UNIVERSIDADE FEDERAL DE GOIÁS – UFG

CAMPUS CATALÃO – CaC

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO – DCC

Bacharelado em Ciência da Computação

Projeto Final de Curso

Emprego de testes estatísticos em geradores de números aleatórios para verificação de eficácia de algoritmos criptográficos

Autor: Amanda Cristina Davi Resende

Orientador: Vaston Gonçalves da Costa

Catalão - 2012

Amanda Cristina Davi Resende

Emprego de testes estatísticos em geradores de números aleatórios para verificação de eficácia de algoritmos criptográficos

Monografia apresentada ao Curso de
Bacharelado em Ciência da Computação da
Universidade Federal de Goiás – Campus Catalão
como requisito parcial para obtenção do título de
Bacharel em Ciência da Computação

Área de Concentração: Matemática Computacional
Orientador: Vaston Gonçalves da Costa

Catalão - 2012

Resende, Amanda Cristina Davi

Emprego de testes estatísticos em geradores de números aleatórios para verificação de eficácia de algoritmos criptográficos/**Amanda Cristina Davi Resende- Catalão - 2012**

Número de paginas: 67

Projeto Final de Curso (Bacharelado) Universidade Federal de Goiás, Campus Catalão, Curso de Bacharelado em Ciência da Computação, 2012.

Palavras-Chave: 1. Criptografia. 2. Gerador de Números Aleatórios. 3. Testes Estatísticos

Amanda Cristina Davi Resende

Emprego de testes estatísticos em geradores de números aleatórios para verificação de eficácia de algoritmos criptográficos

Monografia apresentada e aprovada em _____ de _____
Pela Banca Examinadora constituída pelos professores.

_____

Vaston Gonçalves da Costa – Presidente da Banca

_____

Thiago Jabur Bittar

_____

Donald Mark Santee

Dedico essa monografia em especial ao meu pai José Pedro de Resende Filho e minha avó/mãe Eunice Maria de Jesus, pois sempre estiveram ao meu lado oferecendo-me todo o apoio necessário para que eu pudesse concretizar este trabalho, e também a todas as pessoas que de alguma forma contribuiram para a realização do mesmo.

AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer a Deus, por ter me dado força para superar todos os obstáculos que surgiram, e por ter colocado pessoas tão especiais no meu caminho.

Em segundo, a minha imensão gratidão a minha família, especialmente meu pai José Pedro de Resende Filho, minha vó/mãe Eunice Maria de Jesus, minha dindinha Silvia Maria Correia Lima e minha madrinha Silvânia Davi Dias de Oliveira, que sempre me incentivaram e confiaram em mim para que eu pudesse concretizar esse trabalho.

Ao meu orientador, Vaston Gonçalves da Costa por ter me apresentado o tema, assim como pelo tempo disponibilizado para me auxiliar na realização desse trabalho. Agradeço também os professores, desde os da alfabetização até os da graduação, que de uma forma ou outra, me proporcionaram vários conhecimentos e lições.

Aos meus amigos/amigas e colegas que conheci em Catalão, Adriano, Bety, Bruno, Carla, Cassiana, Fábio, Rafa e Vinícius, pelo apoio necessário ao longo desse tempo. A Karla e a Lorena, pelos 3 anos e meio que dividimos moradia. Ao meu amigo Humberto, por sempre ter me ajudado, seja nas máterias da faculdade ou na vida particular. Ao meu amigo Cleriston, que me ajudou em todos os momentos necessários, principalmente para que este trabalho fosse concluído e a minha amiga Ariane, que sempre me deu a força e o incetivo necessário para concretizar esse trabalho.

Aos meus amigos de Monte Carmelo, Daniel, Isadora, Jack, João Paulo, Nayara, Paulo, Taty e Warley, que mesmo com a distância, estivaram presente à medida do possível.

E a todos, que de alguma forma, contribuíram para que esse fosse concretizado.

RESUMO

Em aplicações criptográficas tem-se a necessidade de utilizar tanto números aleatórios como pseudoaleatórios para a criação de sequências, utilizadas como chaves, que fazem, juntamente com o algoritmo, a criptografia da informação. A produção dessas sequências pode ser feita por Geradores de Número Aleatórios ou por Geradores de Números PseudoAleatórios. Os testes estatísticos, em geradores de números aleatórios, são úteis, como um passo inicial para determinar se um gerador é ou não, adequado para um aplicativo de criptografia específico. Nesse sentido, é apresentada neste trabalho, a construção de um protótipo de sistema baseado em testes estatísticos de aleatoriedade. Para tanto, serão apresentados os fundamentos estatísticos que justificam tal abordagem e os principais testes conhecidos, além das técnicas de criptoanálise empregadas em sistemas criptográficos baseados em blocos que não se comportam como geradores de números aleatórios. Todo o trabalho segue as normas do *National Institute of Standards and Technology* (NIST) e utiliza a linguagem Java para o seu desenvolvimento. O protótipo gerado, a partir de um texto cifrado, retorna informações que permitirão o criptógrafo corrigir falhas em sua cifra.

Palavras-Chaves: Criptografia, Gerador de Números Aleatórios, Testes Estatísticos

ABSTRACT

In cryptographic applications there is the need to use both random and pseudo random numbers to create sequences, used as keys, which, together with the algorithm, make the information encryption. The production of these sequences can be made by Random Numbers Generators or Generators of Pseudo-Random Numbers. Statistical tests of random numbers generators are useful as an initial step to determinate if a generator is suitable for an specific cryptographyc application. In this sense, it is presented in this work the construction of a prototype system based on statistical tests of randomness. Thus it will be presented the statistical foundations that justify such an approach and the main tests known, and also the cryptoanalysis techiniques used in cryptographic systems based on blocks that do not behave like random number generators. This work follows the guidelines of the National Institute of Standards and Technology and uses the Java language to its development. The prototype generated from a ciphertext, returns information that will allow the cryptographer correct flaws in your chiper.

Keywords: Cryptography, Random Numbers Generators, Statistics Test

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AES	*Advanced Encryption Standard*
DES	*Data Encryption Standard*
DH	Diffie-Hellman
ECH	*Elliptic Curve* Diffie-Hellman
FEAL	*Fast Data Encipherment Algorithm*
ID	Identificador
IDEA	*International Data Encryption Algorithm*
NIST	*National Institute of Standards and Technology*
PRNG	*Pseudo-Random Numbers Generator*
RNG	*Random Number Generators*

Capítulo 1

Introdução

Quando se fala em informação, de antemão se associa, à preocupação de salvaguardá-la de pessoas não autorizadas. Atualmente, esta preocupação conta com o apoio de tecnologias que podem ser utilizadas para garantir sua disponibilidade, integridade e autenticidade.

Uma vez que, a tecnologia usada para assegurar a informação também pode ser utilizada para dela se apropriar de maneira ilícita, é necessário que pesquisadores e profissionais em segurança da informação realizem buscas constantes por mecanismos que sirvam de contramedida aos ataques conhecidos.

A ferramenta mais utilizada nos dias atuais, baseada em tecnologias de *software* e *hardware*, é a criptografia. Se antes a criptografia utilizava de dispositivos mecânicos e palpáveis, hoje, com o uso disseminado das redes de computadores, a criptografia funda suas bases no desenvolvimento e aprimoramento de algoritmos que conseguem cifrar a informação de forma eficiente e viável computacionalmente [Stallings, 2008].

De forma simplificada, o objetivo da criptografia é transformar um texto original (ou informação qualquer), chamado de texto claro, em um texto cifrado utilizando uma chave. Tal transformação, também deve prever a recuperação da informação original, isto é, de posse do texto cifrado deve ser possível obter o texto claro utilizando também uma chave.

Durante o processo de construção de algoritmos criptográficos, o desenvolvedor precisa garantir que tal algoritmo não sucumbirá a ataques que analisam a distribuição dos bits do texto cifrado. De fato, tais recomendações, vide [Rukhin et al., 2010], fazem parte do que é considerado o básico em construção de algoritmos criptográficos. Em suma, um algoritmo criptográfico deve se comportar como um gerador de números aleatórios.

Desde 1997, o *National Institute of Standard and Techology (NIST)* recomenda uma bateria de testes estatísticos que deve ser utilizada por desenvolvedores de sistemas criptográficos para verificar, em primeira instância, se o sistema em desenvolvimento se porta como um gerador de números aleatórios.

Há ainda, outras aplicações, relacionadas a criptografia, que fazem uso de geradores

de números aleatórios, podendo citar geradores de chaves. De fato, uma vez que as fontes verdadeiramente aleatórias se tornam muitas vezes inviáveis, se admite (e se faz) o uso de geradores de números pseudoaleatórios, os quais utilizam uma fonte não aleatória para gerar os números que tem características de números aleatórios.

O objetivo, portanto, deste trabalho, é apresentar em detalhes, por meio de um revisão bibliográfica, geradores de números aleatórios e pseudoaleatórios utilizados em criptografia, bem como desenvolver um protótipo que poderá ser empregado por desenvolvedores de sistemas criptográficos para atestar o comportamento aleatório de seus sistemas. Para tal, é descrito no Capítulo 2 um breve histórico sobre criptografia, bem como os conceitos necessários para a compreensão do trabalho, como criptoanálise, geradores de números aleatórios e pseudoaleatórios, estatística p-valor e testes estatísticos. No Capítulo 3, são descritos em detalhes os testes utilizados na implementação do protótipo aqui proposto. No Capítulo 4, é mostrada a interface, como a inserção de dados é feita e a visualização dos resultados no protótipo. No Capítulo 5, são apresentados os resultados obtidos utilizando o protótipo. E por fim, no Capítulo 6, são apresentadas as conclusões obtidas através deste estudo, bem como possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo, serão abordados conceitos necessários para a compreensão do trabalho proposto, a começar por discorrer brevemente a história da criptografia, algumas definições e tipos de criptografia. Também, será introduzido o conceito de criptoanálise e geradores de números aleatórios e pseudoalatórios, além de, alguns conceitos matemáticos como estatistíca p-valor e testes estatísticos.

2.1 Criptografia

A palavra criptografia surgiu da união das palavras gregas "*kryptós*", que significa "oculto" e da palavra "*gráphein*", que significa "escrever" [do Nascimento, 2005], ou seja, o objetivo da criptografia é ocultar informações de pessoas não autorizadas.

A criptografia é tão antiga quanto a escrita, sendo utilizada por civilizações desde os tempos antes de Cristo. Uma prova disso, é que a criptografia já estava presente na escrita hieroglífica dos egípcios e em códigos secretos entre assírios, hebreus, hititas, persas e outros [Moreira, 2001]. Não se sabe exatamente qual foi a sua origem, tendo como seu primeiro relato, uma inscrição esculpida por volta do ano de 1900 a.C no túmulo de Khnomhotep II (servente real egípcio), que é descrita por Kahn na sua obra "The Codebreakers" [Kahn, 1996] referente aos monumentos que este construiu à serviço do faraó Amenemhet II, sendo o registro, visto na Figura 2.1, a substituição de alguns hieróglifos comuns por outros símbolos invulgares [Martins, 2005].

Figura 2.1: Inscrição esculpida por volta de 1900 a.C, onde aparece a substituição de alguns hieróglifos comuns por outros símbolos invulgares. Fonte: [Martins, 2005].

Em Esparta, por volta de 400 a.C., a técnica de escrever mensagens secretas envolvia enrolar, de forma espiral, uma tira de pergaminho ou papiro ao longo de um bastão cilíndrico [ZATTI e Beltrame, 2009]. Esta técnica é chamada de *Scytale* e é representada na Figura 2.2. Ela era utilizada pelos generais dos exércitos que desejavam comunicar com os éforos, os oficiais da antiga Esparta.

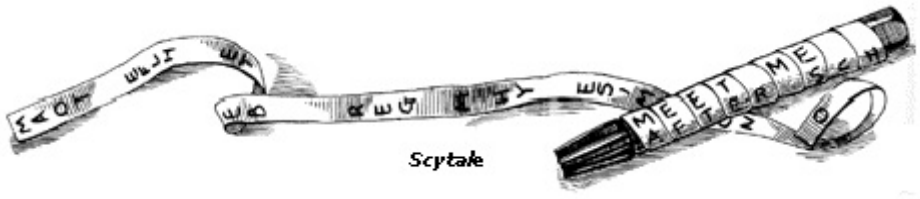

Figura 2.2: Bastão espartano utilizado para criptografia. Fonte: [Almeida e Appelt, 2003]

Na Roma antiga, por volta de 100 a.C., o Imperador Caio Júlio César utilizava um método para cifrar suas correspondências secretas que envia as seus generais, denominada Cifra de César (conhecida também como cifra de troca, código de César ou troca de César) no qual cada letra do texto era substituída pela terceira letra subseqüente do alfabeto, conforme Tabela 2.1.

Tabela 2.1: Substituição do código de César.

Original	**Cifrado**	**Original**	**Cifrado**
A	D	B	E
C	F	D	G
E	H	F	I
G	J	H	K
I	L	J	M
K	N	L	O
M	P	N	Q

O	R	P	S
Q	T	R	U
S	V	T	W
U	X	V	Y
W	Z	X	A
Y	B	Z	C

Nos tempos atuais, a maioria das informações são mantidas em meios digitais, sendo representadas por números binários, onde algoritmos criptográficos são utilizados para fazer o seu embaralhamento.

Para uma melhor compreensão do restante do trabalho, segue uma descrição de termos empregados no decorrer do texto.

