Revista Brasileira de Ensino de Física, vol. 38, nº 1, 1313 (2016)
www.scielo.br/rbef
DOI: http://dx.doi.org/10.1590/S1806-11173812125

Influência da lei de Zipf na escolha de senhas

Influence of Zipf's law on the password choices

Leonardo Carneiro de Araújo*[1], João Pedro Hallack Sansão[1], Hani Camille Yehia[2]

[1]Universidade Federal de São João del Rei, Campus Alto Paraopeba, São João del Rei, MG, Brasil
[2]Departamento de Engenharia Eletrônica, Universidade Federal de Minas Gerias, Belo Horizonte, MG, Brasil

Recebido em 5 de setembro de 2015. Aceito em 5 de dezembro de 2015

Este artigo apresenta uma análise sob a ótica de teoria da informação de algumas maneiras de criar senhas para dificultar um ataque por força bruta. A lei de Zipf é observada nas línguas naturais e, por conseguinte, a entropia é reduzida quando as utilizamos ao criar uma senha. Muitas das empresas utilizam a política de restringir o número de caracteres de uma senha. Além disso, queremos criar senhas que não sejam demasiadas longas, nem que sejam difíceis de se memorizar. Mostraremos que, dado este cenário, a melhor estratégia (relação de compromisso entre criar uma senha forte e uma senha de fácil memorização e utilização) é a utilização de acrônimos, senhas formadas pela combinação (aparentemente sem nexo) da primeira letra de palavras em uma frase. Com esta abordagem podemos aumentar em aproximadamente 80% a entropia por caractere.
Palavras-chave: senhas, lei de Zipf, segurança, entropia.

Under the perspective of information theory, the present work performs an analysis of some methods used to create passwords in order to harden the defense against brute force attacks. Zipf's law is ubiquitous in natural languages and therefore it implies an entropy reduction when any language is used to create a password. Many companies impose a length restriction on passwords. Also, we do not want to create long passwords (that would take longer time to type), nor we want passwords that are hard to remember. On those terms, the best approach to create a password (the best tradeoff between creating a strong and an easy to memorize and use password) is the acronym approach, selecting the first character of each word in a sentence and combining them to form a gibberish string. Using this approach we are able to increase in 80% the entropy per character.
Keywords: passwords, Zipf's law, security, entropy.

1. Introdução

A preocupação com segurança da informação armazenada ou transmitida é antiga. Muitas vezes criptografia é confundida com estenografia. Esta possui como objetivo esconder a mensagem que será enviada ou armazenada. Aquela possui como objetivo garantir que o significado da mensagem não seja desvendado, mesmo que a mensagem caia em mãos erradas. Criptografia é a ciência que preocupa-se com a comunicação segura e usualmente secreta. O nome criptografia é composto de duas palavras gregas: *kryptos*, significando 'escondido' e *graphein*, significando 'escrita' [1]. Criptografia é uma forma de transmitir uma mensagem de forma confidencial que será incompreensível para alguém que a intercepte sem conhecer previamente o segredo para desencriptá-la. Desta forma, a criptografia era utilizada usualmente por militares, espiões e diplomatas, para transmissão de mensagens secretas. Podemos dizer que a história da criptografia é quase tão antiga quanto a palavra escrita. Algumas escrituras egípcias feitas há 4 milênios apresentam um elemento fundamental utilizado pela criptografia: a substituição [2]. As primeiras formas de criptografia eram bem rudimentares, como por exemplo a cifra de César, que consistia na mera substituição dos caracteres realizando um deslocamento fixo nas letras do alfabeto.

*Endereço de correspondência: leolca@ufsj.edu.br.

Se de um lado temos agentes que desejam realizar uma comunicação secreta, de outro, temos aqueles que desejam interceptar estas mensagens e decodificá-las. Destarte, surge a criptoanálise, ciência que estuda os sistemas de informação em busca de descobrir os aspectos ocultos de um sistema criptográfico, visando quebrá-lo e obter assim acesso à mensagem, mesmo quando a chave é desconhecida. Até o início do Século XX, a criptoanálise baseava-se exclusivamente na análise de padrões linguísticos e lexicográficos. Posteriormente, matemática, estatística, simulações e cálculos computacionais tornaram-se importantes ferramentas na criptoanálise moderna. Os desenvolvimentos realizados em *Bletchley Park* durante a Segunda Guerra Mundial são notórios, em especial o desenvolvimento das máquinas computacionais *Bombes* e *Colossus* importantes para quebrar os segredos da máquina electro-mecânica de criptografia alemã, intitulada Enigma.

Recentemente, a utilização de criptografia entrou na pauta da mídia jornalística após as revelações de Edward Snowden sobre a abrangência das informações coletadas pela Agência Nacional de Segurança Americana (em inglês: *National Security Agency* - NSA), configurando uma situação de invasão de privacidade de milhões de cidadãos e levantando questionamentos sobre os reais motivos para esta coleta, aparentemente indiscriminada, de informações. Como uma resposta, algumas empresas de informática, como Apple, Facebook, Google e Microsoft, passaram a disponibilizar, de forma mais acessível, criptografia de dados no armazenamento e transmissão. Estes sistemas adotados são categorizados como sistema de criptografia civil e baseiam-se no princípio de Kerckhoffs (veja secção 3.2), ou seja, a segurança deve depender inteiramente da senha secreta e, desta forma, o tamanho do espaço de chaves[1] é importante para garantir entropia ou aleatoriedade suficiente para manter o sistema seguro. Vários estudos mostram que o usuário é a ligação fraca, pois eles costumam escolher senhas muito simples [3–5]. Desta forma, é extremamente importante que o usuário saiba criar uma senha robusta o suficiente para dificultar o sucesso de ataques que buscam quebrá-la.

