Utilização de testes estatísticos para verificação de eficácia de algoritmos criptográficos

Amanda Cristina Davi Resende [1], **Vaston Gonçalves da Costa**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Goiás (UFG) – Campus Catalão
Avenida Dr. Lamartine Pinto de Avelar – 1120 – Setor Universitário –
CEP: 75704-020 – Catalão – GO – Brasil

{amandadavi7,vaston}@gmail.com

Abstract. *This paper presents the description of a prototype of system based on statistical tests of randomness. Therefore it will be presented the statistical foundations which justify this approach and the main tests known. All this work follows the guidelines of the National Institute of Standards and Technology statistics and uses libraries developed in C language. The prototype generated from a cipher text, returns information that will allow the cryptographer correct flaws in his cipher.*

Resumo. *Este trabalho apresenta a descrição 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. Todo o trabalho segue as normas do National Institute of Standards and Technology e utiliza bibliotecas estatísticas desenvolvidas em linguagem C. O protótipo gerado, a partir de um texto cifrado, retorna informações que permitirão o criptógrafo corrigir falhas em sua cifra.*

1. Introdução

Quando se fala em informação, de antemão associa-se a ela a 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 contramedidas para os ataques conhecidos.

A ferramenta mais utilizada nos dias atuais para salvaguardar informações, 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 à criptografia, que fazem uso de geradores de números aleatórios, podendo-se citar geradores de chaves. De fato, uma vez que as fontes verdadeiramente aleatórias se tornam muitas vezes inviáveis em uma aplicação, se admite (e se faz) uso de geradores de números pseudoaleatórios, os quais utilizam uma fonte não aleatória para gerar os números que tem características de números aleatórios.

Neste trabalho serão apresentados, por meio de uma revisão bibliográfica, geradores de números aleatórios e pseudoaleatórios utilizados em criptografia, bem como descrever as funcionalidades de um protótipo que poderá ser empregado por desenvolvedores de sistemas criptográficos para atestar o comportamento aleatório de suas cifras.

2. Fundamentação Teórica

2.1. 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. A produção dessas sequências pode ser feita por dois tipos básicos de geradores: Geradores de Números Aleatórios e os Geradores de Números Pseudoaleatórios.

2.1.1. Geradores de números aleatórios

Geradores de Números Aleatórios (RNG de *Random Number Generators*) usam uma fonte não determinística (ou seja, a fonte de entropia, como por exemplo ruídos em um circuito elétrico), juntamente com algumas funções de processamento (ou seja, o processo de destilação da entropia) para produzir aleatoriedade. 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 and Sutherland 1995].

A melhor maneira de se obter números verdadeiramente aleatórios é realizando 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 geradores de números aleatórios usando fontes de entropia apropriada. Dentre estes, pode-se citar o artigo de [Fairfield et al. 1985] que mostra como se gera um fluxo de bits aleatórios baseados na instabilidade da frequência de um oscilador.

É evidente que o estudo e construção de tais geradores requer um embasamento teórico/prático que foge ao escopo proposto neste trabalho. O que deve ficar evidente 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 tais *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 que medem tempo usado para pressionar as teclas ao se inserir a senha de um usuário [Plumb 1994, Zimmermann 1995]. Estas abordagens mostram ser inseguras se forem utilizadas em separado, contudo sua utilização se faz presente em associações criptográficas envolvendo Geradores de Números Pseudoaleatórios (PRNGs), conforme seção 2.1.2.

2.1.2. Geradores de números pseudoaleatórios

Geradores de Números Pseudoaleatórios *(PRNG de Pseudo-Random Numbers Genarator)* usam uma ou mais entradas e geram múltiplos "pseudo" números, as entradas para PRNGs são chamadas de sementes. 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 determinísticas 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 produzida a partir do valor anterior através das transformações que parecem introduzir uma aleatoriedade adicional [Rukhin et al. 2010].

2.1.3. Considerações sobre RNG e PRNG

Ambos os geradores 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.