- Texto claro (do inglês *plaintext*): É o texto (mensagem) original, inteligível.
- Texto cifrado (do inglês *ciphertext*): É o texto resultante do processo de criptografia aplicada por algum sistema criptográfico.
- Cifragem: Processo de converter o texto claro em um texto cifrado.
- Decifragem: Processo reverso da cifragem, onde restaura-se o texto claro a partir do cifrado.
- Algoritmo Criptográfico: Algoritmo que transforma uma mensagem legível por qualquer pessoa (texto claro) em uma mensagem criptografada.
- Chave: Seqüência de caracteres, que pode conter letras, dígitos e símbolos, que funciona junto com o algoritmo criptográfico para produzir um determinado texto cifrado.

2.2 Tipos de Criptografia

Segundo [de Souza, 2011], pode-se classificar criptografia em tipos, a simétrica e a assimétrica. A criptografia é simétrica (de chave secreta ou convencional) se a chave usada para cifrar é igual à chave usada para decifrar ou se uma chave é facilmente deduzida a partir da outra, sendo seu processo representado pela Figura 2.3.

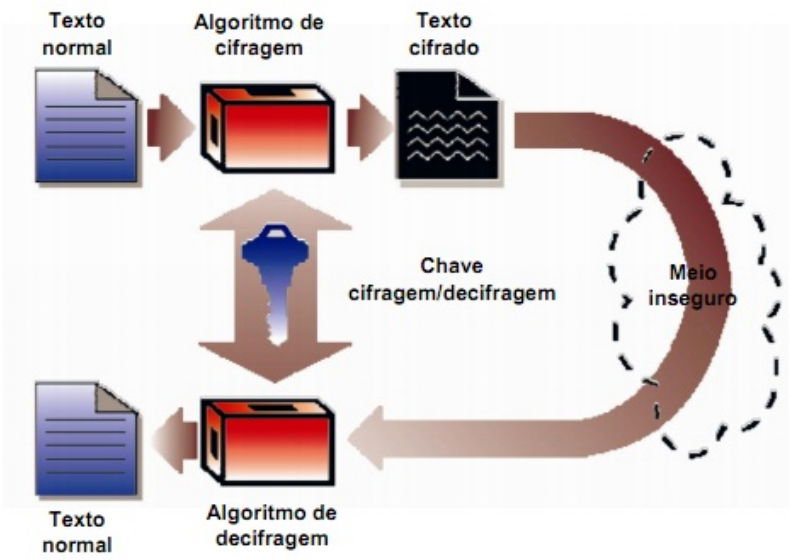

Figura 2.3: Processo de criptografia por chave secreta. Fonte: [Mendes, 2007].

A criptografia simétrica pode ser dividida em duas categorias: as cifras de substituição e as cifras de transposição [Cavalcante, 2005]. Na primeira, as letras do texto claro são substituídas por outras letras, símbolos ou números e, na segunda, uma permutação dos caracteres da mensagem é feita de acordo com algum critério.

Já a criptografia é dita assimétrica ou de chave pública quando a chave usada para cifrar é diferente da utilizada para decifrar e não se pode deduzir, em tempo computacional viável, uma chave a partir da outra, sendo o seu processo representado pela Figura 2.4.

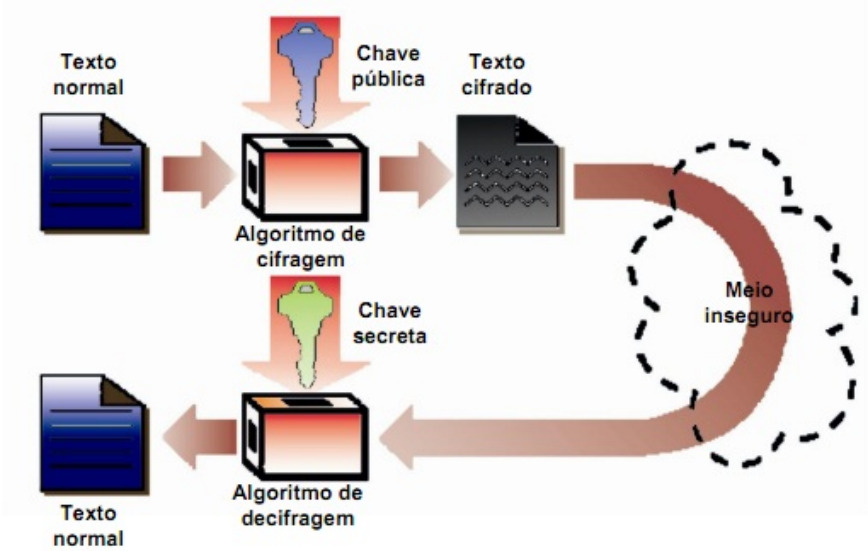

Figura 2.4: Processo de criptografia por chave pública. Fonte: [Mendes, 2007].

Alguns exemplos de cifras simétrica e assimétrica podem ser vistos na Tabela 2.2.

Tabela 2.2: Cifras Simétricas e Assimétricas.

Cifras Simétricas	**Cifras Assimétricas**
Data Encryption Standard (DES)	Diffie-Hellman (DH)
Triple DES	ElGamal
Advanced Encryption Standard (AES)	RSA
International Data Encryption Algorithm (IDEA)	*Elliptic Curve* Diffie-Hellman (ECDH)

2.3 Criptoanálise

A criptoanálise é a ciência que estuda as diversas técnicas utilizadas para se chegar à mensagem original (texto claro), sem o conhecimento da chave, a partir do texto cifrado. Ela também é utilizada para analisar cifras durante o período de seu desenvolvimento, de forma a torná-las eficientes, com chaves difíceis de serem quebradas. Duas técnicas mais promissoras e poderosas segundo [Stallings, 2008] é a criptoanálise diferencial e criptoanálise linear.

2.3.1 Criptoanálise Diferencial

A criptoanálise diferencial começou a ser relatada na literatura aberta por volta de 1990, sendo o primeiro trabalho publicado por Murphy [Murphy, 1990] em que é utilizada a criptoanálise em uma cifra de bloco[1] chamada *Fast Data Encipherment Algorithm* (FEAL). Depois, diversos trabalhos foram realizados, principalmente por Bihan e Shamir, que demostraram a utilização dessa técnica em vários tipos de algoritmos, tendo seus trabalhos resumidos em [Biham e Shamir, 1993].

O ataque de criptoanálise diferencial é complexo, tendo sua descrição completa em [Biham e Shamir, 1993]. Segundo [Stallings, 2008], o raciocínio é observar o comportamento de pares de blocos de textos evoluindo a cada rodada da cifra, em vez de observar a evolução de um único bloco de texto.

2.3.2 Criptoanálise Linear

A criptoanálise linear, que é descrita em [Matsui, 1994], é um ataque com textos claros conhecidos que usa aproximações lineares para descrever as transformações realizadas no

[1] O texto claro é dividido em blocos, com um tamanho fixo, cifrando cada um destes blocos separadamente

DES, requerendo 2^{43}(aproximadamente nove trilhões) pares de textos claros conhecidos para ser bem sucedido [de Souza, 2011].

Além da criptoanálise diferencial e linear, existem outras técnicas de criptoanálise, como a de força bruta que consiste em testar todas as possibilidades de chave para verificar qual está sendo utilizada [Mendes et al., 2011], a *ciphertext only*, que tenta chegar à chave ou ao texto original conhecendo-se apenas uma parte da mensagem codificada, a *known text*, que tenta-se chegar à chave conhecendo-se uma pequena parte do texto original e do cifrado correspondente, dentre outras [Paulo P. Flor et al., 2007].

2.4 Geradores de Números Aleatórios e Pseudoaleatórios

Em aplicações criptográficas tem-se a necessidade de utilizar tanto números aleatórios como pseudoaleatórios para a criação de sequências, utilizadas como chaves, que fazem juntamente com o algoritmo, a criptografia da informação [Resende e da Costa, 2011]. A produção dessas sequências pode ser feita por dois tipos básicos de geradores: Geradores de Números Aleatórios e os Geradores de Números Pseudoaleatórios.

2.4.1 Geradores de Números Aleatórios

Geradores de Números Aleatórios (RNG de *Random Number Generators*) utilizam uma fonte não-determinística (a fonte de entropia[2]), juntamente com algumas funções de processamento (o processo de destilação da entropia) para produzir aleatoriedade [Rukhin et al., 2010] e [Resende e da Costa, 2011]. As saídas desse tipo de gerador podem ser usadas diretamente como um número aleatório, nesse caso a saída precisa satisfazer critérios rigorosos de aleatoriedade, ou pode servir como entrada para geradores de números pseudoaleatórios [Schneier e Sutherland, 1995].

A melhor maneira de se obter números verdadeiramente aleatórios, é utilizar como fonte medidas de fenômenos físicos, tais como decaimento radioativo, ruído termal em semicondutores, amostras de som retiradas de um ambiente barulhento, dentre outros [Gutmann, 1998].

Na literatura é possível encontrar alguns artigos que retratam em detalhes a construção de RNGs usando fontes de entropia apropriadas. Dentre estes, pode-se citar o artigo de [Fairfield et al., 1985], que mostra como se gera um fluxo de bits aleatórios baseado na instabilidade da frequência de um oscilador.

Entretanto, é evidente que a construção de tais geradores, requer um embasamento

[2]Grandeza termodinâmica geralmente associada ao grau de desordem.

teórico e prático que foge do escopo proposto neste trabalho. O que deve ficar claro com relação a tais geradores, é que poucos computadores (ou usuários) tem acesso a *hardwares* especializados necessários para analisar as fontes aleatórias e, portanto, os mesmos precisam utilizar de outros métodos para obter dados aleatórios [Gutmann, 1998].

Existem abordagens que não precisam de *hardwares* especiais, tais como, as que medem o tempo de turbulência de ar no movimento das cabeças de leitura do disco rígido [Davis et al., 1994] assim como, as que medem o tempo usado para pressionar as teclas ao se inserir a senha de um usuário [Plumb, 1994] e [Zimmermann, 1995]. Tais abordagens mostram-se inseguras, se forem utilizadas separadamente, contudo, sua utilização se faz presente em associações criptográficas envolvendo Geradores de Números Pseudoaleatórios (PRNG de *Pseudo-Random Numbers Genarator*), conforme seção 2.4.2.

Infelizmente, os conselhos sobre segurança da literatura são muitas vezes ignorados, o que acaba por resultar na produção de geradores de números aleatórios inseguros, os quais, por sua vez, produzem senhas criptográficas que são mais fáceis de serem atacadas do que o sistema criptográfico utilizado pela senha. Uma fonte popular de números aleatórios ruins é a que se baseia no tempo atual e no identificador (ID) de processos. Esse tipo de gerador ficou notoriamente conhecido no final de 1995, quando se quebrou a encriptação do navegador *web* Netscape® consumindo aproximadamente 1 minuto. Devido a um espaço limitado de valores fornecidos por sua fonte é que se pode realizar a quebra do Netscape, tal fato causou tanta comoção que ganhou fama na impresa mundial [Sandberg, 1995].

2.4.2 Geradores de Números Pseudoaleatórios

Geradores de Números Pseudoaleatórios (PRNG's) usam uma ou mais entradas e geram múltiplos números "pseudo" aleatórios, sendo essas entradas chamadas de sementes [Resende e da Costa, 2011]. Em contextos em que a imprevisibilidade é necessária, a própria semente deve ser aleatória e imprevisível, assim, por padrão, um PRNG deve obter suas sementes a partir das saídas de um RNG [Rukhin et al., 2010].

As saídas de um PRNG normalmente são funções determinístas da semente, por isso se utiliza o termo "pseudo".

Os números pseudoaleatórios muitas vezes parecem ser mais aleatórios do que números aleatórios obtidos de fontes físicas, pois se uma pseudo-sequência está bem construída, cada valor na sequência é produzido a partir do valor anterior por meio de transformações que parecem introduzir uma aleatoriedade adicional [Rukhin et al., 2010].

Segundo [Vieira et al., 2004], os tipos de geradores de números aleatórios e pseudoaleatórios mais conhecidos e utilizados são:

- Geradores Congruentes Lineares [Press et al., 1992]: É um dos mais antigos e conhecidos algoritmos para a criação de números aleatórios, sendo o número gerado

pela fórmula:

$$x_{n+1} = (ax_n + c) \quad mod \quad m, \qquad n \geq 0 \tag{2.1}$$

onde, x_{n+1} é o número gerado, x_n é o número anterior, x_0 é a semente (que deve ser fornecida por um RNG) e a, c e m são constantes[3].

- Geradores de Atraso de Fibonacci [Marsaglia, 1985]: Tem-se esse nome por causa similaridade com a sequência de Fiboncacci, sendo o número gerado pela fórmula:

$$x_n = x_{n-l} \quad op \quad x_{n-k}, \qquad 0 < k < l \tag{2.2}$$

onde, x_n é o número gerado e op pode ser uma adição, subtração, multiplicação módulo m ou a operação "ou exclusivo"[4].

- Geradores de Registradores de Deslocamento [Marsaglia, 1985]: Consideram os bits de uma palavra de computador como os elementos de um vetor binário, na qual, iterativamente, através de transformações lineares[5], geram sequências de vetores binários interpretadas como sequências de números inteiros aleatórios uniformes.

- Geradores Híbridos [Marsaglia, 1985]: É a combinação de dois ou mais geradores com o objetivo de obter um gerador melhor.

2.4.3 Considerações sobre RNG e PRNG

Tanto os geradores aleatórios quanto os pseudoaleatório, produzem uma sequência de bits (0's e 1's) que podem ser divididas em subfluxos ou blocos. Na divisão em blocos, a sequência de bits é dividida em M-blocos de tamanho n, onde M é a quantidade de blocos e n é a quantidade de bits dentro do bloco. Já em subfluxo, a sequência é dividida bit a bit, ou seja, são M-blocos de tamanho 1, onde M é o comprimento da cadeia de bits.