[1]Espaço de chaves é o conjunto formado por todas as possíveis chaves. O tamanho deste conjunto é determinado pelo comprimento máximo das chaves e pelo tamanho do alfabeto utilizado. Se C é o espaço de chaves, $\mathcal{X}$ é o alfabeto e n o comprimento máximo das chaves, então $|C| = |\mathcal{X}|^n$.

Uma forma de quebrar uma senha é realizar uma busca exaustiva dentre todas as possibilidades.

Algorithm 1 Busca exaustiva

```
1: procedure BUSCA-EXAUSTIVA (C)
2:     for ∀_c ∈ C do
3:         if eh_correto(c) then return c
return null
```

Esta busca, ilustrada no Algoritmo 1, garantidamente encontrará a senha e, para tanto, levará um tempo de no máximo $k|C|$, onde k é o tempo necessário para o algoritmo realizar um único teste para verificar se uma determinada senha c é correta, $|C|$ é a cardinalidade do conjunto de escolhas. Podemos supor que o agente externo de um ataque não possui maneiras para alterar k, pois k é determinado pela arquitetura do sistema de criptografia, porém o usuário, inconscientemente, pode ter restringido o conjunto C. Sabendo disso, o agente, ao realizar um ataque, pode realizar a busca em apenas um subconjunto de C, no qual, é mais provável, embora não garantido, encontrar a senha. Este tipo de busca é conhecido como ataque baseado em dicionários. Estamos aqui supondo que podemos testar se c é a senha quantas vezes desejarmos, o que não é verdade para ataques *online*, uma vez que os administradores geralmente adotam políticas para bloquear novas tentativas ou tornar k cada vez maior à medida que tentativas são realizadas. Se, de alguma forma, obtivéssemos a base de dados com as senhas e a função de encriptação, seria possível realizar um ataque *offline* e, desta forma, o limite imposto por k será apenas o custo computacional de verificar uma senha.

Em busca de evitar ataques baseados em dicionários (muitos deles constituídos a partir de listas de senhas de serviços virtuais que acabaram tornando-se públicas após ataques maliciosos), os administradores de sistemas muitas vezes tentam impor certas limitações às senhas, exigindo que elas satisfaçam algumas restrições como, por exemplo, número mínimo de caracteres, presença de caracteres não alfanuméricos, utilização de algarismos numéricos, ou ainda exigindo que o usuário mude sua senha com frequência. Entretanto, usuários buscam escolher senhas que sejam de fácil memorização, tarefa esta que fica mais difícil à medida que o comprimento da senha cresce e à medida em que a senha torna-se mais imprevisível. Em entrevista ao programa do apresentador John Oliver [6],

 DOI: http://dx.doi.org/10.1590/S1806-11173812125

Edward Snowden explica que ao invés de utilizar combinações de palavras, números e anagramas, é melhor mudar o paradigma e passar a utilizar *passphrases*, ou seja, criar uma frase para ser utilizada como senha. Nesta entrevista ele oferece como exemplo a *passphrase* 'MargaretThatcheris110%SEXY', ou seja, uma sequência de palavras constituindo uma frase. Por outro lado, não queremos que a senha seja constituída de muitos caracteres, pois isto implicará em mais tempo gasto para digitá-la sempre que for utilizada. Além do mais, muitos serviços impõem um limite máximo sobre o número de caracteres para uma senha.

Neste artigo iremos analisar, sob o ponto de vista de teoria da informação, quatro diferentes abordagens para criar senhas:

1. utilização de palavras;
2. utilização de *passphrases*;
3. método *Diceware* [7];
4. utilização de acrônimos, sequências de caracteres formadas pela primeira letra de cada palavra em uma frase.

Argumentaremos que, quando imposta uma relação de compromisso entre segurança e facilidade de memorização e uso, a melhor estratégia será utilizar acrônimos, pois, dentre as estratégias analisadas, é aquela que fornece maior entropia por caractere.

2. Utilização de senhas

Para grande parte dos usuários, e até mesmo especialistas na área de segurança digital, a utilização de senhas sempre foi um fardo e, por conseguinte, sua importância muitas vezes é dirimida, possibilitando assim a instauração de brechas de segurança. Buscando resolver este problema, novas tecnologias vêem sendo desenvolvidas com a finalidade de sobrepujar as convencionais senhas [8–10]. Alguns exemplos são as chaves eletrônicas, métodos biométricos, métodos de autenticação por dois fatores, dentre outros. Entretanto, a utilização de senhas convencionais ainda é predominante, barata, simples, ubíqua e está sempre presente com o usuário, não necessitando que este carregue algum outro objeto [8].