2.2. Testes Estatísticos

Testes Estatísticos podem ser aplicados a uma sequência de bits para analisar se ela se comporta de maneira aleatória.

Tais testes 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 es-

pecífico, contudo, a aprovação nestes testes não garante a eficácia do sistema criptográfico à criptoanálise, que é o processo usado para tentar recuperar chaves criptográficas ou textos claros associados a um sistema criptográfico.

O NIST desenvolveu um pacote com 15 testes estatísticos para testar a existência ou não de aleatoriedade em uma sequência binária produzida por *hardware* e *software* de criptografia baseada em RNGs ou PRNGs. Os testes com suas descrições são apresentados na tabela a seguir:

Nome do Teste	**Descrição**
Teste de frequência (Monobit)	O objetivo deste teste é determinar se o número de uns e zeros em uma sequência têm aproximadamente o mesmo que seria esperado para uma sequência verdadeiramente aleatória.
Teste de frequência dentro de blocos	O propósito deste teste é determinar se a freqüência de m-bit blocos em uma sequência aparece tão frequentemente quanto seria de se esperar para uma sequência verdadeiramante aleatória.
Teste de corrida	O objetivo deste teste é determinar se o número de corridas de uns e zeros de vários comprimentos é o esperado para uma sequência aleatória. Em particular, este teste determina se a oscilação entre substrings é muito rápida ou muito lenta.
Teste para a mais longa corrida de 1's em um bloco	O objetivo deste teste é determinar se a distribuição de longas corridas de 1's coincide com as probabilidades teóricas.
Teste do posto para matrizes binárias	O objetivo deste teste é determinar se a distribuição da classificação de matrizes de 32x32 bits coincide com as probabilidades teóricas.
Teste da transformação discreta de Fourier (Espectral)	O propósito deste teste é determinar se a frequência espectral da sequência binária coincide com o que seria esperado para uma sequência verdadeiramente aleatória.
Teste de não sobreposiçao de padrão	O objetivo deste teste é determinar se o número de ocorrências de um modelo não periódico específico coincide com o número esperado para uma sequência verdadeiramente aleatória.
Teste de sobreposição de padrão	O objetivo deste teste é determinar se o número de ocorrências para todo modelo de 1's coincide com o que se espera para uma sequência verdadeiramente aleatória.
Teste estatístico Universal de Maurer	O objetivo deste teste é determinar se uma sequência binária não comprime além do que é esperado de uma sequência verdadeiramente aleatória.
Teste de complexidade linear	O objetivo deste teste é determinar se a sequência é ou não complexa o suficiente para ser considerada verdadeiramente aleatória.
Teste serial	O objetivo deste teste é determinar se o número de ocorrências de padrões sobrepostos de tamanho 2^m m-bits é aproximadamente o mesmo que seria esperado para uma sequência verdadeiramente aleatória.
Teste de entropia aproximada	O objetivo deste teste é comparar a frequência de sobreposição de blocos de comprimento 2 consecutivos / adjacentes (m e m+1) contra o resultado esperado para uma seqüência normalmente distribuídos. Em resumo, ele determina se uma seqüência aparece mais regular do que se espera de uma sequência verdadeiramente aleatória.

Teste das somas cumulativas (Cusums)	O objetivo deste teste é determinar se o máximo de somas acumuladas em uma sequência é muito grande ou muito pequena; indicativo de muitos 1's ou 0's no início das primeiras (últimas) etapas.
Teste de excursões aleatórias	O objetivo deste teste é verificar o número de ciclos dentro de uma sequência e determinar se o número de visitas a um determinado estado, [-4, -1] e [1, 4], excede o esperado para um sequência verdadeiramente aleatória.
Teste variante de excursões aleatórias	O propósito deste teste é determinar se o número total de visitas aos estados, entre [-9, -1] e [1, 9] excede o esperado para uma sequência verdadeiramente aleatória.