2.5 Estatística P-Valor

P-valor, na estatística clássica, é uma estatística utilizada para sintetizar o resultado de um teste de hipótese, sendo definido como a probabilidade de se obter uma estatística

[3]A operação mod encontra o resto da divisão de um número por outro. Dados dois números a (o dividendo) e b (o divisor), a mod b e o resto da divisão de a por b.

[4]Operação lógica entre dois operandos que resulta em um valor lógico verdadeiro (1) se e somente se apenas um dos operandos tem um valor verdadeiro (1).

[5]Tipo particular de função entre dois espaços vetoriais que preserva as operações de adição vetorial e multiplicação por escalar.

de teste igual ou mais extrema quanto àquela observada em uma amostra, assumindo verdadeira a hipótese nula. Para o teste de hipótese são definidas duas hipótese: a hipótese nula (H_0) e a hipótese alternativa (H_1 ou H_a), a primeira é a ausência do efeito que se quer verificar, com por exemplo, para verificar se uma sequência é não aleatória, a hipótese nula é considerar que a sequência é aleatória, já a segunda é a investigação do que se quer verificar, ou seja, para o exemplo a hipótese alternativa é considerar que a sequência é não aleatória.

A priori, H_0 é considerada verdadeira, sendo o objetivo provar o contrário, ou seja, que H_0 é falsa. Para a tomada de decisão, representada na tabela 2.3, utiliza-se uma amostra da população[6] e não a população inteira. Por esse motivo, dois erros podem acontecer: erro do tipo I (Falso Positivo) e erro do tipo II (Falso Negativo).

Tabela 2.3: Decisões possíveis. Fonte: [Rukhin et al., 2010].

Situação	Conclusão	
	Aceita H_0	Rejeita H_0
H_0 é verdadeiro	Sem erro	Erro tipo I
H_1 é verdadeiro	Erro tipo II	Sem erro

O primeiro erro é rejeitar H_0 quando, na verdade, ela é verdadeira, sendo a probabilidade de cometer este erro designada por α (nível de significância). O α é um valor fixo definido pelo pesquisador antes mesmo da coleta de dados, sendo geralmente definido no intervalo [0,001 , 0,01]. Já o segundo erro é aceitar H_0 quando, na verdade ela é falsa, tendo a probabilidade de cometer este erro designada por β. Diferentemente do valor de α, não é possível determinar o valor de β, pois assumindo-se que a hipótese nula é falsa, o parâmetro que se pretende testar poder assumir infinitos valores, possibilitando assim valores diferentes de β.

Quando o valor de α é 0,001 indica que espera-se em 1 amostra de cada 1000 H_0 seja rejeitado erroneamente. Para um valor $P\_valor < 0,001$, significaria que a hipótese alternativa seria considerada verdadeira com uma confiança de 99,9% [Rukhin et al., 2010].

O valor de α em 0,01 indica que espera-se em 1 amostra de cada 100 H_0 seja rejeitado erroneamente. O $P\_valor < 0,01$ significaria que a hipótese alternativa seria considerada verdadeira com uma confiança de 99,9% [Rukhin et al., 2010].

[6]Conjunto de todos os elementos ou resultados sob investigação.

2.6 Testes Estatísticos

Para verificar se um RNG ou um PRNG estão produzindo sequências verdadeiramente aletórias, pode-se empregar dois tipos de testes: os testes teóricos e os testes empíricos (estatísticos) [L'Ecuyer, 1992].

O primeiro, é específico para cada tipo de gerador e o analisa com o propósito de extrair propriedades do comportamento da sequência de pontos ao longo do período. Observa-se que geradores com boas propiedades teóricas tem uma boa performace nos testes empíricos [Vieira e Ribeiro, 2004].

Já o segundo tipo de teste aparece como um complemento do primeiro, onde o objetivo é avaliar (por meio de técnicas estatísticas) o quão boa é uma sequência produzida pelo gerador e consequentemente, encontrar evidências contra a hipótese nula (vide seção 2.5). Este foi o tipo de teste utilizado para o desenvolvimento deste trabalho.

Vários testes estatísticos podem ser aplicados em uma sequência para avaliar se ela é realmente aleatória, cada qual tentando detectar a presença ou ausência de algum tipo de padrão. Pelo fato da diversidade de testes, nenhum conjunto de teste pode ser considerado suficientemente completo [Rukhin et al., 2010]. Na tabela 2.4, Soto [Soto, 1999] apresenta 4 baterias de testes.

Tabela 2.4: Baterias de testes estatísticos. Fonte: [Soto, 1999].

Origem/Afiliação	**Teste Estatístico**
Donald Knuth/Stanford University	The Art Of Computer Programming Vo.2 Seminumerical Algorithmos
George Marsaglia/Florida State University	DIEHARD
Alfred Menezes et. al./CRC Press, Inc.	Handbook of Applied Cryptography
Andrew Rukhin et. al./NIST ITL	NIST Statistical Test Suite

A explicação detalhada das baterias de testes podem ser encontradas em [Knuth, 1981], [Marsaglia, 1996], [Menezes et al., 1997] e [Rukhin et al., 2010] respectivamente.

A bateria de teste proposta pelo NIST é a que será abordada no desevolvimento deste trabalho sendo melhor explicada na seção 2.6.1.

2.6.1 Bateria de teste do NIST

Testes estatísticos podem ser aplicados a uma sequência de bits para analisar se ela se comporta de maneira aleatória. Tais testes podem ser utilizados em geradores de números aleatórios como um passo inicial para determinar se um gerador é, ou não, adequado para

um aplicativo de criptografia específico. Contudo, a aprovação nesses testes não garante a eficácia do sistema criptográfico à criptoanálise.

O NIST desenvolveu um pacote com 15 testes estatísticos [Rukhin et al., 2010], para testar a existência ou não de aleatoriedade em uma sequência binária produzida por *hardware* e *software* de criptografia baseada em RNG's ou PRNG's.

Os 15 testes são descritos resumidamente a seguir conforme [Rukhin et al., 2010], [Soto, 2000] e [Resende e da Costa, 2011].

Frequência (Monobit)

O objetivo desse teste, é analisar a proporção de 0's e 1's em toda a amostra da sequência de bits, a fim de verificar se a ocorrência de tais valores se dá como seria esperado para uma sequência verdadeiramente aleatória, ou seja, o número de 0's e 1's na sequência seria aproximadamente igual.

Se a amostra de bits for reprovada nesse teste, ela será considerada não aleatória e não precisará ser submetida aos demais.

Frequência dentro de Blocos

Para a realização desse, deve-se dividir a amostra em blocos de M bits, onde M é a quantidade de bits dentro de cada bloco, então, é analisada a proporção de dígitos 1's em cada bloco de tamanho M.

O propósito do teste é determinar se a frequência de 1's em cada um dos blocos é aproximadamente $\frac{M}{2}$, como seria esperado caso houvesse aleatoriedade. Para blocos de tamanho $M = 1$, esse teste degenera-se para o Teste de Frequência (Monobit).

Corridas

Esse teste analisa o número total de corridas na amostra de bits, onde chama-se de corrida uma sequência ininterrupta de bits idênticos. Uma corrida de comprimento K consiste de exatamente K bits idênticos e limitados antes e depois por bits de valores opostos.

O objetivo do teste é determinar se o número de corridas de 1's e 0's de vários comprimentos comporta-se como esperado para uma sequência aleatória. Em suma, esse teste determina se a oscilação entre 0's e 1's é muito rápida ou muito lenta.

Mais longa corrida de 1's em um Bloco

Esse teste analisa as mais longas corridas de 1's dentro de blocos de N bits, para isso a amostra deve ser dividida em M blocos, como o teste de frequência dentro de blocos. Neste teste os blocos são divididos de acordo com o tamanho da amostra de bits.

Deseja-se então determinar se o comprimento da mais longa corrida de 1's dentro da sequência testada é consistente com o comprimento da mais longa corrida de 1's que seria esperada em uma sequência aleatória. Nota-se que uma irregularidade no comprimento esperado da mais longa corrida de 1's implica que também haverá irregularidade no comprimento esperado da mais longa corrida de 0's, assim sendo, é necessária somente a verificação do comprimento da corrida de 1's.

Posto para matrizes binárias

O objetivo desse teste é analisar o posto de sub-matrizes disjuntas de uma sequência de bits a fim de verificar se há dependência linear entre sub-strings de comprimentos pré-fixados, sub-strings estas oriundas da sequência original.

Espectral para transformação discreta de Fourier

Este teste analisa as alturas dos picos na Transformação Discreta de Fourier em uma sequência.

O propósito do teste é detectar características periódicas (isto é, padrões repetitivos próximos uns dos outros) na sequência testada, o que indicaria um desvio da assumpção de aleatoriedade. A intenção é detectar se o número de picos excedendo o padrão de 95% é significativamente diferente ao nível de 5%.

Não-sobreposição de padrão

O objetivo deste teste é o número de ocorrências de *strings* alvo pré-especificadas para detectar geradores que produzem muitas ocorrências de um dado padrão não-periódico.

Para esse e para o "Teste de Sobreposição de Padrão", uma janela de m-bits será usada para busca de um padrão específico, com o mesmo tamanho da janela. Se o padrão não for encontrado, a janela desliza (é transladada) da posição de um bit. Se o padrão é encontrado a janela é transladada para o bit após o padrão encontrado, e a busca continua.

Sobreposição de padrão

Este teste também trabalha com as ocorrências de *strings* alvo pré-especificadas e com uma janela de m-bits para buscar por um padrão específico de m-bits, assim como o teste de não-sobreposição de padrão. Se o padrão não for encontrado, a janela é transladada de um bit de sua posição.

A diferença entre esse e o de não-sobreposição de padrão é que nesse, quando um padrão é encontrado, a janela é transladada somente um bit e a busca continua.

Estatística universal de Maurer

O objetivo deste teste é encontrar o número de bits entre padrões (uma medida que está relacionada ao comprimento de uma sequência comprimida), detectando se a sequência pode ou não ser significativamente comprimida sem perda de informação. Uma sequência significativamente comprimível é considerada ser não aleatória.

Complexidade linear

O foco deste teste é o comprimento de um retorno linear de registro shiftado (LFSR).

O propósito do teste é determinar se uma sequência é complexa o suficiente para ser considerada aleatória. Uma LFSR que é muito simples implica em não-aleatoriedade.

Serial

O foco deste teste é encontrar a frequência de todas as possíveis sobreposições de padrões de m-bits na sequência toda, determinando se o número de ocorrências dos $2m$ padrões de sobreposição de m-bits é aproximadamente o mesmo do esperado para uma sequência aleatória.

Sequências aleatórias tem uniformidade, ou seja, todo padrão de m-bits têm a mesma chance de aparecer como qualquer outro padrão de m-bits. Note que para $m = 1$, o Teste Serial é equivalente ao Teste de Frequência (Monobit).

Entropia aproximada

Como no Teste Serial, o foco desse teste é encontrar a frequência de todas as possíveis sobreposições de padrões de m-bits na sequência toda, comparando a frequência de sobreposições de dois blocos consecutivos/adjacentes de comprimentos m e $m + 1$ contra o resultado esperado para uma sequência aleatória.

Somas cumulativas

O foco deste teste é encontar a excursão máxima (de 0's) de caminho aleatório definido pela soma cumulativa de dígitos ajustados (bit 0 = -1, bit 1 = +1) na sequência. Em cada caminho soma-se os dois primeiros dígitos ajustados, depois adiciona-se mais um e soma-se os 3 primeiros e assim consecutivamente. O maior valor, sendo ele de 0's ou de 1's, encontrado em uma rodada é usada como valor para a sequência do teste.

O teste determina se a soma cumulativa de sequências parciais, ocorridas na sequência testada, é muito grande ou muito pequena relativamente ao comportamento esperado da soma cumulativa de uma sequência aleatória. Esta soma cumulativa pode ser conside-

rada como um caminho aleatório. Para uma sequência aleatória, a excursão do caminho aleatório seria próxima de 0.

Para certos tipos de sequências não-aleatórias, a excursão deste caminho aleatório de 0's será muito grande.

Excursões aleatórias

O foco desse teste é encontrar o número de ciclos tendo exatamente k visitas em uma soma cumulativa em um caminho aleatório. O caminho aleatório da soma cumulativa é derivado de somas parciais depois que a sequência (de zeros e uns) é transformada na sequência apropriada de -1 e + 1.

Um ciclo de um caminho aleatório consiste de uma sequência de passos de comprimento unitário, tomado ao acaso, começando e terminando na origem. O propósito deste teste é determinar se o número de visitas a um particular estado dentro de um ciclo desvia do que é esperado em uma sequência aleatória.

Este teste é realmente uma série de oito (8) testes (e conclusões), um teste e conclusão para cada um dos estados: - 4, -3, -2, -1 e 1, 2, 3, 4.

Variante de excursões aleatórias

O foco desse teste é encontar o número total de vezes que um estado particular é visitado (isto é, que ocorre) em uma soma cumulativa de um caminho aleatório, cujo propósito é detectar desvios do número esperado de visitas para vários estados em um caminho aleatório.

Este teste é realmente uma série de dezoito (18) testes (e conclusões), um teste e conclusão para cada um dos estados: - 9, -8, ..., -1 e + 1, +2, ..., +9.

Capítulo 3

Descrição dos Testes Utilizados

Neste capítulo serão apresentados em detalhes os 5 testes implementados no trabalho. Para tal, cada teste descrito aqui terá seu objetivo descrito, os parâmetros a serem utilizados, seu funcionamento, a conclusão e a interpretação dos resultados, bem como um exemplo.