Enquanto as senhas ainda perduram em nosso cotidiano, algumas empresas desenvolveram gerenciadores de senhas, tais como KeyPass, LastPass, iCloud Keychain, dentre outros. Entretanto, estes gerenciadores armazenam uma base de dados com a relação de senhas e contas associadas. As senhas são armazenadas em sua forma textual, e não um *hash*[2] para elas, pois elas deverão ser posteriormente utilizadas para realizar a autenticação em algum outro serviço. A base de dados é armazenada na forma criptografada, utilizando para tanto uma senha mestre. Desta forma, os gerenciadores de senhas também estão sujeitos a ataques, com o agravante de que um ataque bem sucedido é capaz de revelar diversas senhas de um determinado usuário.

A história recente é repleta de incidentes de vazamento de informações de usuários [11], muitos dos quais através da apropriação de senhas de usuários, como por exemplo o vazamento de fotos de celebridades ocorrido em Agosto de 2014 [12, 13]. Além deste, muitos outros casos foram amplamente divulgados na mídia, como por exemplo: AOL (2006), RockYou! (2009), Sony (2011), Yahoo (2012), Adobe Systems (2013), Sony (2014), Ashley Madison (2015) [11]. A análise de senhas mostra que os usuários utilizam estratégias fracas para criar suas senhas. As senhas mais comuns geralmente são rudimentares como '123456', '12345', 'password', ou 'abc123', evidenciando que os usuários escolhem senhas preocupados com a sua facilidade de memorização. Um estudo empírico mostra que 80% das senhas utilizam apenas os 26 caracteres do alfabeto, 65,7% dos usuários utilizam senhas que tenham entre 4 e 6 caracteres, 14% utilizam senhas com 7 caracteres e 13% utilizam senhas com 8 caracteres [14]. Estas análises ilustram que a abordagem utilizada para a criação de senhas é equivocada. Existem formas mais seguras e intuitivas de criar senhas, uma delas foi ilustrada, de forma bem humorada, nos quadrinhos de xkcd [15].

Uma forma comum de mensurar a robustez de uma senha é através do cálculo da entropia por caractere. A medida de entropia surgiu com a teoria da informação formulada por Claude Shannon em 1948 [16]. Shannon explica que a entropia é uma medida estatística de quanta informação é produzida na média por símbolo de uma fonte como, por exemplo, letras de texto em uma determinada

[2]Uma função *hash* é uma função determinística que realiza o mapeamento de um conjunto de vários (ou até mesmo infinitos) membros em valores de um conjunto com um número fixo de membros. Uma função *hash* possui a propriedade de não ser reversível, fácil de ser calculada e, além disso, é difícil modificar um membro do domínio sem gerar um valor de *hash* distinto para ele, é ainda improvável encontrar dois membros do domínio que possuam o mesmo mapeamento pela função *hash*. São exemplo conhecidos de função *hash*: MD5, SHA1 e SHA2.

língua. Claramente, o tamanho do alfabeto é fator preponderante na estimativa da entropia. Além disso, a probabilidade de ocorrência dos símbolos determina a entropia associada à fonte. Isto implica que uma senha mais robusta é aquela produzida de forma completamente aleatória. Se, de alguma forma, obtivermos conhecimento sobre a frequência de ocorrência de símbolos utilizados para gerar senhas, a tarefa de determinação destas senhas será então facilitada.

3. Teoria da informação

A teoria da informação fornece uma maneira de quantificar informação, estuda a compressão de dados, transmissão através de um canal ruidoso e correção de erros. Claude Shannon é considerado o pai da teoria da informação e da criptografia moderna. O advento dos computadores nos forneceu uma maneira de manipular e processar informação de forma eficiente, enquanto Shannon nos forneceu uma maneira de interpretar e entender informação [16]. Shannon evidencia as relações entre o estudo de sua teoria de comunicação, abarcando correção de erros e teoria da informação, e criptografia [17]. No problema de comunicação, temos o receptor que, ao receber uma mensagem através de um canal ruidoso, busca decodificar a mensagem recebida e recuperar a informação original. Este problema é similar à criptografia, em que a chave é utilizada para ofuscar a informação de forma que ainda seja possível recuperá-la sem perda.

3.1. Entropia

Shannon quantificou e explicou o significado de informação. A informação associada a um evento específico, ou a uma realização de uma variável aleatória, é dada pelo logaritmo de sua probabilidade. Ele definiu então informação como o valor esperado da informação por evento, que é expressa através da equação

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x), \qquad (1)$$

onde H, chamado entropia, é a medida de informação, ou incerteza, associada à variável aleatória X. H é uma função exclusivamente das probabilidades $p(x)$ dos eventos x no alfabeto $\mathcal{X}$. Usualmente, utiliza-se o logaritmo na base 2 e desta forma a informação é medida em bits. Adiante, quando utilizarmos log dentro do contexto de teoria da informação, estaremos sempre utilizando a base 2 e consequentemente teremos as medidas em bits.

Golomb explica que informação é quantificada pela "quantidade de informação adquirida (ou entropia removida) ao aprender a resposta sobre uma pergunta cujas duas respostas possíveis são igualmente prováveis" [18].

