D. ***Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography***. Tese de Doutorado, Queensland University of Technology, 2005. URL http://eprints.qut.edu.au/16023/.

CARTER, G., DAWSON, E. e NIELSEN, L. **Key Schedule Classification of the AES Candidates**, 1999. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7286.

CARVALHO, C. A. B. **O uso de técnicas de recuperação de informações em criptoanálise**. Dissertação de Mestrado, Instituto Militar de Engenharia, 2006.

CHEN, C.-N. e YEN, S.-M. **Differential fault analysis on AES key schedule and some countermeasures**. Em *Proceedings of the 8th Australasian conference on Information security and privacy*, ACISP'03, págs. 118–129, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-40515-1. URL http://portal.acm.org/citation.cfm?id=1760479.1760494.

CHEN, H. e FENG, D. **An effective evolutionary strategy for bijective S-boxes**. Em *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, págs. 2120 – 2123. IEEE Computer Society, 2004. URL http://dx.doi.org/10.1109/CEC.2004.1331158.

DAEMEN, J. e RIJMEN, V. **AES Proposal: Rijndael**, 1998.

DAEMEN, J. e RIJMEN, V. ***The Design of Rijndael***. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. ISBN 3540425802.

DENNING, D. E. ***Cryptography and Data Security***. Addison-Wesley, United States of America, 1^o edition edition, 1982. ISBN 0201101505.

DILEEP, A. D. e SEKHAR, C. C. **Identification of block ciphers using support vector machines**. *International Joint Conference on Neural Networks*, 2006.

DUNKELMAN, O. e KELLER, N. **The effects of the omission of last round's MixColumns on AES**. *Inf. Process. Lett.*, 110:304–308, April 2010. ISSN 0020-0190. URL http://dx.doi.org/10.1016/j.ipl.2010.02.007.

DUSART, P., LETOURNEUX, G. e VIVOLO, O. **Differential Fault Analysis on A.E.S**. *CoRR*, cs.CR/0301020, 2003. URL http://arxiv.org/abs/cs.CR/0301020.

DWORKIN, M. **Recommendation for Block Cipher Modes of Operation**. SP-800-38a, U.S. DoC/National Institute of Standards and Technology, U.S. DoC/National Institute of Standards and Technology, 2001. URL csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

FEISTEL, H. **Cryptography and computer privacy**. *Scientific American*, 228(5): 15–23, 1973.

FULLER, J. E. ***Analysis of affine equivalent boolean functions for cryptography***. Tese de Doutorado, Queensland University of Technology, 2003. URL http://eprints.qut.edu.au/15828/.

GIRAUD, C. **DFA on AES**. Em *Advanced Encryption Standard - AES, 4th International Conference, AES 2004*, volume 3373, págs. 27–41. Springer, 2004. URL http://dx.doi.org/10.1007/11506447_4.

JANADI, A. e ANAS TARAH, D. **AES immunity Enhancement against algebraic attacks by using dynamic S-Boxes**. Em *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, págs. 1 –6, april 2008.

JUNOD, P. ***Statistical Cryptanalysis of Block Ciphers***. Tese de Doutorado, École polytechnique fédérale de Lausanne, Institut de systèmes de communication, 2005.

KAM, J. B. e DAVIDA, G. I. **Structured Design of Substitution-Permutation Encryption Networks**. *IEEE Trans. Comput.*, 28:747–753, October 1979. ISSN 0018-9340. URL http://portal.acm.org/citation.cfm?id=1311935.1312261.

KAZLAUSKAS, K. e KAZLAUSKAS, J. **Key-Dependent S-Box Generation in AES Block Cipher System**. *Informatica*, 20:23–34, January 2009. ISSN 0868-4952. URL http://portal.acm.org/citation.cfm?id=1516702.1516704.

KAZMI, S. e IKRAM, N. **Random Walk Algorithm Based Design Technique for S-Box**. Em *International Journal of Cryptology Research - Volume 1 (1)*, volume 1, págs. 65– 72. MSCR, 2009. URL http://www.mscr.org.my/V1(1)/PP%2065-72.pdf.

KIM, K. **A Study on the Construction and Analysis of Substitution Boxes for Symmetric Cryptosystems**. Dissertação de Mestrado, Yokohama National University, 1990.

KNUDSEN, L. R. e MEIER, W. **Correlations in RC6 with a Reduced Number of Rounds**. Em *Proceedings of the 7th International Workshop on Fast Software Encryption*, FSE '00, págs. 94–108, London, UK, 2001. Springer-Verlag. ISBN 3-540-41728-1. URL http://dl.acm.org/citation.cfm?id=647935.741057.

KUDOU, H., ICHIRO NAKAYAMA, S., WATANABE, A., NAGASE, T. e YOSHIOKA, Y. Aes immunity enhancement against algebraic attacks by using dynamic s-boxes. *Availability, Reliability and Security, International Conference on*, 0:792–797, 2009.

LI, X., CHEN, J., LIU, W. e WAN, W. **An improved AES encryption algorithm**. Em *Wireless Mobile and Computing (CCWMC 2009), IET International Communication Conference on*, págs. 694 –698, dec. 2009.

LIU, J., WEI, B., CHENG, X. e WANG, X. **An AES S-Box to Increase Complexity and Cryptographic Analysis**. Em *Proceedings of the 19th International Conference on Advanced Information Networking and Applications - Volume 1*, AINA '05, págs. 724–728, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2249-1. URL http://dx.doi.org/10.1109/AINA.2005.84.

MAR, P. e LATT, K. **New Analysis Methods on Strict Avalanche Criterion of S-Boxes**. Em *World Academy of Science, Engineering and Technology*, volume 48, págs. 150– 155. WASET, 2008. URL http://www.waset.org/journals/waset/v48/v48-24.pdf.