3.1 Descrição dos testes utilizados

Nessa seção os 5 testes implementados no protótipo serão descritos detalhamente. A descrição é baseada em [Rukhin et al., 2010] e o valor de significância (α) foi fixado em 0,01.

3.1.1 Teste de frequência

O objetivo do teste é analisar a proporção de 0's e 1's em toda a amostra da sequência de bits, verificando se o número de 0's e 1's na sequência ocorrem como seria esperado para uma sequência verdadeiramente aleatória, ou seja, o número de 0's e 1's na sequência seriam aproximadamente iguais.

A hipótese nula (H_0) e a hipótese alternativa (H_1)

- H_0: A frequência de 1's é igual à de 0's;
- H_1: As frequências de 1's e 0's são diferentes.

Parâmetros

E = cadeia de bits;
n = tamanho da cadeia de bits.

Descrição do teste

A descrição do teste é dividida nos seguintes passos:

(1) Conversão em ± 1: Os 0's e 1's da sequência de entrada (E) são convertidos respectivamente em valores de -1 e 1 e são somados para produzir $S_n = X_1 + X_2 + ... + X_n$, onde $X_i = 2E_i - 1$.

Por exemplo, se $E = 0010101001$, então $n = 10$ e $S_n = (-1) + (-1) + 1 + (-1) + 1 + (-1) + 1 + (-1) + (-1) + 1 = -2$.

(2) Calcular o valor de $S_{obs} = \frac{|S_n|}{\sqrt{n}}$.

Para o exemplo desta seção: $S_{obs} = \frac{|-2|}{\sqrt{10}} = 0{,}6324$.

(3) Calcular o $p - valor = erfc(\frac{S_{obs}}{\sqrt{2}})$, onde $erfc$ é a função erro complementar definida na Seção A.1 do Apêndice A.

Para o exemplo desta seção: $p - valor = erfc(\frac{S_{obs}}{\sqrt{2}}) = 0{,}5270$.

Conclusão e interpretação do resultado

Como o resultado do $p - valor$ obtido no passo 3 é $\geq 0{,}01$, a conclusão é que não há evidências, a um nível de significância de 0,01, de que as frequências sejam diferentes.

Exemplo

Entrada $(E) = 1010011010010100100100100010100100100100111000101001111010010010100110100101100101100011010101000 11;$
Entrada $(n) = 100;$
Processamento $(S_n) = S_{100} = 12;$
Processamento $(S_{obs}) = 1{,}2;$
Saída $(p - valor) = 0{,}2301;$
Conclusão = Como o $p - valor$ é $\geq 0{,}01$, a sequência é considerada aleatória.

3.1.2 Teste de frequência dentro de blocos

O objetivo deste teste é analisar a proporção de 1's dentro de cada um dos blocos formados por M bits, determinando se a frequência de 1's em cada um dos blocos de M

bits é aproximadamente $\frac{M}{2}$ como seria esperado caso ocorresse aleatoriedade. Para blocos de tamanho $M = 1$, este teste degenera-se para o Teste de Frequência (Monobit).

A hipótese nula (H_0) e a hipótese alternativa (H_1)

- H_0: As frequências de 0's e 1's dentro de todos os blocos são iguais;
- H_1: As frequências de 1's e 0's em ao menos um dos blocos é diferente.

Parâmetros

M = comprimento de cada bloco;
n = comprimento da cadeia de bits;
E = Cadeia de bits;
N = quantidade de blocos.

Descrição do teste

A descrição do teste é dividida nos seguintes passos:

(1) Partição da sequência de entrada em $N = \lfloor \frac{n}{M} \rfloor$. Rejeitar os bits sobressalentes.

Por exemplo, se $n = 10$. $M = 3$ e $E = 0110011011$, 3 blocos ($N = 3$) seriam criados, constituídos por 011, 001, 101. O 1 final é descartado.

(2) Determinar a proporção π_i de 1's em cada M-bloco usando a equação $\pi_i = \frac{\sum_{j=1}^{M} E_{(i-1)M+j}}{M}$, para $1 \leq i \leq N$.

Para o exemplo desta seção: $\pi_1 = \frac{2}{3}$, $\pi_2 = \frac{1}{3}$ e $\pi_2 = \frac{2}{3}$.

(3) Calcular o $\chi^2(obs) = 4M\sum_{i=1}^{N}(\pi_i - \frac{1}{2})^2$.

Para o exemplo desta seção: $\chi^2 = 4 \times 3 \times ((\frac{2}{3} - \frac{1}{2})^2 + (\frac{1}{3} - \frac{1}{2})^2 + (\frac{2}{3} - \frac{1}{2})^2) = 1$.

(4) Calcular o $p - valor = igmac(\frac{N}{2}, \frac{\chi^2(obs)}{2})$, onde $igmac$ é a função gama incompleta definada na Seção A.2 do Apêndice A.

Para o exemplo dessa seção: $p - valor = igmac(\frac{3}{2}, \frac{1}{2}) = 0{,}8012$.

Conclusão e interpretação do resultado

Como o resultado do p-valor obtido no passo 4 é $\geq$ 0,01, a conclusão é não há evidências, a um nível de significância 0,01, de que as frequências dentro dos blocos são diferentes.

Exemplo

Entrada (E) = 10100110100101001001001000101001001001001110001010
01110100100101010011010010110010110001101010100011;
Entrada (n) = 100;
Entrada (M) = 10;
Procesamento (N)= 10;
Processamento (χ^2) = 4;
Saída $(p - valor)$ = 0,9473;
Conclusão = Como o $p - valor$ é $\geq$ 0,01, a sequência é considerada aleatória.

3.1.3 Teste de Corrida

O objetivo deste teste é analisar o número total de corridas na amostra de bits, onde chama-se de corrida uma sequência ininterrupta de bits idênticos. Uma corrida de comprimento K consiste de exatamente K bits idênticos e limitados antes e depois por bits de valores opostos, determinando se o número de corridas de 1's e 0's de vários comprimentos comporta-se como esperado para uma sequência aleatória.

A hipótese nula (H_0) e a hipótese alternativa (H_1)

- H_0: A sequência é equilibrada e aleatória;
- H_1: A sequência é desequilibrada ou não é aleatória.

Parâmetros

E = Cadeia de bits;
n = tamanho da cadeia de bits.

Descrição do teste

A descrição do teste é dividida nos seguintes passos:

(1) Calcular um pré-teste da proporção de π na sequência de entrada: $\frac{\sum_j E_j}{n}$. Por exemplo se $E = 1010110010$, então n=10 e $\pi = \frac{5}{10} = \frac{1}{2}$.

(2) Para determinar se o pré-teste foi aprovado, basta realizar a seguinte verificação: se $|\pi - \frac{1}{2}| < \tau$, onde $\tau = \frac{2}{\sqrt{n}}$. Caso contrário o teste não precisa ser mais realizado, pois ele não foi aprovado no pré-requisito.

Para o exemplo desta seção: $\tau = \frac{2}{\sqrt{10}} = 0{,}63246$, então $|\pi - \frac{1}{2}| = |\frac{5}{10} - \frac{1}{2}| = 0 < \tau$.

(3) Calcular $\nu_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$, onde $r(k) = 0$ se $E_k = E_{k+1}$ e $r(k) = 1$ caso contrário.

Para o exemplo desta seção temos: $\nu_{10} = (1+1+1+1+0+1+0+1+1)+1 = 8$.

(4) Calcular o $p-valor = erfc(\frac{|\nu_n(obs)-2n\pi(1-\pi)|}{2\sqrt{2n\pi(1-\pi)}})$, onde $erfc$ é a função erro complementar descrita na Seção A.1 do Apêndice A.

Para o exemplo desta seção: $p - valor = erfc(\frac{|8-2\times10\times(\frac{5}{10}(1-\frac{5}{10}))|}{2\sqrt{2\times10}\times\frac{5}{10}(1-\frac{5}{10})}) = 0,05777$.

Conclusão e interpretação do resultado

Como o resultado do $p - valor$ obtido no passo 4 é $\geq$ 0,01, a conclusão é não há evidências, a um nível de significância de 0,01 de que a sequência não é aleatória.

Exemplo

Entrada (E) = 11001001000011111101101010100010001000010110100011
10001000100100110101001100011001100010100010111000;
Entrada (n) = 100;
Entrada (τ) = 0,2;
Processamento (π) = 0,43;
Processamento ($V_n(obs)$) = ($V_{100}(obs)$) = 54;
Saída ($p - valor$) = 0,3096;
Conclusão = Como o $p - valor$ é $\geq$ 0,01, a sequência é considerada aleatória.

3.1.4 Teste de não sobreposição de padrão

O objetivo do teste é encontrar o número de ocorrências de sequências pré-definidas, para detectar se os geradores produzem muitas ocorrências de um determinado padrão

aperiódico. Uma janela de m bits é utilizada para procurar um padrão de m-bits específico. Se o padrão não for encontrado, a janela desliza um bit de posição. Se o padrão for encontrado, a janela é redefinida para um bit depois do padrão encontrado e a busca recomeça. Os padrões aperiódicos para $2 \leq m \leq 5$ são definidos conforme a Tabela 3.1 e os demais padrões para $6 \leq m \leq 9$, pode ser vistos no Apêndice B.

Tabela 3.1: Padrões aperiódicos para $2 \leq m \leq 5$. Fonte: [Rukhin et al., 2010].

$m = \mathbf{2}$	$m = \mathbf{3}$	$m = \mathbf{4}$	$m = \mathbf{5}$
01	001	0001	00001
10	011	0011	00011
	100	0111	00101
	110	1000	01011
		1100	00111
		1110	01111
			11100
			11010
			10100
			11000
			10000
			11110

A hipótese nula (H_0) e a hipótese alternativa (H_1)

- H_0: Não existe o padrão aperiódico B em nenhum dos blocos.
- H_1: Existe o padrão aperiódico B em ao menos um dos blocos.

Parâmetro

E = cadeia de bits;
n = tamanho da cadeia de bits;
M = tamanho do bloco;
B = O modelo de m bits;
m = tamanho do modelo de bits;
N = números de blocos.

Descrição do teste

A descrição do teste é dividida nos seguintes passos:

(1) Partição da sequência (E) em N blocos independentes de comprimento M.

Por exemplo, se E = 10010110010100110101, então $n = 20$. Se $N = 3$ e $M = 6$, então os três blocos seriam 100101, 100101 e 001101. Os bits 0 e 1 finais são descartados.

(2) $W_j(j = 1, ..., N)$ é o número de vezes que o modelo (B) ocorre dentro do bloco J. A busca para encontrar correspondências se dá através da criação de uma janela de m bits na sequência, comparando os bits dentro dessa janela contra o modelo. Se não houver correspondência, a janela desliza um bit, por exemplo, se $m = 5$ e a janela atual contém os bits 5-9, então a próxima janela conterá os bits 6-10. Se houver uma correspondência, a janela desliza m bits, por exemplo, se a janela atual contém bits 5-9, então a próxima janela conterá bits 10-14.

Para o exemplo acima, se $m = 2$ e do modelo $B = 01$, então a análise é conforme a Tabela 3.2.

Tabela 3.2: Análise do exemplo. Fonte: [Rukhin et al., 2010].

Posição do bit	Bloco 1		Bloco 2		Bloco 3	
	Bits	W_1	Bits	W_2	Bits	W_3
1-2	10	0	10	0	00	0
2-3	00	0	00	0	01	Incrementa para 1
3-4	01	Incrementa para 1	01	Incrementa para 1	Não examinado	
4-5	Não examinado		Não examinado		10	1
5-6	01	Incrementa para 2	01	Incrementa para 2	01	Incrementa para 2

Assim, $W_1 = 2$, $W_2 = 2$ e $W_3 = 2$.

(3) Calcular a média teórica μ e a variância σ^2:

$$\mu = \frac{(M - m + 1)}{2^m} \tag{3.1}$$

$$\sigma^2 = M(\frac{1}{2^m} - \frac{2m-1}{2^{2m}}) \tag{3.2}$$

Para o exemplo desta seção: $\mu = \frac{(6-2+1)}{2^2} = 1{,}25$ e $\sigma = 6(\frac{1}{2^2} - \frac{2*2-1}{2^{2*2}}) = 0,375$.

(4) Calcular $\chi^2(obs) = \sum_{j=1}^{N} \frac{(W_j-\mu)^2}{\sigma^2}$

Para o exemplo desta seção: $\chi^2(obs) = \frac{(2-1,25)^2+(2-1,25)^2+(2-1,25)^2}{0,375} = 4,5$.

(5) Calcular o $p-valor = igmac(\frac{N}{2}, \frac{\chi^2(obs)}{2})$, onde $igmac$ é a função gama incompleta definada na Seção A.2 do Apêndice A. É importante ressaltar que para cada modelo, um novo $p-valor$ é calculado.

Para o exemplo desta seção $p-valor = igmac(\frac{2}{2}, \frac{4,5}{2}) = 0,1053$.

Conclusão e interpretação do resultado

Como o resultado do $p-valor$ obtido no passo 5 é $\geq$ 0,01, a conclusão não há evidências, a um nível de significância de 0,01, de que em algum bloco há o padrão aperiódico B.

Exemplo

Para um modelo $B = 001$, cujo tamanho é $m = 3$.

Entrada (E) = 10010110010100110101100101101100110101 1;
Entrada $(n) = 39$;
Entrada $(B) = 001$;
Processamento $(W_j) = W_1 = 1$, $W_2 = 1$, $W_3 = 1$ e $W_4 = 1$;
Processamento (μ)= 0.875;
Processamento $(\sigma^2) = 0.421875$;
Processamento $(\chi^2) = 0.1481$;
Saída $(p-valor) = 0.9973$;
Conclusão = Como o $p-valor$ é $\geq$ 0,01, a sequência é considerada aleatória.