3.2. Análise de frequência de ocorrência

Podemos definir a criptoanálise como a arte e ciência de resolver códigos e cifras. Em geral o problema envolve inicialmente descobrir qual é a linguagem utilizada na comunicação, o tipo genérico de codificação ou cifra adotado, as chaves específicas e finalmente a reconstrução da mensagem enviada. Para realizar esta análise e determinar estes parâmetros, devemos nos basear em uma grande quantidade de texto cifrado e informações relacionadas ao texto, como por exemplo, informações sobre o autor e destinatário, algum conhecimento específico sobre o conteúdo da mensagem, etc.

Vamos aqui assumir o princípio de Kerckhoffs, ou seja, um sistema de criptografia deve ser seguro mesmo que tudo sobre o sistema seja conhecido, exceto a chave. Auguste Kerckhoffs [19] estabeleceu seis princípios para criptografia:

1. o sistema deve ser praticamente indecifrável;
2. não deve ser necessário sigilo sob a forma como é realizada a criptografia, de forma que não seja um problema se o método cair nas mãos do inimigo;
3. é desejável que seja possível transmitir e lembrar a chave sem a necessidade de notas textuais, sendo também possível modificá-la se necessário ou desejável;
4. deve ser apropriado para a comunicação telegráfica;
5. deve ser portável e não deve requerer várias pessoas para operacionalizar;
6. o sistema deve ser de fácil manuseio, não necessitando grande esforço dos usuários.

Alguns destes não são mais relevantes, devido à capacidade computacional de realizar criptografia complexa disponível nos tempos atuais, entretanto, o segundo axioma ainda é crítico e portanto conhecido como o princípio de Kerckhoffs. Este mesmo princípio é formulado por Shannon em sua máxima:

 DOI: http://dx.doi.org/10.1590/S1806-11173812125

"o inimigo conhece o sistema" [17], ou seja, um sistema de criptografia deve ser seguro mesmo que tudo sobre o sistema, exceto a chave, seja público. Este paradigma de criptografia se opõe claramente à "segurança através da obscuridade".

Assumindo então que o método de criptografia é público, o ataque de força bruta consiste em procurar, através de uma busca exaustiva no espaço de chaves, aquela utilizada em um caso particular. Se esta for determinada com sucesso, o invasor terá então a capacidade de desvendar todas as futuras mensagens até que a chave seja alterada. O ataque força bruta é um método passivo usualmente feito *offline*. Ele consiste em uma busca exaustiva pela senha, o que é a alternativa adotada quando nenhuma outra informação existe sobre as fragilidades de um sistema de criptografia, ou nenhuma outra informação sobre o remetente, destinatário ou a mensagem em si. No pior dos casos, o ataque força bruta irá percorrer todo espaço de busca possível. Uma chave com comprimento de N bits poderá ser quebrada, em no máximo 2^N tentativas, e na média, metade do número de tentativas será suficiente.

O princípio de Landauer estabelece que existe uma quantidade mínima de energia necessária para apagar um bit de informação, ou seja, este é um processo dissipativo. Esta quantidade mínima, segundo Landauer [20] está relacionada com a temperatura e a constante de Boltzmann, sendo dada por $kT\ln 2$, onde k é a constante de Boltzmann (aproximadamente $1,38\times10^{-23}$ J/K) e T é a temperatura do circuito em Kelvin. Apesar de sua importância para a teoria da informação e ciência da computação, o princípio não havia sido demonstrado até poucos anos atrás, quando Eric Lutz e seus colegas mostram experimentalmente a existência do limite de Landauer em um modelo genérico de uma memória de um bit, tomado como hipótese por Landauer [21]. Este modelo consiste em uma única partícula que poderia estar em um dos dois poços de potencial, sendo que em um deles representaria o estado '0' e o outro o estado '1'. O bit será apagado ao forçar a partícula a entrar no estado '1'. Considerando que a tarefa será realizada à temperatura ambiente de 20 ℃ (293,15 K), o limite de Landauer será aproximadamente 0,0172 elétron-volt (eV) ou 2,75 zeptojoules (zJ). Considerando uma chave de 128 bits, serão necessários 2^{128} - 1 *flips* de bit para gerar todas as possíveis combinações de chaves. Vamos desconsiderar os recursos necessários para verificar a validade da chave gerada. Serão então necessários $10^{128\log_{10}2}\times2,75\times10^{-21}=9,36\times10^{17}$J = 260TWh, ou seja, toda energia gerada por 2,82 Usinas de Itaipu durante um ano inteiro (considerando a média dos últimos 10 anos), para gerar todas as possíveis chaves de 128 bits. Se considerarmos ainda o gasto de energia para verificar cada uma dessas chaves, seria necessário um consumo ainda muito maior.

Se, de alguma forma (consciente ou não), o usuário delibera por um espaço de chaves restrito, a tarefa de quebrar a criptografia utilizando força bruta será facilitada. Em criptoanálise, utiliza-se a ideia de que as línguas naturais apresentam padrões determinados e redundâncias que podem ser utilizados para ajudar a quebrar uma criptografia. A redundância em uma língua é uma questão também de interesse de linguistas que buscam modelos matemáticos para descrever as línguas naturais. Apesar da aparente diversidade encontrada entre diversas línguas, de diferentes famílias linguísticas, o ordenamento dos constituintes[3] utilizados em uma língua possui uma entropia relativa associada que apresenta-se como uma característica universal das linguagens naturais [22].