2.3. Aplicações dos Testes no *Advanced Encryption Standard* (AES)

O Padrão de Criptografia Avançada *(AES-Advanced Encryption Standard)* foi publicado pelo NIST em 2001. O objetivo do AES era substituir o Padrão de Criptografia de Dados (DES - *Data Encryption Standard*) que tinha sido adotado pelo NIST em 1977. Uma das vantagens do AES é que este utiliza cifras de bloco simétricas de 128 bits com suporte para tamanhos de chaves de 192 bits e 256 bits enquanto o DES utiliza cifras de tamanho de 64 bits e chaves com o tamanho de 56 bits. Por motivos de segurança e eficiência, quanto maior o tamanho do bloco maior a confiabilidade do algoritmo.

Para a escolha do algoritmo, o NIST utilizou alguns critérios para avaliar os candidatos, dentre estes, o critério de aleatoriedade, no qual os testes estatísticos foram empregados para verificar se esses algoritmos demonstram ou não um desvio na aleatoriedade, o qual poderia comprometer a segurança do mesmo.

3. Descrição do aplicativo

Para a construção utilizou-se a linguagem C, por ter a característica de fácil manipulação de bits com ponteiros.

A aprovação de uma amostra em algum teste envolve o cálculo de um valor de probabilidade (p-valor). Tal p-valor é definido como a probabilidade de se obter uma estatística de teste igual ou mais extrema quanto aquela observada em uma amostra, assumindo verdadeira a hipótese nula. Fixado um valor de significância α, um teste falha se p-valor $< \alpha$. Geralmente, α é escolhido no intervalo [0,001, 0,01]. No sistema foi escolhido α = 0,01.

Para concluir se a sequência é ou não aleatória para algum teste, é necessário calcular o p-valor que é definido a partir da função erro complementar (eq. 1) ou da função gamma incompleta (eq. 2).

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

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

onde

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

A função erro complementar $(erfc)$ utilizada é fornecida pela biblioteca math.h e a função $igmac$ é fornecida na biblioteca Cephes, ambas utilizadas na linguagem C [Goldberg and Wagner 1996]. O código 1 apresenta a função erro complementar.

A descrição detalhada destes e dos demais testes pode ser observado em [Rukhin et al. 2010].

Código 1 Código da função erro complementar

```
#include <stdio.h>
#include <math.h>
#include "../include/cephes.h"
double
cephes_erf(double x){
        static const double two_sqrtpi = 1.128379167095512574;
        double  sum = x, term = x, xsqr = x * x;
        int             j = 1;
        if ( fabs(x) > 2.2 )
                return 1.0 - cephes_erfc(x);
        do {
                term *= xsqr/j; sum -= term/(2*j+1); j++;
                term *= xsqr/j; sum += term/(2*j+1); j++;
        } while ( fabs(term)/sum > rel_error );
        return two_sqrtpi*sum; }

double
cephes_erfc(double x){
        static const double one_sqrtpi = 0.564189583547756287;
        double  a = 1, b = x, c = x, d = x*x + 0.5;
        double  q1, q2 = b/d, n = 1.0, t;
        if ( fabs(x) < 2.2 )
                return 1.0 - cephes_erf(x);
        if ( x < 0 )
                return 2.0 - cephes_erfc(-x);
        do {
                t = a*n + b*x; a = b; b = t; t = c*n + d*x;
                c = d; d = t; n += 0.5; q1 = q2; q2 = b/d;
        } while ( fabs(q1-q2)/q2 > rel_error );
        return one_sqrtpi*exp(-x*x)*q2; }
```

4. Resultados de uma aplicação

A aplicação proposta tem o intuito de analisar, como um passo inicial, se um algoritmo criptográfico está se comportando de maneira aleatória ou não. Para tal, é analisado um arquivo criptografado pelo AES através da aplicação dos testes no arquivo que retorna algumas informações (por exemplo, os p-valores) das quais a análise pode ser realizada.