3.1.5 Teste de sobreposição de padrão

O objetivo do teste é o número de ocorrências de sequências pré-definidas, para detectar se os geradores produzem muitas ocorrências de um determinado padrão periódico. Uma janela de m bits é utilizada para procurar um padrão específico de m-bits. Se o padrão não for encontrado, a janela desliza um bit de posição, o mesmo ocorre se o padrão for encontrado.

A hipótese nula (H_0) e a hipótese alternativa (H_1)

- H_0: Não existe o padrão periódico B em nenhum dos blocos;
- H_1: Existe o padrão periódico B em ao menos um dos blocos.

Parâmetro

E = cadeia de bits;
n = tamanho da cadeia de bits;
M = tamanho do bloco;
B = O modelo de m bits, que é formado apenas por 1's;
m = tamanho do modelo de bits;
N = números de blocos.

Descrição do teste

A descrição do teste é dividida nos seguintes passos:

(1) Partição da sequência (E) em N blocos independentes de comprimento M.

Por exemplo, se E = 10010011100110110010010011100100111001100010010011, então n = 50. Se N = 5 e M = 10, então os cincos blocos serão 1001001110, 0110110010, 0100111001, 0011100110 e 0010010011. Quando houver bits sobressalentes, estes serão descartados.

(2) Calcular o número de ocorrências de um modelo (B) em cada um dos N blocos. A busca por encontrar correspondências é realizada através da criação de uma janela de m bits na sequência, comparando os bits dentro dessa janela contra o modelo e incrementando um contador quando houver uma correspondência. A janela desliza sobre o bit após cada exame, por exemplo, se m = 3 e a primeira janela contém os bits 3-6, a próxima janela consiste dos bits 4-7. Anote o número de ocorrências de B em cada bloco, incrementando um vetor v_i (onde i = 0, ..., 5), de tal forma que v_0 é incrementado quando não

há ocorrências de B, v_1 é incrementado para uma ocorrência de B, ... e v_5 é incrementado para 5 ou mais ocorrências de B.

Para o exemplo acima, se $m = 2$ então $B = 11$, tem-se o exame do primeiro bloco conforme Tabela 3.3.

Tabela 3.3: Análise do primeiro bloco.

Posição do bit	**Bit**	**N° de ocorrências de B**
1-2	10	0
2-3	00	0
3-4	01	0
4-5	10	0
5-6	00	0
7-8	11	Incrementa para 1
8-9	11	Incrementa para 2
9-10	10	2

Assim, após o bloco 1, há duas ocorrências de 11, v_2 é incrementado, e $v_0 = 0$, $v_1 = 0$, $v_2 = 1$, $v_3 = 0$, $v_4 = 0$ e $v_5 = 0$. De maneira semelhante, os blocos 2-5 são examinados. No bloco 2, há duas ocorrências de 11, então v_2 é incrementado. No bloco 3, existem duas ocorrências de 11, então v_2 é incrementado. No bloco 4, há três ocorrências de 11, então v_3 é incrementado. No bloco 5, há uma ocorrência de 11, então v_1 é incrementado. Portanto, $v_0 = 0$, $v_1 = 1$, $v_2 = 3$, $v_3 = 1$, $v_4 = 0$ e $v_5 = 0$ depois que todos os blocos forem examinados.

(3) Calcular os valores de λ e η que serão usados para calcular as probabilidade teóricas π_i correspondentes às classes de v_0.

$$\lambda = \frac{(M - m + 1)}{2^m} \tag{3.3}$$

$$\eta = \frac{\lambda}{2} \tag{3.4}$$

Para o exemplo desta sessão: $\lambda = \frac{(10-2+1)}{2^2} = 2,25$ e $\eta = \frac{2,25}{2} = 1,125$.

(4) Calcular $\chi^2(obs) = \sum_{1=0}^{5} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, onde os valores de π_0, π_1, π_2, π_3, π_4 e π_5, são calculados conforme as fórmulas a seguir dadas por [Rukhin et al., 2010] e [Hamano e Kaneko, 2007]:

$$\pi_0 = e^{-\eta} \tag{3.5}$$

$$\pi_1 = \frac{\eta}{2}e^{-\eta} \tag{3.6}$$

$$\pi_2 = \frac{\eta e^{-\eta}}{8}[\eta + 2] \tag{3.7}$$

$$\pi_3 = \frac{\eta e^{-\eta}}{8}[\frac{\eta^2}{6} + \eta + 1] \tag{3.8}$$

$$\pi_4 = \frac{\eta e^{-\eta}}{16}[\frac{\eta^3}{24} + \frac{\eta^2}{2} + \frac{3\eta}{2} + 1] \tag{3.9}$$

$$\pi_5 = 1 - \pi_4 + \pi_3 + \pi_2 + \pi_1 + \pi_0. \tag{3.10}$$

onde o valor de η foi calculado no passo 3 dessa seção.

Para o exemplo desta seção: $\chi^2(obs) = \frac{(0-5*0,3246)^2}{5*0,3246} + \frac{(1-5*0,1826)^2}{5*0,1826} + \frac{(3-5*0,1426)^2}{5*0,1426} + \frac{(1-5*0,1066)^2}{5*0,1066} + \frac{(0-5*0,0771)^2}{5*0,0771} + \frac{(0-5*0,1662)^2}{5*0,1662} = 10,5871$, onde $\pi_0 = 0,3246$, $\pi_1 = 0,1826$, $\pi_2 = 0,1426$, $\pi_3 = 0,1066$, $\pi_4 = 0,0771$ e $\pi_5 = 0,1662$, que foram calculados conforme o passo 3.

(5) Calcular o $p - valor = igmac(\frac{5}{2}, \frac{\chi^2(obs)}{2})$, onde $igmac$ é a função gama incompleta definada na Seção A.2 do Apêndice A.

Para o exemplo desta seção: $p - valor = igmac(\frac{5}{2}, \frac{10,5871}{2}) = 0.06020$.

Conclusão e interpretação do resultado

Como o resultado do $p - valor$ obtido no passo 4 é $\geq$ 0,01, a conclusão é que não há evidência, a um nível de significância de 0,01, de que em algum bloco há um padrão periódico B.

Exemplo

Entrada (E) = 00100111101010011101011101010111010001101001001001
11101011101001011111011011100100011001100100111100;
Entrada (n) = 100;
Entrada (B) = 111;
Processamento $(v_i) = v_0 = 3$, $v_1 = 3$, $v_2 = 3$, $v_3 = 1$, $v_4 = 0$ e $v_5 = 0$;

Processamento $(\chi^2(obs)) = 8.6269$;
Saída $(p - valor) = 0.1248$;
Conclusão = Como o $p - valor$ é $\geq$ 0,01, a sequência é considerada aleatória.

Os testes aqui descritos, juntamente com as funções matemáticas aqui utilizadas, foram reunidos afim de, se construir um aplicativo que analisa arquivos criptográficos e retorna resultados, de forma que, seja possível, verificar se o algoritmo criptográfico está se portando da forma esperada. No Capítulo 4 será demonstrado, o funcionamento do aplicativo.

Capítulo 4

Aplicativo

Neste capítulo é apresentado o aplicativo construído tendo como base os testes expostos no Capítulo 3. Tal aplicativo foi criado a fim de facilitar a utilização dos testes apresentados, já que possui uma interface amigável e funcionalidades que tornam mais fácil a inserção dos dados de entrada e a visualização dos resultados obtidos.

O aplicativo criado é apenas um protótipo, assim sendo, possui apenas as funcionalidades básicas referentes à utilização dos testes estatísticos e não fornece todos os testes da bateria do NIST descritos na seção 2.6.1.

Existem algumas implementações em C, e em outras linguagens estruturais, dos testes estatísticos empregados pelo NIST, entretanto, pouco se sabe sobre implementações orientadas a objetos. Visando facilitar o processo de modularização dos teste, optou-se por se fazer uma implementação orientada a objetos. Pois linguagens de tal paradigma permitem mais facilidade em relação ao reuso do código criado e possuem modelagem que descreve as estruturas e comportamentos presentes no mundo real, dentre outras vantagens.

A linguagem de programação Java foi a escolhida para o desenvolvimento do protótipo aqui proposto. Tal escolha se funda na existência abundante de documentação da linguagem e de bibliotecas que facilitam a incorporação dos conceitos matemáticos aqui utilizados, além de já haver experiência prévia da utilização de tal linguagem.

Por ter como objetivo oferecer uma maior facilidade para o usuário de testes estatísticos aplicados a arquivos para sua verificação, buscou-se por opções que facilitassem esse propósito e que reunidas formassem uma interface simples e intuitiva.

A interface criada para o aplicativo proposto neste trabalho é dividida em duas abas, onde o usuário escolhe o teste e realiza as entradas de dados na primeira e pode analisar os resultados obtidos na segunda.

4.1 Aba de Testes

Na Figura 4.1 pode-se visualizar a tela inicial do protótipo, que é sua primeira aba e possibilita a escolha dos testes a serem realizados e a inserção dos dados necessários para isso.

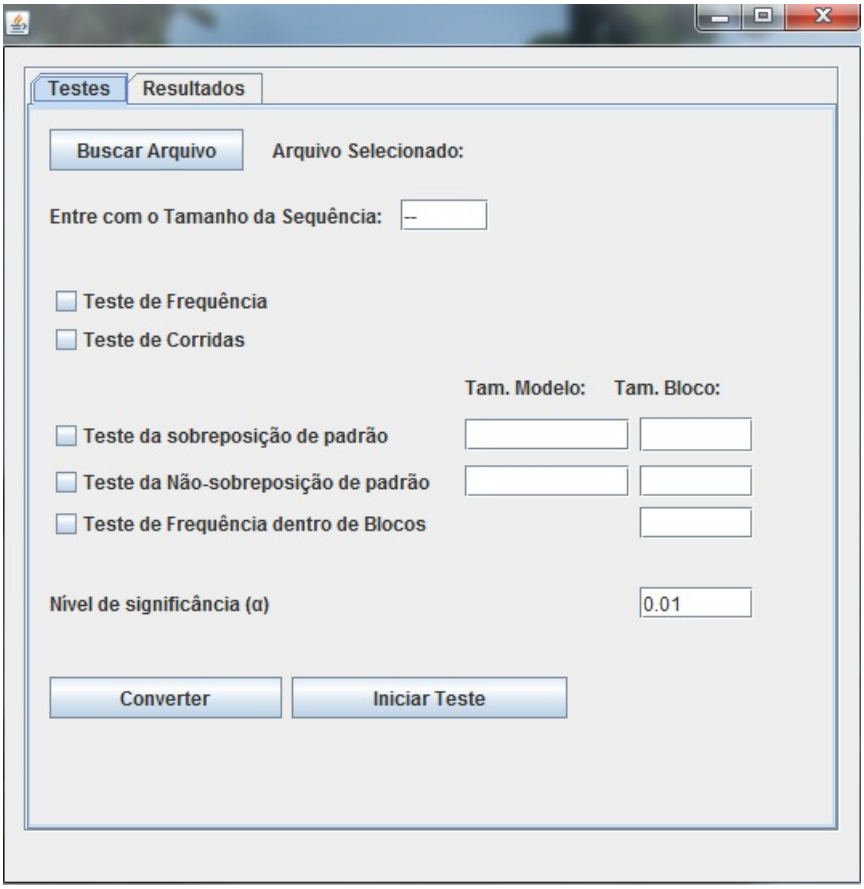

Figura 4.1: Interface do protótipo.

As opções disponíveis para a aba "Testes" são:

- Botão Buscar arquivo: Os arquivos a passarem pelos testes são escolhidos através deste botão, que quando clicado abre a caixa de diálogo vista na Figura 4.2;
- Entra com o tamanho da sequência: Aqui é onde insere-se o tamanho da sequência de bits que se pretende analisar, pois é possível analisar apenas parte do arquivo selecionado, dependendo da escolha do usuário;
- *Checkbox* teste de frequência: Quando selecionado indica que o teste de frequência será aplicado à sequência escolhida do arquivo selecionado;
- *Checkbox* teste de corrida: Quando selecionado indica que o teste de corrida será aplicado à sequência escolhida do arquivo selecionado;

- *Checkbox* teste de não sobreposição de padrão: Quando selecionado indica que o teste de não sobreposição de padrão será aplicado à sequência escolhida do arquivo selecionado, caso isso ocorra é necessário entrar com o tamanho do modelo e o tamanho do bloco a serem utilizados;
- *Checkbox* teste de sobreposição de padrão: Quando selecionado indica que o teste de sobreposição de padrão será aplicado à sequência escolhida do arquivo selecionado, caso isso ocorra é necessário entrar com o tamanho do modelo e o tamanho do bloco a serem utilizados;
- *Checkbox* teste de frequência dentro de bloco: Opção para ativar o teste de frequência dentro de bloco. Nele é necessário entrar com o tamanho do bloco;
- Nível de significância (α): Aqui insere-se o valor do nível de significância, o qual é comparado com o resultado do p-valor, para saber se a sequência está aprovada ou não em algum teste.
- Botão Converter: Quando o arquivo selecionado apresenta sua codificação em formato ASCII, esta opção converte tal arquivo para o formato binário, tornando-o hábil a ser utilizado na análise;
- Botão Iniciar Teste: Quando acionado esse botão, os testes selecionados através das *checkbox* são então executados sobre o arquivo selecionado.