George Kingsley Zipf observou uma relação de potência entre o ranqueamento e a frequência de ocorrência de palavras em textos [23],

$$f_k \propto k^{-s}, \tag{2}$$

onde k é a posição que a palavra ocupa em uma lista ordenada (ranqueamento, posição ou ordem; em inglês *rank*), f_k é a frequência de ocorrência da k-ésima palavra e s é uma constante que caracteriza a distribuição. A Fig. 1 apresenta a relação observada entre a ordem de uma palavra e a sua frequência de ocorrência. Foi utilizado o *Open American National Corpus* (veja a secção 5).

Para gerar o gráfico ilustrado na Fig. 1, primeiramente selecionamos os arquivos de texto contidos no corpus que representam dados da língua escrita. Podemos então contabilizar a frequência de ocorrência de cada palavra, ordená-las segundo a frequência de ocorrência e, por fim, realizar o gráfico do *rank* (ordem) versus a frequência de ocorrência. Notamos que, ao utilizar uma escala logarítmica no eixos das abscissas e ordenadas, a relação torna-se aproximadamente linear.

[3] A ordem dos constituintes de uma sentença é um critério tipológico para a classificação das línguas de acordo com sua sintaxe.

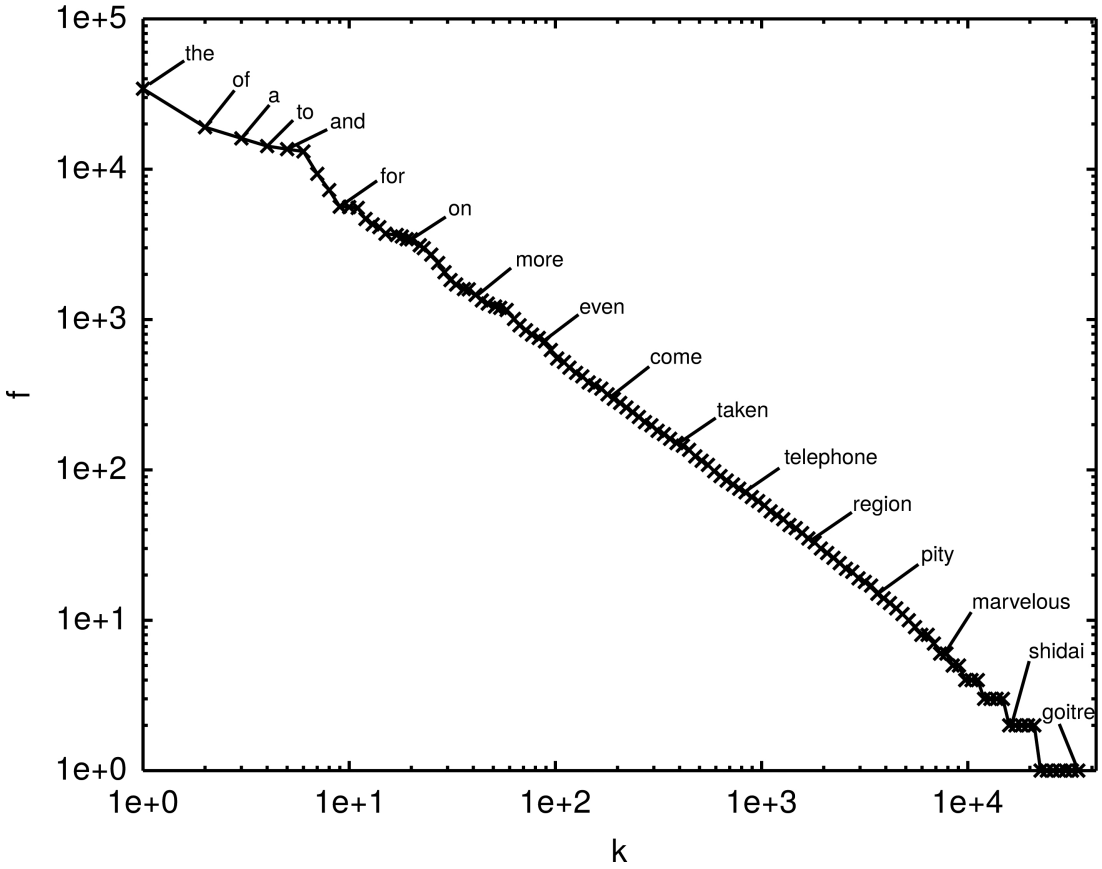

Figura 1: Lei de Zipf no Inglês escrito (dados do OANC). *Rank* (k) versus frequência de ocorrência (f).