Cada teste tem um p-valor, que é calculado de forma diferente. Isso se dá pelo fato de que cada teste tem uma finalidade diferente e analisa a sequência de bits de uma

maneira peculiar. Todos os testes tem apenas um p-valor, exceto os testes serial (2 p-valores), somas cumulativas (2 p-valores), excursão aleatórias (8 p-valores), variante de excursões aleatória (18 p-valores) e o de não sobreposição de padrão (148 p-valores).

Para cada teste estatístico, seu respectivo p-valor é analisado, se o p-valor for maior que 0,01 então conclui-se que a sequência é aleatória, caso contrário, ela não o é. Para os testes nos quais existem mais de um p-valor, se todos o p-valores forem maior que 0,01 conclui-se que a sequência é aletória, se todos os p-valores forem menor que 0,01 conclui-se que a sequência não é aleatória e caso haja alguma contradição, como por exemplo, o resultado de um dos p-valores for menor que 0,01 e outro for maior que 0,01 então a sequência deverá ser analisada para determinar se esse comportamento é ou não típico do gerador [Rukhin et al. 2010].

A Tabela 2 é referente aos resultados obtidos através da aplicação de um arquivo (BBS.dat disponível online[1]) criptografado com a utilização do algoritmo AES[2] ao conjunto de testes estatísticos. O tamanho da sequência utilizada é de 10.000.000 bits. Para o teste de não sobreposição de padrão, teste de excursões aleatórias e teste da variante de excursões aleatórias foram utilizados apenas 1 dos possíveis 148, 8 e 18 p-valores, respectivamente.

Tabela 2. Resultado dos testes

Nome do Teste	**Valor do P-valor**
Teste de frequência (Monobit)	0.863913
Teste de frequência dentro de blocos	0.070482
Teste de corrida	0.048251
Teste para a mais longa corrida de 1's em um bloco	0.762485
Teste do posto para matrizes binárias	0.119786
Teste da transformação discreta de Fourier (Espectral)	0.071532
Teste de não sobreposiçao de padrão	0.151992
Teste de sobreposição de padrão	0.301858
Teste estatístico da universal de Maurer	0.583059
Teste de complexidade linear	0.576030
Teste serial	0.106250 0.173104
Teste de entropia aproximada	0.249482
Teste das somas cumulativas (Cusums)	0.608609 0.464215
Teste de excursões aleatórias	0.377836
Teste variante de excursões aleatórias	0.501979

Pode-se verificar que este arquivo submetido à criptografia AES produziu um arquivo criptografado que foi aprovado em todos os testes, exceto o teste de não sobreposição de padrão que teve 2 de seus 148 p-valores (p-valor = 0.001987 e p-valor = 0.001574) menores do que 0,01 e o teste de excursão aleatória que teve 1 dos 8 p-valores

[1] `http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html`

[2] Disponível em `ttp://novatec.com.br/downloads.pp`

(p-valor = 0.007738) menor que 0,01, então para esses dois testes específicos deve ser feita uma análise para verificar se esse comportamento é típico do AES. Os p-valores menores que 0,01 foram obtidos, mas não foram utilizados na tabela.

5. Conclusão

A utilização dos testes estatísticos determina se uma sequência é verdadeiramente aleatória, podendo determinar assim se um sistema criptográfico está imune a algum tipo de ataque. Com as informações supridas pelo aplicativo aqui apresentado será possível ao criptógrafo corrigir falhas em sua cifra antes de sua publicação.

Conclui-se portanto, que o algoritmo AES utilizado para criptografar o arquivo, é considerado aleatório em 13 dos 15 testes utilizados, e em apenas 2 deles deve-se fazer uma análise mais detalhada na sequência utilizada para verificar se esse comportamento é próprio do AES.

Referências

Davis, D., Ihaka, R., and 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.

Fairfield, R. C., Mortenson, R. L., and 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.

Goldberg, I. and Wagner, D. (1996). Randomness and the Netscape browser. *Dr. Dobbs Journal*.

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

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

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., and Iii, L. E. B. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications.

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

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

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