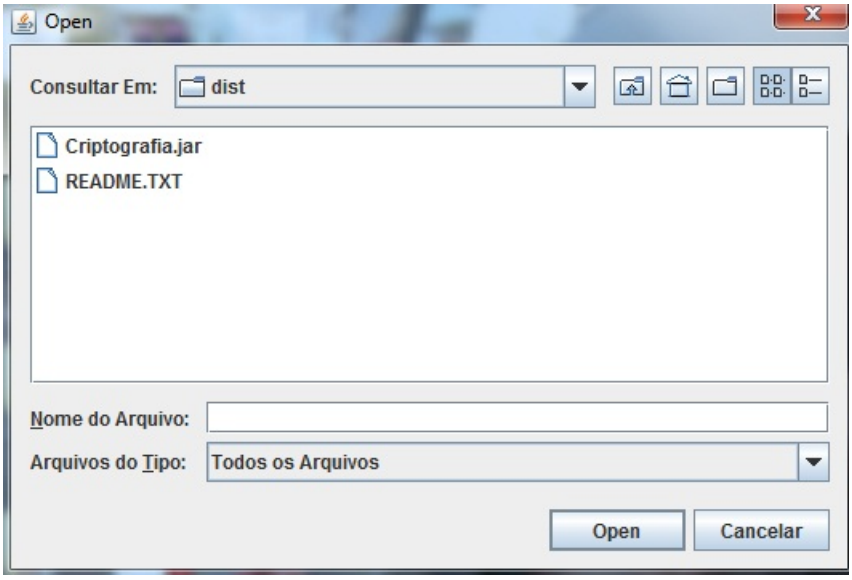

Figura 4.2: Seleciona o arquivo a ser analisado.

4.1.1 Inserção de dados

A entrada de dados para a análise pode ocorrer de duas maneiras:

- Inserção de arquivos contendo apenas números binários (0's e 1's): Através do botão *BuscarArquivo*, uma janela de busca de arquivo, que pode ser visualizada na Figura 4.2, onde é selecionado o arquivo a ser utilizado, nesse tipo de inserção, o arquivo a ser analisado deve conter apenas dígitos binários.

- Inserção de arquivos contendo caracteres no formato ASCII: Quando utilizado esse tipo de arquivo, também através da seleção do botão *BuscarArquivo*, é necessário utilizar a opção fornecida pelo botão converter, já que os testes analisam apenas bits. O texto em ASCII então é convertido em um arquivo contendo apenas números binários, chamado *binario.txt* com os *bytes* correspondentes e pode assim ser analisado pelo aplicativo.

4.2 Aba de Resultados

Após a inserção do arquivo a ser analisado e a seleção dos testes a serem aplicados, o processo de análise pode ser inicializado e seu resultado observado através da aba resultados (c.f. Figura 4.3), nela é possível visualizar o nome do teste, seu p-valor resultante e seu resultado (aprovado ou reprovado).

Figura 4.3: Resultados da análise.

Para o teste de não sobreposição de padrão, devido ao grande número de p-valores gerados (como já descrito na Seção 3.1.4), os resultados são armazenados em um arquivo à parte chamado *arquivo.txt*, que pode ser visualizado na Figura 4.4. Note que no arquivo obtêm-se as mesmas informações que as contidas no protótipo, como o resultado do p-valor e a situação da sequência no teste, o diferencial deste arquivo é o fato de que também é mostrado o modelo correspondente ao resultado do p-valor.

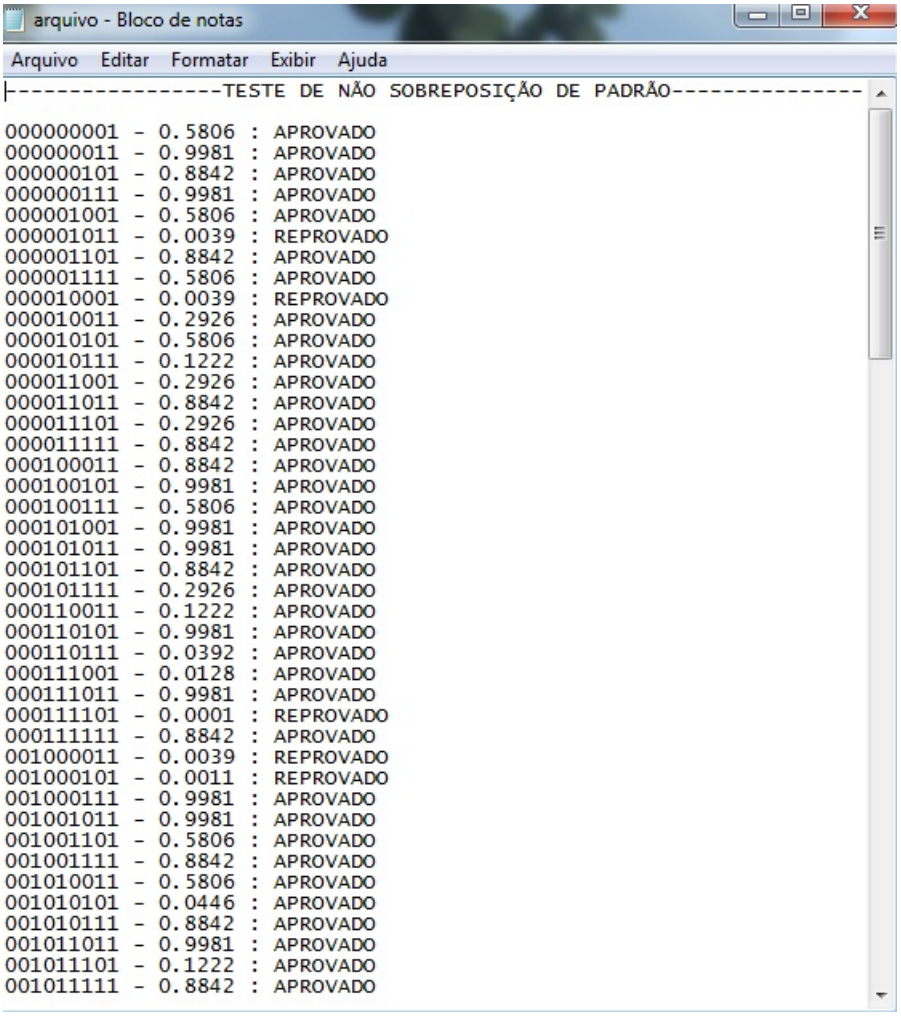

Figura 4.4: Resultados dos p-valores do teste de não sobreposição de padrão.

Como foi possível observar, o protótipo possui uma interface aparentemente amigável e de fácil manipulação. Os resultados são obtidos de uma maneira simples podendo ser visualizados na aba resultado do próprio protótipo. No Capítulo 5, serão demonstrados experimentos feitos através do protótipo.

Capítulo 5

Resultados Utilizando o Aplicativo

Neste capítulo serão apresentados os resultados obtidos utilizando o protótipo criado neste trabalho e apresentado no Capítulo 4, ao todo são 5 arquivos que passaram pela análise: três arquivos de teste fornecidos pelo NIST, um arquivo de texto claro e um arquivo criptografado utilizando o AES. O valor de significância (α) utilizado foi fixado em 0,01. A quantidade de p-valores do teste de não sobreposição de padrão varia de acordo com o tamanho de seu padrão, se algum p-valor for reprovado, ou seja, p-valor $<$ 0,01, deve ser feita uma análise para verificar se esse comportamento é típico do gerador que foge do escopo do trabalho.

5.1 Resultado Dos Arquivos Fornecidos Pelo NIST

Nessa seção serão apresentados os dados de entrada utilizados e os resultados obtidos através da utilização dos arquivos de teste *data.sqrt2*, *data.pi* e *data.e* que são fornecido pelo NIST[1] e podem ser encontrado no CD que acompanha essa monografia.

5.1.1 Arquivo *data.sqrt2*

Para a execução do protótipo, alguns parâmetros necesários foram definidos:

- Tamanho da sequência = 10.000;
- Tamanho do modelo para o teste de sobreposição de padrão = 9;
- Tamanho do modelo para o teste de não sobreposição de padrão = 9;
- Tamanho do bloco para o teste de frequência dentro de bloco = 100;
- Tamanho do bloco para o teste de sobreposição de padrão = 1032;

[1]Disponível online em: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html, arquivo sts-2.1.1

- Tamanho do bloco para o teste de não sobreposição de padrão = 1250.

A Tabela 5.1 apresenta os resultados obtidos a partir do protótipo para os 5 testes implementados. Nessa tabela foi colocado apenas 1 p-valor (referente ao modelo 000000001) dos 148 possíveis para o teste de não sobreposição de padrão, os demais resultados podem ser apreciados na Tabela C.1 do Apêndice C.

Tabela 5.1: Resultado do arquivo *data.sqrt2* fornecido pelo NIST.

Nome do teste	**P-valor**
Teste de frequência	0.5353
Teste de frequência dentro de bloco	0.4071
Teste de corrida	0.4989
Teste de sobreposição de padrão	0.0402
Teste de não sobreposição de padrão	0.3812

5.1.2 Arquivo *data.pi*

Para a execução do protótipo alguns parâmetros necesários foram definidos:

- Tamanho da sequência = 10.000;
- Tamanho do modelo para o teste de sobreposição de padrão = 9;
- Tamanho do modelo para o teste de não sobreposição de padrão = 8;
- Tamanho do bloco para o teste de frequência dentro de bloco = 100;
- Tamanho do bloco para o teste de sobreposição de padrão = 1032;
- Tamanho do bloco par ao teste de não sobreposição de padrão = 1250.

A Tabela 5.2 apresenta os resultados obtidos a partir do protótipo para os 5 testes implementados. Nessa tabela foi colocado apenas 1 p-valor (referente ao modelo 00000001) dos 74 possíveis para o teste de não sobreposição de padrão, os demais resultados podem ser vistos na Tabela C.4 do Apêndice C.

Tabela 5.2: Resultado do arquivo *data.pi* fornecido pelo NIST.

Nome do teste	**P-valor**
Teste de frequência	0.7795
Teste de frequência dentro de bloco	0.4344
Teste de corrida	0.8565
Teste de sobreposição de padrão	0.8005
Teste de não sobreposição de padrão	0.8110

5.1.3 Arquivo *data.e*

Para a execução do protótipo alguns parâmetros necesários foram definidos:

- Tamanho da sequência = 10.000;
- Tamanho do modelo para o teste de sobreposição de padrão = 9;
- Tamanho do modelo para o teste de não sobreposição de padrão = 8;
- Tamanho do bloco para o teste de frequência dentro de bloco = 100;
- Tamanho do bloco para o teste de sobreposição de padrão = 1032;
- Tamnho do bloco par ao teste de não sobreposição de padrão = 1250.

A Tabela 5.3 apresenta os resultados obtidos a partir do protótipo para os 5 testes implementados. Nessa tabela foi colocado apenas 1 p-valor (referente ao modelo 00000001) dos 74 possíveis para o teste de não sobreposição de padrão, os demais resultados podem ser conferidos na Tabela C.6 do Apêndice C.

Tabela 5.3: Resultado do arquivo *data.e* fornecido pelo NIST.

Nome do teste	**P-valor**
Teste de frequência	0.6745
Teste de frequência dentro de bloco	0.2396
Teste de corrida	0.7655
Teste de sobreposição de padrão	0.7387
Teste de não sobreposição de padrão	0.8674

5.2 Resultado Para Um Texto Claro

Nesta seção será apresentado o resultado do arquivo *texto_ claro.rtf*, que é fornecido no Apêndice D. Vale ressaltar que a formatação do arquivo interfere nos resultado dos testes, devido ao fato de conter espaços a mais ou a menos.

5.2.1 Arquivo *texto_ claro.rtf*

Para a execução do protótipo alguns parâmetros necesários foram definidos:

- Tamanho da sequência = 10.000;
- Tamanho do modelo para o teste de sobreposição de padrão = 9;
- Tamanho do modelo para o teste de não sobreposição de padrão = 8;
- Tamanho do bloco para o teste de frequência dentro de bloco = 100;
- Tamanho do bloco para o teste de sobreposição de padrão = 1032;
- Tamnho do bloco par ao teste de não sobreposição de padrão = 1250.

A tabela 5.4 apresenta os resultados obtidos a partir do protótipo para os 5 testes implementados. Nessa tabela foi colocado apenas 1 p-valor (referente ao modelo 00000001) dos 74 possíveis para o teste de não sobreposição de padrão, os demais resultados podem serem visto na Tabela C.8 do Apêndice C.

Tabela 5.4: Resultado do arquivo *texto_ claro.rft*.

Nome do teste	**P-valor**
Teste de frequência	0.0000
Teste de frequência dentro de bloco	0.1847
Teste de corrida	0.0001
Teste de sobreposição de padrão	0.0086
Teste de não sobreposição de padrão	0.0012

Como pode ser observado na Tabela 5.4, o arquivo *texto_ claro.rtf* foi reprovado no teste de frequência e não necessitaria ser submetido aos demais testes, pois a sequência seria considerada não aleatória.

5.3 Resultado Para o Arquivo Criptografado pelo AES

Nesta seção será apresentado o resultado do arquivo *texto_ cifrado_ Aes*. Para a cifragem do arquivo foi utilizado um programa que criptografa o arquivo utilizando o algoritmo criptográfico AES [2], que pode ser encontrado no CD que acompanha essa monografia. O algorimto AES foi publicado pelo NIST em 2001 com o objetivo de substituir o DES que até então era adotado como padrão pelo NIST desde 1977.