A lei de Zipf é uma lei enigmática, controversa e amplamente observada na natureza, inclusive nas linguagens humanas. Em alguns casos, temos grandezas em que a escala para sua medição varia em torno de um determinado valor, outras grandezas variam por uma enorme extensão, muitas vezes esta extensão cobre muitas ordens de grandeza. Um exemplo típico para o primeiro caso é a altura de um ser humano adulto; já para o segundo caso, podemos tomar como exemplo o tamanho da população de cidades. Distribuições desta forma são ditas seguir uma lei de potência. Esta relação de potência é também observada em outros diferentes fenômenos, tais como: magnitude de terremotos [24]; população de cidades [25]; economia [26]; expressão gênica [27]; sistemas dinâmicos caóticos [28]; magnitude de avalanches [29]; tráfegos de dados na Internet [30]; número de citações de artigos científicos [31]; tiragem de livros e discos [32, 33]; e muitos outros. A lei de Zipf pode ser interpretada como uma reformulação do princípio do Pareto que afirma que, para muitos fenômenos, grande parte dos efeitos proveem de uma porção minoritária das possíveis causas. O nome deste princípio foi dado em homenagem ao economista italiano Vilfredo Pareto que, em 1906, observou que 80% das terras da Itália estavam nas mãos de apenas 20% da população. Quando nos referimos à lei de Zipf, a utilizamos num contexto em que será relacionada frequência de ocorrência de um evento com o seu *rank*, ou seja, podemos ver esta relação como uma função massa de probabilidade sobre o *rank*. A lei do Pareto, por outro lado, é dada em termos da função distribuição acumulada, quer dizer, o número de eventos maiores que um determinado valor é inversamente proporcional ao valor dado.

Uma distribuição de lei de potência também é dita invariante à escala, uma vez que esta é a única distribuição que possui o mesmo padrão em qualquer escala em que for observada [34]. Isto pode ser demonstrado seguindo os passos apresentados por Mark Newman [34]. Primeiramente, vamos supor que existe uma distribuição $p(k)$ que é invariante à mudança de escala, ou seja, a seguinte relação deverá ser satisfeita

$$p(bk) = g(b)p(k). \tag{3}$$

 DOI: http://dx.doi.org/10.1590/S1806-11173812125

Para qualquer mudança de escala, por um fator b, na variável de medição k, teremos como reflexo uma mudança de escala na distribuição dada por um fator multiplicativo $g(b)$. Esta relação deverá ser verdadeira para qualquer b e ter validade ao longo de todos os valores de k. Vamos inicialmente supor $k = 1$, o que nos fornece $p(b) = g(b)p(1)$, e assim, substituindo na Eq. (3) teremos

$$p(bk) = \frac{p(b)p(k)}{p(1)}. \tag{4}$$

Como esta equação deverá ser verdadeira para todo b, vamos diferenciá-la com relação à b, obtendo

$$kp'(bk) = \frac{p'(b)p(k)}{p(1)}, \tag{5}$$

onde p' é a derivada de p com relação ao seu argumento. Escolhendo agora $b = 1$, teremos

$$k\frac{dp}{dk} = \frac{p'(1)}{p(1)}p(k). \tag{6}$$

Esta equação diferencial de primeira ordem possui como solução

$$\ln p(k) = \frac{p(1)}{p'(1)} \ln x + \text{constante}. \tag{7}$$

Fazendo $k = 1$, obtemos o valor da constante como sendo $\ln p(1)$. Podemos fazer a exponencial de ambos os lados da nossa solução e obteremos

$$p(k) = p(1)k^{-s}, \tag{8}$$

onde $s = -p(1)/p'(1)$. Obtivemos assim a lei de potência como sendo a distribuição invariante a uma mudança de escala.

A análise de senhas, que tornaram-se públicas após incidentes de segurança, mostra que estas também apresentam a mesma relação observada por Zipf [35, 36]. O expoente s da distribuição de Zipf é importante na caracterização da fonte. Ele pode ser utilizado, por exemplo, como um parâmetro importante para constituir a assinatura de um autor, evidenciada ao analisar quantitativamente seus textos [37]. As linguagens naturais apresentam um expoente característico $s \approx 1$ [38]. Para alguns tipos de comunicação humana o valor do expoente é maior, por exemplo, um expoente $s \approx 1,66$ foi encontrado na fala das crianças; $s \approx 1,43$ em comunicação militar e $s < 1$ para formas avançadas de esquizofrenia [38]. Um expoente $0 < s < 1$ restringe o léxico a um tamanho finito, já um expoente $s > 1$ permite que léxico cresça sem limites [39]. Um expoente próximo de 1 é encontrado como resultado da minimização da função de energia, combinando o esforço do ouvinte e o esforço do falante [40]. A estimação da entropia é mais sensível a erros quando s está nas proximidades de 1, podendo ser severamente afetada pelo truncamento amostral [39].

Os primeiros experimentos para estimar a entropia da língua inglesa escrita foram realizados por Shannon [41] e posteriormente por Cover e King [42]. Estes experimentos buscavam averiguar se as pessoas eram capazes de predizer o próximo caractere em um texto. Através deste experimento, Shannon estimou um valor entre 0,6 e 1,3 bits por caractere para a entropia do inglês escrito [41], já Conver e King estimaram 1,25 bits por caractere [42]. Posteriormente, foram realizados experimentos com auxílio de recursos computacionais para comprimir textos em inglês, buscando uma compressão eficiente, que deve aproximar-se da real entropia da fonte. Teahan e Cleary utilizaram uma variação do algoritmo *Prediction by Partial Matching*, que utiliza como modelo a cadeia de Markov, atingindo 1,46 bits por caractere [43], valor este bem próximo dos valores de entropia reportados anteriormente [41, 42].