MATSUI, M. **Linear cryptanalysis method for DES cipher**. Em *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, págs. 386–397, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc. ISBN 3-540-57600-2. URL http://dl.acm.org/citation.cfm?id=188307.188366.

MATSUI, M. e YAMAGISHI, A. **A new method for known plaintext attack of FEAL cipher**. Em *Proceedings of the 11th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'92, págs. 81–91, Berlin, Heidelberg, 1993. Springer-Verlag. ISBN 3-540-56413-6. URL http://portal.acm.org/citation.cfm?id=1754948.1754958.

MEIER, W. e STAFFELBACH, O. **Nonlinearity criteria for cryptographic functions**. Em *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, págs. 549–562, New York, NY, USA, 1990. Springer-Verlag New York, Inc. ISBN 3-540-53433-4. URL `http://portal.acm.org/citation.cfm?id=111563.111614`.

MENEZES, A. J., VANSTONE, S. A. e OORSCHOT, P. C. V. ***Handbook of Applied Cryptography***. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996. ISBN 0849385237.

NAGIREDDY. **A Pattern Recognition Approach to Block Cipher Identification**. Dissertação de Mestrado, Indian Institute of Technology Madras, 2008.

NIST. **Data Encryption Standard (DES)**. Fips pub 46-3, U.S. DoC/National Institute of Standards and Technology, U.S. DoC/National Institute of Standards and Technology, 1999. URL `csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`.

NIST. **ADVANCED ENCRYPTION STANDARD (AES)**. FIPS-197, U.S. DoC/National Institute of Standards and Technology, U.S. DoC/National Institute of Standards and Technology, 2001. URL `csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

OLIVEIRA, G. A. **A aplicação de algoritmos genéticos no reconhecimento de padrões criptográficos**. Dissertação de Mestrado, Instituto Militar de Engenharia, 2011.

PIEPRZYK, J. e FINKELSTEIN, G. **Towards effective nonlinear cryptosystem design**. *Computers and Digital Techniques, IEE Proceedings E*, 135(6):325–335, 1988.

ROTHAUS, O. S. **On "Bent"Functions**. *J. Comb. Theory, Ser. A*, 20(3):300–305, 1976. URL `http://dx.doi.org/10.1016/0097-3165(76)90024-8`.

RUKHIN, A., SOTO, J., NECHVATAL, J., SMID, M., BARKER, E., LEIGH, S., LEVENSON, M., VANGEL, M., BANKS, D., HECKERT, A., DRAY, J. e VO, S. **A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications**. NIST special publication 800-22, National Institute of Standards and Technology (NIST), National Institute of Standards and Technology (NIST), 2001.

RUKHIN, A., SOTO, J., NECHVATAL, J., BARKER, E., LEIGH, S., LEVENSON, M., BANKS, D., HECKERT, A., DRAY, J., VO, S., RUKHIN, A., SOTO, J., SMID, M., LEIGH, S., VANGEL, M., HECKERT, A., DRAY, J. e III, L. E. B. **A statistical test suite for random and pseudorandom number generators for cryptographic applications**, 2010.

SHANNON, C. **Communication Theory of Secrecy Systems**. *Bell System Technical Journal*, 28:656–715, 1949.

SOUZA, W. A. R. **Identificação de padrões em criptogramas usando técnicas de classificação de textos**. Dissertação de Mestrado, Instituto Militar de Engenharia, 2007.

SOUZA, W. A. R., XEXÉO, M. A. J. e OLIVEIRA, C. **Método de Agrupamento de Criptogramas em Função das Chaves de Cifrar**. *IV Workshop em Algoritmos e Aplicações de Mineração de Dados (SBBD/SBES)*, 2008.

TORRES, H. R., OLIVEIRA, A. G., XEXÉO, M. A. J., SOUZA, R. A. W. e LIDEN, R. **Identificação de chaves e algoritmos criptográficos utilizando Algoritmo Genético e Teoria dos Grafos**. *9th International Information and Telecommunication Technologies Symposium*, 2010.

TORRES, H. R., OLIVEIRA, A. G., XEXÉO, M. A. J., SOUZA, R. A. W. e LIDEN, R. **Detecção de padrões em criptogramas como suporte às atividades de criptoanálise**. *IME -Série Informática UERJ*, 29, 2011. artigo aceito para publicação.

TRAN, M. T., BUI, D. K. e DUONG, A. D. **Gray S-Box for Advanced Encryption Standard**. Em *Proceedings of the 2008 International Conference on Computational Intelligence and Security - Volume 01*, CIS '08, págs. 253–258, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3508-1. URL `http://dx.doi.org/10.1109/CIS.2008.205`.

UEDA, T. e TERADA, R. **Uma Versão Mais Forte do Algoritmo RC6 contra a criptoanálise**. *VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, 2007.

WEBSTER, A. F. e TAVARES, S. E. **On the design of S-boxes**. Em *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, págs. 523–534, New York, NY, USA, 1986. Springer-Verlag New York, Inc. ISBN 0-387-16463-4. URL `http://portal.acm.org/citation.cfm?id=18262.25423`.

WHEELER, D. J. e NEEDHAM, R. M. **TEA, a Tiny Encryption Algorithm**. Em *FSE*, volume 1008 of *Lecture Notes in Computer Science*, págs. 363–366. Springer, 1994. URL `http://dx.doi.org/10.1007/3-540-60590-8_29`.

ZAÏBI, G., KACHOURI, A., PEYRARD, F. e FOURNIER-PRUNARET, D. **On dynamic chaotic S-BOX**. Em *Proceedings of the Second international conference on Global Information Infrastructure Symposium*, GIIS'09, págs. 51–55, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-4623-0. URL `http://portal.acm.org/citation.cfm?id=1719570.1719578`.