5.3.1 Arquivo *texto_ cifrado_ Aes*

Para a execução do protótipo, alguns parâmetros necesários foram definidos:

- Tamanho da sequência = 10.000;
- Tamanho do modelo para o teste de sobreposição de padrão = 9;
- Tamanho do modelo para o teste de não sobreposição de padrão = 8;
- Tamanho do bloco para o teste de frequência dentro de bloco = 100;
- Tamanho do bloco para o teste de sobreposição de padrão = 1032;
- Tamanho do bloco para o teste de não sobreposição de padrão = 1250.

A Tabela 5.5 apresenta os resultados obtidos a partir do protótipo para os 5 testes implementados. Nessa tabela foi colocado apenas 1 p-valor (referente ao modelo 000000001) dos 74 possíveis para o teste de não sobreposição de padrão, os demais resultados podem ser apreciados na Tabela C.10 do Apêndice C.

Tabela 5.5: Resultado do arquivo *texto_ cifrado_ Aes*.

Nome do teste	**P-valor**
Teste de frequência	0.9203
Teste de frequência dentro de bloco	0.2959
Teste de corrida	0.3575
Teste de sobreposição de padrão	0.2743
Teste de não sobreposição de padrão	0.5130

Como pode ser observado na Tabela 5.5, o arquivo *texto_ cifrado_ Aes* foi aprovado em todos os testes. Isso já era esperado, pois o AES é o algoritmo padrão do NIST.

[2]Disponível online em: http://novatec.com.br/livros/criptografia/

Capítulo 6

Conclusão

O objetivo deste trabalho foi a construção de um protótipo de sistema baseado no conjunto de testes estatísticos fornecidos pelo NIST, onde o criptógrafo entra com um arquivo cifrado e obtém um resultado (p-valor) que pode, como um passo inicial, dizer se o algoritmo criptográfico utilizado para cifrar o arquivo está comportando-se de maneira aleatória.

Alguns conceitos como criptoanálise, geradores de números aleatório e pseudoaleatório, estatística p-valor e a bateria de teste estatístico do NIST foram abordados para que o trabalho pudesse ser realizado.

Como já dito anteriormente, atualmente com o disseminado uso das redes de computadores, a criptografia funda sua base na criação e aprimoramento de algoritmos criptográfico para aumentar a segurança dos dados. Os testes estatisticos vem, então, como um passo inicial, para verificar se o algoritmo criptográfico está se portando como um verdadeiro gerador de números aleatórios.

Optou-se por uma implementação orientada a objetos, visando a fácil aclopação a eventuais sistemas criptográficos, seja testes ou geradores de números pseudoaleatórios. Ressalta-se que não foi encontrado na literatura nenhum sistema livre com os objetivos do protótipo desenvolvido.

Evidente que durante o processo de confecção do trabalho final, houveram algumas dificuldades. Destas, pode-se destacar a compreensão das fórmulas matemáticas empregadas[1] e o desenvolvimento das mesmas em Java, também foi motivo de grande trabalho compreender e ajustar a manipulação dos arquivos tratados pelo protótipo.

Estas dificuldades seriam justificativas para a não implementação de todos testes estatísticos recomendados pelo NIST.

Contudo, tal protótipo já pode ser empregado na análise de aleatoriedade e pode ser facilmente incrementado uma vez que as principais funções matemáticas usadas já foram

[1] Formúlas desevolvidas em [Rukhin et al., 2010].

incorporadas.

6.1 Trabalhos Futuros

Este trabalho apresentou uma introdução de um grande campo de pesquisas, envolvendo matemática computacional e criptografia. Portanto, há ainda muito o que se desenvolver, a saber:

- **Incorporação de outros testes**: O protótipo tem implementado 5 testes estatísticos da bateria do NIST. Futuralmente pode-se implementar os demais testes dessa bateria, bem com outros testes que existem, como por exemplo os da bateria DIEHARD.
- **Engenharia de *Software***: Por ser ainda apenas um protótipo, as técnicas de engenharia de *software* como diagramas de fluxo, sequência entre outros não foram utilizados, além de testes de usabilidade.
- **Verificação Formal:** Uma vez que, tal sistema deve ser empregado para validar, em primeira instância, sistemas críticos, esse deve passar por um crivo mais robusto de confiabilidade. Neste sentido, seria necessário o emprego de técnicas de verificação formal de sistemas no produto completo produzido.

Referências

Almeida, G. e Appelt, R. (2003). Escrita escondida - criptografia - parte.

Biham, E. e Shamir, A. (1993). Differential cryptanalysis of the *data encryption standard*.

Cavalcante, A. (2005). Teoria dos números e criptografia. *Revista Virtual*.

Davis, D., Ihaka, R., e Fenstermacher, P. (1994). Cryptographic randomness from air turbulence in disk drives. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 114–120, London, UK. Springer-Verlag.

de Souza, W. (2011). *Identificação de Contextos Linguísticos em Linguagens Desconhecidas Geradas por Cifras de Blocos*. PhD thesis, Universidade Federal do Rio de Janeiro.

do Nascimento, R. (2005). Criptografia tradicional simétrica e criptografia de chave pública. análise das vantagens e desvantagens. *Revista Cesubra Scientia - Revista do Centro Universitário Planalto do Distrito Federal*, page 595.

Fairfield, R. C., Mortenson, R. L., e Coulthart, K. B. (1985). An LSI random number generator (RNG). In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 203–230, New York, NY, USA. Springer-Verlag New York, Inc.

Gutmann, P. (1998). Software generation of random numbers for cryptographic purposes. In *Proceedings of the 1998 Usenix Security Symposium*, pages 243–257.

Hamano, K. e Kaneko, T. (2007). Correction of overlapping template matching test included in nist randomness test suite. *IEICE Trransactions on Fundamentals of Eletronics Communications and Computer Sciences e Series A*, 90(9):1788.

Kahn, D. (1996). *The Codebreakers: the story of secret writing*. Scribner Book Company.

Knuth, D. (1981). *The art of computer programming. Volume 2, seminumerical algorithms*. Addidon-Wesley.

L'Ecuyer, P. (1992). Testing random number generators. In *Winter Simulation Conference*, pages 305–305. Association for Computing Machinery.

Marsaglia, G. (1985). A current view of random number generators. In *Computer Science and Statistics, Sixteenth Symposium on the Interface. Elsevier Science Publishers, North-Holland, Amsterdam*, pages 3–10.

Marsaglia, G. (1996). Diehard: a battery of tests of randomness. *See http://stat. fsu. edu/ geo/diehard. html*.

Martins, A. (2005). Elementos de criptologia: uma aplicação da álgebra.

Matsui, M. (1994). Linear cryptanalysis method for des cipher. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 386–397, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Mendes, A. (2007). Estudo de criptografia com chave pública baseada em curvas elípticas. *Monografia Depto de Ciências, Montes Claros*.

Mendes, A., Paulicena, E., e de Souza, W. (2011). Criptografia quântica: Uma abordagem direta. *Revista de Sistemas de Informação da FSMA*, (7):39–48.

Menezes, A., Van Oorschot, P., e Vanstone, S. (1997). *Handbook of applied cryptography*. CRC.

Moreira, N. S. (2001). *Segurança Mínima - Uma visão Corporativa da Segurança de Informações*. Axcel Books, São Paulo, SP, Brasil.

Murphy, S. (1990). The cryptanalysis of feal-4 with 20 chosen plaintexts. *Journal of Cryptology*, 2(3):145–154.

Paulo P. Flor, J., Luis Monteiro, J., e Hebeda, S. (2007). Um método criptográfico utilizando geometria analítica - cifra de hill.

Plumb, C. (1994). Truly random numbers. *Dr. Dobbs Journal*, 19(13):113–115.

Press, W., Flannery, B., Teukolsky, S., e Vetterling, W. (1992). *Numerical recipes in C. The art of scientific computing*. Cambridge Univ Press, 2^a edition.

Resende, A. e da Costa, V. (2011). Utilização de testes estatísticos para verificação de eficácia de algoritmos criptográficos. In *IX Encontro Anual de Computação - ENACOMP*, Catalão-GO. Brasil.

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. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications.

Sandberg, J. (1995). Netscape's Internet software contains flaw that jeopardizes security of data. *The Wall Street Journal Western Edition*.

Schneier, B. e Sutherland, P. (1995). *Applied cryptography: protocols, algorithms, and source code in C*. John Wiley & Sons, Inc. New York, NY, USA.

Soto, J. (1999). Statistical testing of random number generators. In *Proceedings of the 22nd National Information Systems Security Conference*, volume 10, page 99. NIST Gaithersburg, MD.

Soto, J. (2000). Randomness testing of the aes candidate algorithms. *NIST. Available via csrc. nist. gov*.

Stallings, W. (2008). *Criptografia e Segurança de Redes: Princípios e Práticas*. Pearson Education, 4^a edition.

Vieira, C. e Ribeiro, C. (2004). Um estudo comparativo entre três geradores de números aleatórios.

Vieira, C., Ribeiro, C., e e Souza, R. (2004). *Geradores de números aleatórios*. PUC.

ZATTI, S. e Beltrame, A. (2009). A presença da álgebra linear e da teoria dos números na criptografia.

Zimmermann, P. (1995). *PGP source code and internals*. MIT Press, Cambridge, MA, USA.

Apêndice A

Fórmulas Matemáticas

Aqui são demonstradas as funções erro complementar[1] e a função gama incompleta[2] que são utilizadas para a implementação dos 5 testes.

A.1 Função Erro Complementar

Esta função é utilizada em estatística para prever o comportamento de uma amostra em relação à média da população.

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-u^2} du \tag{A.1}$$

A.2 Função Gama Incompleta

Essa função é usada extensivamente em testes de hipóteses e foi abordada nos trabalhos do NIST.

$$Q(a,x) = \frac{\gamma(a,x)}{\Gamma(a)} = \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} dt \tag{A.2}$$

onde,

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{A.3}$$

[1]Disponível online: http://www.fisica.ufs.br/egsantana/estadistica/maxwell/maxwell.html.
[2]Disponível online: http://commons.apache.org/math/download_math.cgi.

Apêndice B

Modelos aperiódicos para $6 \leq m \leq 9$

Tabela B.1: Padrões aperiódicos para $6 \leq m \leq 8$. [Rukhin et al., 2010].

m = 6	**m = 7**	**m = 7**	**m = 8**	**m = 8**	**m = 8**	**m = 8**
000001	0000011	1010100	00000001	00110111	10110000	11110000
000011	0000101	1011000	00000011	00111011	10110100	11110010
000101	0000111	1011100	00000101	00111101	10111000	11110100
000111	0001001	1100000	00000111	00111111	10111100	11110110
001011	0001011	1100010	00001001	01000011	11000000	11111000
001101	0001101	1100100	00001011	01000111	11000010	11111010
001111	0001111	1101000	00001101	01001011	11000100	11111100
010011	0010011	1101010	00001111	01001111	11001000	11111110
010111	0010101	1101100	00010011	01010011	11001010	
011111	0010111	1110000	00010101	01010111	11010000	
100000	0011011	1110010	00010111	01011011	11010010	
101000	0011101	1110100	00011001	01011111	11010100	
101100	0011111	1110110	00011011	01100111	11011000	
110000	0100011	1111000	00011101	01101111	11011010	
110010	0100111	1111010	00011111	01111111	11011100	
110100	0101011	1111100	00100011	10000000	11100000	
111000	0101111	1111110	00100101	10010000	11100010	
111010	0110111		00100111	10011000	11100100	
111100	0111111		00101011	10100000	11100110	
111110	1000000		00101101	10100100	11101000	
	1001000		00101111	10101000	11101010	
	1010000		00110101	10101100	11101100	

Tabela B.3: Padrões aperiódicos para m = 9. [Rukhin et al., 2010].

m = 9	**m = 9**	**m = 9**	**m = 9**	**m = 9**
000000001	001001101	011000111	110010010	111101000
000000011	001001111	011001111	110010100	111101010
000000101	001010011	011010111	110011000	111101100
000000111	001010101	011011111	110011010	111101110
000001001	001010111	011101111	110100000	111110000
000001011	001011011	011111111	110100010	111110010
000001101	001011101	100000000	110100100	111110100
000001111	001011111	100010000	110101000	111110110
000010001	001100101	100100000	110101010	111111000
000010011	001100111	100101000	110101100	111111010
000010101	001101011	100110000	110110000	111111100
000010111	001101101	100111000	110110010	111111110
000011001	001101111	101000000	110110100	
000011011	001110101	101000100	110111000	
000011101	001110111	101001000	110111010	
000011111	001111011	101001100	110111100	
000100011	001111101	101010000	111000000	
000100101	001111111	101010100	111000010	
000100111	010000011	101011000	111000100	
000101001	010000111	101011100	111000110	
000101011	010001011	101100000	111001000	
000101101	010001111	101100100	111001010	
000101111	010010011	101101000	111001100	
000110011	010010111	101101100	111010000	
000110101	010011011	101110000	111010010	
000110111	010011111	101110100	111010100	
000111001	010100011	101111000	111010110	
000111011	010100111	101111100	111011000	
000111101	010101011	110000000	111011010	
000111111	010101111	110000010	111011100	
001000011	010110011	110000100	111100000	
001000101	010110111	110001000	111100010	
001000111	010111011	110001010	111100100	
001001011	010111111	110010000	111100110	

Apêndice C

Resultados dos p-valores do teste de não sobreposição de padrão

Tabela C.1: Resultado do teste de não sobreposição de padrão para o arquivo data.sqrt2 fornecido pelo NIST.