A Fig. 2 apresenta uma estimativa da probabilidade de ocorrência dos caracteres no Inglês. Analisando esta probabilidade ao longo de um texto, constatamos que elas não sofrem grandes alterações e portanto podemos considerar que a fonte em questão é ergódica estacionária, ou seja, qualquer amostra estatisticamente significante do processo representa as características estatísticas do processo (ergótico) e estas características não se alteram ao longo do tempo (estacionário). Para a analisar o comportamento da estimativa para a função massa de probabilidade ao longo do texto, selecionamos trechos com 1 milhão caracteres, com sobreposição de 995 mil caracteres, e estimamos as probabilidades ao longo da base de dados textual (veja a Fig. 3, na qual a barra de erro representa a incerteza sobre a medição, dada por um desvio padrão). Conhecer a distribuição da fonte é importante quando desejamos estabelecer um critério de busca mais eficiente do que a mera busca exaustiva por todas as possibilidades. Veremos adiante como a distribuição, ou melhor, a entropia afeta o tamanho do espaço de chaves.

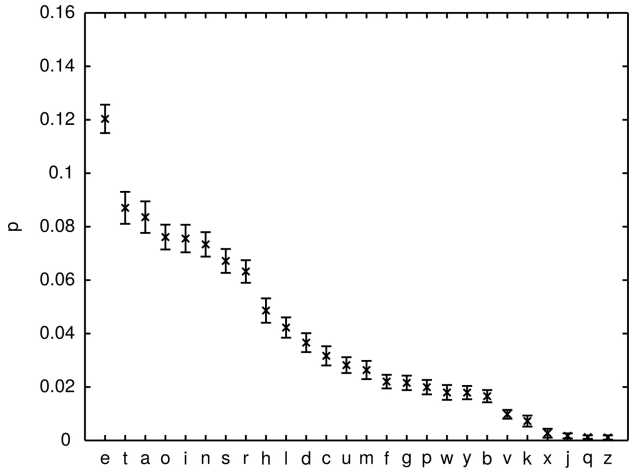

Figura 2: Probabilidade dos caracteres em Inglês (a-z) ao longo de um *corpus* (foram observadas sequências subsequentes de 1 milhão caracteres com sobreposição de 995 mil caracteres).

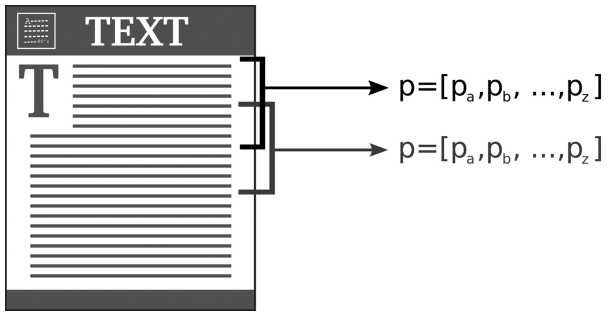

Figura 3: Obtenção da probabilidade dos caracteres ao longo de um texto.

3.3. Conjunto típico

Analisando sequências de símbolos sob a ótica de teoria da informação, utiliza-se o conceito de conjunto típico para designar o menor conjunto das sequências que correspondem a todas as sequências observáveis, ou seja, tudo aquilo que ocorre. Se expressarmos o conjunto típico das sequências de comprimento n por $A_\epsilon^{(n)}$, teremos que $\Pr(A_\epsilon^{(n)}) \approx 1$ [44].

Considerando então uma sequência de n variáveis aleatórias independentes e identicamente distribuídas, podemos afirmar que a probabilidade de observamos uma determinada realização para esta sequência será dada por $p(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} p(X_i = x_i)$. A informação de Hartley-Shannon associada a um evento x é dada por $I(x) = -\log p(x)$. Desta forma, a informação associada a uma sequência de símbolos $x_1, \ldots, x_n$ será dada por $I(x_1, \ldots, x_n) = \sum_{i=1}^{n} I(x_i)$. Teremos assim

$$\frac{1}{n}\sum_{i=1}^{n} I(X_i) \xrightarrow[n\to\infty]{p} H(X), \tag{9}$$

e desta forma poderemos mostrar que $p(x_1, \ldots, x_n) \approx 2^{-nH}$. As sequências típicas terão aproximadamente a mesma probabilidade e, portanto, existem 2^{nH} sequências deste tipo [44].

Supondo, por exemplo, que tenhamos um ensaio de Bernoulli[4] com $p = 0{,}9$. Em uma sequência binária de comprimento 100, esperamos observar 90 vezes o número 1. Uma sequência típica será aquela em que o número de observações do número 1 for próximo de 90. Para o valor de p dado, teremos que $H(p) = 0{,}469$ bits. Teremos então que o número de sequências típicas será aproximadamente $2^{nH(p)} = 2^{100\times 0{,}469} = 2^{46.9} \approx 1{,}31 \times 10^{14}$. Este número é, entretanto, bem menor do que o número de possíveis sequências de comprimento 100, $2^{100} \approx 10^{30{,}1}$, ou seja, usualmente teremos $2^{nH} \ll 2^{n\log|\mathcal{X}|}$. A igualdade só ocorrerá quando a entropia for máxima, ou seja, quando a distribuição da fonte for uniforme. Podemos verificar que a razão entre o tamanho do conjunto típico e o tamanho do conjunto de todas sequências é exponencialmente decrescente com n, ou seja, $2^{n(H-\log|\mathcal{X}|)}$, teremos então um decrescimento mais acentuado quanto mais distante a entropia for do seu valor máximo.