Modelo	**P-valor**	**Modelo**	**P-valor**	**Modelo**	**P-valor**
000000001	0.3812	000100111	0.8813	001010011	0.8273
000000011	0.1860	000101001	0.7397	001010101	0.0091
000000101	0.3699	000101011	0.4883	001010111	0.4163
000000111	0.5518	000101101	0.8212	001011011	0.1062
000001001	0.5933	000101111	0.1894	001011101	0.4437
000001011	0.0781	000110011	0.2479	001011111	0.3563
000001101	0.6599	000110101	0.7865	001100101	0.5933
000001111	0.1860	000110111	0.7800	001100111	0.3195
000010001	0.2479	000111001	0.7734	001101011	0.3729
000010011	0.4437	000111011	0.7994	001101101	0.6669
000010101	0.7600	000111101	0.4914	001101111	0.4163
000010111	0.0594	000111111	0.0714	001110101	0.9198
000011001	0.9154	001000011	0.3145	001110111	0.7465
000011011	0.7800	001000101	0.2356	001111011	0.0467
000011101	0.5794	001000111	0.2181	001111101	0.0000
000011111	0.0085	001001011	0.5015	001111111	0.7191
000100011	0.5214	001001101	0.8511	010000011	0.5015
000100101	0.5147	001001111	0.5112	010000111	0.3046

Modelo	P-valor	Modelo	P-valor	Modelo	P-valor
010001011	0.0358	101100100	0.4163	111001100	0.3618
010001111	0.7800	101101000	0.9198	111010000	0.2397
010010011	0.8568	101101100	0.3812	111010010	0.7329
010010111	0.6739	101110000	0.0864	111010100	0.3699
010011011	0.9588	101110100	0.5655	111010110	0.7865
010011111	0.0581	101111000	0.1762	111011000	0.3046
010100011	0.5655	101111100	0.0743	111011010	0.0351
010100111	0.6599	110000000	0.9588	111011100	0.6599
010101011	0.2438	110000010	0.0003	111100000	0.4689
010101111	0.5449	110000100	0.0425	111100010	0.9198
010110011	0.7533	110001000	0.6880	111100100	0.2564
010110111	0.0298	110001010	0.7020	111100110	0.4103
010111011	0.3644	110010000	0.0194	111101000	0.8394
010111111	0.3246	110010010	0.7600	111101010	0.1454
011000111	0.9062	110010100	0.0830	111101100	0.4625
011001111	0.4753	110011000	0.6143	111101110	0.0917
011010111	0.4407	110011010	0.9324	111110000	0.8087
011011111	0.8568	110100000	0.5724	111110010	0.8334
011101111	0.7465	110100010	0.9899	111110100	0.1337
011111111	0.3095	110100100	0.1716	111110110	0.0017
100000000	0.3812	110101000	0.0899	111111000	0.3756
100010000	0.6599	110101010	0.0059	111111010	0.4689
100100000	0.2316	110101100	0.3455	111111100	0.1762
100101000	0.0273	110110000	0.6669	111111110	0.3095
100110000	0.7020	110110010	0.7800		
100111000	0.5724	110110100	0.8087		
101000000	0.6387	110111000	0.1021		
101000100	0.1400	110111010	0.4252		
101001000	0.3699	110111100	0.6317		
101001100	0.9324	111000000	0.3046		
101010000	0.7600	111000010	0.0457		
101010100	0.2997	111000100	0.4689		
101011000	0.8680	111000110	0.1427		
101011100	0.9242	111001000	0.0728		
101100000	0.8212	111001010	0.2997		

Tabela C.4: Resultado do teste de não sobreposição de padrão para o arquivo data.pi fornecido pelo NIST.

Modelo	**P-valor**	**Modelo**	**P-valor**	**Modelo**	**P-valor**
00000001	0.8110	01111111	0.4407	11111100	0.4718
00000011	0.8855	10000000	0.8110	11111110	0.4407
00000101	0.5069	10010000	0.2842		
00000111	0.8081	10011000	0.0508		
00001001	0.4877	10100000	0.9701		
00001011	0.7924	10100100	0.6470		
00001101	0.8855	10101000	0.7703		
00001111	0.9371	10101100	0.5611		
00010011	0.8447	10110000	0.6924		
00010101	0.4877	10110100	0.6784		
00010111	0.8774	10111000	0.4163		
00011001	0.8588	10111100	0.9579		
00011011	0.0570	11000000	0.6643		
00011101	0.2349	11000010	0.3335		
00011111	0.8110	11000100	0.5888		
00100011	0.4072	11001000	0.1131		
00100101	0.6784	11001010	0.6924		
00100111	0.4201	11010000	0.3456		
00101011	0.8149	11010010	0.8979		
00101101	0.8774	11010100	0.7865		
00101111	0.0813	11011000	0.1471		
00110101	0.9579	11011010	0.6394		
00110111	0.0821	11011100	0.9968		
00111011	0.7234	11100000	0.5069		
00111101	0.6892	11100010	0.6752		
00111111	0.1314	11100100	0.8206		
01000011	0.9738	11100110	0.9075		
01000111	0.0741	11101000	0.0643		
01001011	0.8855	11101010	0.3623		
01001111	0.3821	11101100	0.1671		
01010011	0.2708	11110000	0.4847		
01010111	0.2936	11110010	0.3383		
01011011	0.0478	11110100	0.7329		
01011111	0.0321	11110110	0.1857		
01100111	0.8447	11111000	0.0625		
01101111	0.1566	11111010	0.6708		

Tabela C.6: Resultado do arquivo data.e fornecido pelo NIST.

Modelo	**P-valor**	**Modelo**	**P-valor**	**Modelo**	**P-valor**
00000001	0.8674	01001111	0.2349	11011100	0.6232
00000011	0.6265	01010011	0.1863	11100000	0.3514
00000101	0.5952	01010111	0.1118	11100010	0.2514
00000111	0.6394	01011011	0.6956	11100100	0.6265
00001001	0.6956	01011111	0.5039	11100110	0.3232
00001011	0.0361	01100111	0.4081	11101000	0.3901
00001101	0.2175	01101111	0.1059	11101010	0.2100
00001111	0.6611	01111111	0.3589	11101100	0.2560
00010011	0.3131	10000000	0.8674	11110000	0.5111
00010101	0.0083	10010000	0.2688	11110010	0.9822
00010111	0.0897	10011000	0.3822	11110100	0.9445
00011001	0.7434	10100000	0.3326	11110110	0.4504
00011011	0.2842	10100100	0.4341	11111000	0.0441
00011101	0.0463	10101000	0.3032	11111010	0.4397
00011111	0.1995	10101100	0.0455	11111100	0.2349
00100011	0.3589	10110000	0.5984	11111110	0.3589
00100101	0.1082	10110100	0.1287		
00100111	0.4967	10111000	0.0494		
00101011	0.0750	10111100	0.1186		
00101101	0.3162	11000000	0.2312		
00101111	0.5952	11000010	0.1812		
00110101	0.4649	11000100	0.7805		
00110111	0.1211	11001000	0.9445		
00111011	0.3456	11001010	0.3649		
00111101	0.2374	11010000	0.0001		
00111111	0.4219	11010010	0.9947		
01000011	0.4136	11010100	0.6611		
01000111	0.1088	11011000	0.2580		
01001011	0.5920	11011010	0.0939		

Tabela C.8: Resultado do arquivo texto claro.

Modelo	**P-valor**	Modelo	**P-valor**	Modelo	**P-valor**
00000001	0.0012	01001011	0.0020	11011000	0.7159
00000011	0.0000	01001111	0.0000	11011010	0.0005
00000101	0.0823	01010011	0.0116	11011100	0.0000
00000111	0.2083	01010111	0.0019	11100000	0.1072
00001001	0.0388	01011011	0.0000	11100010	0.0000
00001011	0.0000	01011111	0.0000	11100100	0.0000
00001101	0.4108	01100111	0.0121	11100110	0.0000
00001111	0.0000	01101111	0.0000	11101000	0.4455
00010011	0.6189	01111111	0.0000	11101010	0.1334
00010101	0.0036	10000000	0.0012	11101100	0.1685
00010111	0.0008	10010000	0.0000	11110000	0.0000
00011001	0.0000	10011000	0.0000	11110010	0.0025
00011011	0.0246	10100000	0.0000	11110100	0.0001
00011101	0.2337	10100100	0.0094	11110110	0.4081
00011111	0.0000	10101000	0.0000	11111000	0.0000
00100011	0.0000	10101100	0.3077	11111010	0.0001
00100101	0.0012	10110000	0.1812	11111100	0.0000
00100111	0.4219	10110100	0.3185	11111110	0.0000
00101011	0.0000	10111000	0.0000		
00101101	0.0689	10111100	0.0025		
00101111	0.0000	11000000	0.0001		
00110101	0.0024	11000010	0.0000		
00110111	0.0000	11000100	0.1409		
00111011	0.5536	11001000	0.0000		
00111101	0.0000	11001010	0.0000		
00111111	0.0000	11010000	0.0008		
01000011	0.0001	11010010	0.0044		
01000111	0.0183	11010100	0.0008		

Tabela C.10: Resultado do arquivo texto_cifrado_Aes.

Modelo	**P-valor**	**Modelo**	**P-valor**	**Modelo**	**P-valor**
00000001	0.5130	01100111	0.8120	11110100	0.8933
00000011	0.0061	01101111	0.7954	11110110	0.9893
00000101	0.5813	01111111	0.6470	11111000	0.3100
00000111	0.1405	10000000	0.5130	11111010	0.8328
00001001	0.5442	10010000	0.7128	11111100	0.8855
00001011	0.2017	10011000	0.4788	11111110	0.6470
00001101	0.4522	10100000	0.4081		
00001111	0.2175	10100100	0.2066		
00010011	0.9053	10101000	0.9291		
00010101	0.7924	10101100	0.5877		
00010111	0.3874	10110000	0.5674		
00011001	0.4313	10110100	0.5568		
00011011	0.3108	10111000	0.3416		
00011101	0.6708	10111100	0.5920		
00011111	0.8420	11000000	0.6848		
00100011	0.9530	11000010	0.2849		
00100101	0.2778	11000100	0.6048		
00100111	0.3359	11001000	0.2892		
00101011	0.3927	11001010	0.2620		
00101101	0.6988	11010000	0.5769		
00101111	0.4465	11010010	0.3465		
00110101	0.5738	11010100	0.7052		
00110111	0.0361	11011000	0.3359		
00111011	0.4718	11011010	0.4397		
00111101	0.5008	11011100	0.4847		
00111111	0.6438	11100000	0.4777		
01000011	0.5706	11100010	0.5611		
01000111	0.8956	11100100	0.1307		
01001011	0.6816	11100110	0.4081		
01001111	0.3040	11101000	0.2175		
01010011	0.8979	11101010	0.3024		
01010111	0.5568	11101100	0.7340		
01011011	0.5738	11110000	0.6362		
01011111	0.1088	11110010	0.2778		

Apêndice D

Texto claro

Geradores de Números Aleatórios

Geradores de Números Aleatórios (RNG de Random Number Generators) utilizam uma fonte não-determinística (a fonte de entropia2), juntamente com algumas funções de processamento (o processo de destilação da entropia) para produzir aleatoriedade [Rukhin et al., 2010] e [Resende e da Costa, 2011]. As saídas desse tipo de gerador podem ser usadas diretamente como um número aleatório, nesse caso a saída precisa satisfazer critérios rigorosos de aleatoriedade, ou pode servir como entrada para geradores de números pseudoaleatórios [Schneier e Sutherland, 1995].

A melhor maneira de se obter números verdadeiramente aleatórios é utilizar como fonte medidas de fenômenos físicos, tais como decaimento radioativo, ruído termal em semicondutores, amostras de som retiradas de um ambiente barulhento, dentre outros [Gutmann, 1998].

Na literatura pode-se encontrar alguns artigos que tratam em detalhes da construção de RNGs usando fontes de entropia apropriadas. Dentre estes, pode-se citar o artigo de [Fairfield et al., 1985], que mostra como se gera um fluxo de bits aleatórios baseado na instabilidade da frequência de um oscilador.

Evidentemente que o construção de tais geradores requer um embasamento teórico/-prático que foge do escopo proposto neste trabalho. O que deve ficar claro com relação a tais geradores, é que poucos computadores (ou usuários) tem acesso a hardwares especializados necessários para analisar as fontes aleatórias e, portanto os mesmos precisam utilizar de outros métodos para obter dados aleatórios [Gutmann, 1998].

Existem abordagens que não precisam de hardwares especiais, tais como as que medem o tempo de turbulência de ar no movimento das cabeças de leitura do disco rígido [Davis et al., 1994] e as que medem o tempo usado para pressionar as teclas ao se inserir a senha de um usuário [Plumb, 1994] [Zimmermann, 1995]. Tais abordagens mostram-se inseguras se forem utilizadas em separado, contudo sua utilização se faz presente em

associações criptográficas envolvendo Geradores de Números Pseudoaleatórios (PRNG de Pseudo-Random Numbers Genarator), conforme seção 2.3.2.

Infelizmente, os conselhos sobre segurança da literatura são muitas vezes ignorados, o que acaba por resultar na produção de geradores de números aleatórios inseguros, os quais, por sua vez, produzem senhas criptográficas que são mais fáceis de serem atacadas do que o sistema criptográfico utilizado pela senha. Uma fonte popular de números aleatórios ruins é a que se baseia no tempo atual e no ID de processos. Este tipo de gerador ficou notoriamente conhecido no final de 1995, quando se quebrou a encriptação do navegador web Netscape consumindo aproximadamente 1 minuto. Devido a um espaço limitado de valores fornecidos por sua fonte é que se pode realizar a quebra do Netscape, tal fato causou tanta comoção que ganhou fama na impresa mundial [Sandberg, 1995].