A probabilidade de encontrarmos, em um ensaio de Bernoulli com n realizações $(x_1, \ldots, x_n)$ e com probabilidade p para 1 (um) e probabilidade (1 - p) para 0 (zero), uma sequência em que existam k ocorrências de 1 (um) será dada por

$$p(S_n = k) = \binom{n}{k} p^k (1-p)^{n-k}\ , \tag{10}$$

onde $S_n = x_1 + \ldots + x_n$, ou seja, a soma das n realizações do ensaio de Bernoulli. A Fig. 4 apresenta esta probabilidade para sequências de comprimentos diferentes. Podemos verificar neste exemplo que, à medida que a sequência cresce, o conjunto das sequências, que efetivamente possui uma probabilidade significativa, torna-se proporcionalmente menor.

Note que a entropia é fator preponderante na determinação do tamanho do conjunto típico, que é

[4]Ensaio de Bernoulli é um experimento aleatório, no qual apenas dois resultados são possíveis: verdadeiro (1) ou falso (0).

 DOI: http://dx.doi.org/10.1590/S1806-11173812125

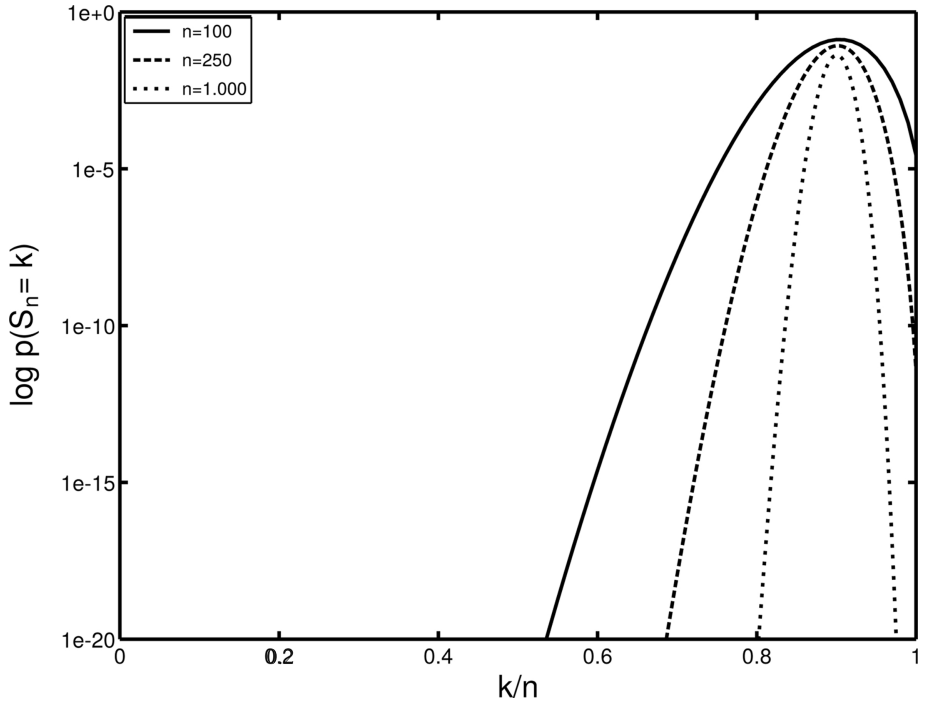

Figura 4: O gráfico ilustra a probabilidade de se obter uma sequência com k ocorrências do 1 em um ensaio de Bernoulli utilizando $p = 0{,}9$. É ilustrado os casos em que o número de realizações é $n = 100$, $n = 250$ e $n = 1000$.

tipicamente muito menor que o tamanho do conjunto de todas as sequências. Se estivermos utilizando uma fonte com menor entropia, logo também teremos um menor conjunto de sequências típicas. No caso da busca exaustiva por uma chave, devemos iniciá-la pelas sequências do conjunto típico. Sabendo disso, para proteger os dados contra um ataque por força bruta, devemos buscar aumentar a entropia, pois, apesar de não alterar o tamanho do conjunto de todas as sequências para um dado tamanho, estaremos tornando o conjunto típico tão maior quanto for a entropia. No limite da entropia máxima, não será possível resumir o espaço de busca pelo conjunto típico.

4. Diceware

O *Diceware* é um método para gerar senha à mão, utilizando apenas um dado de 6 lados e a lista de palavras disponibilizada pelo método. O método recomenda jogar o dado 5 vezes para cada palavra que será utilizada na senha, ou seja, os 5 lances de dados irão gerar um número que corresponderá ao índice de uma das 7.776 possíveis palavras no dicionário *Diceware*. O método recomenda a utilização de 5 palavras para formar a *passphrase*, o que levaria a uma entropia de 64,62 bits na senha final.

É importante ressaltar que o *Diceware* utiliza uma lista de palavras com comprimentos variando de 1 a 6 caracteres, conforme a Tabela 1. Desta forma, a senha final poderá ter de 5 a 30 caracteres e o comprimento esperado será de 21,16 caracteres.

A utilização do *Diceware* traz consigo algumas desvantagens:

1. estaremos utilizando um vocabulário restrito;
2. muitas palavras da lista são desconhecidas, o que torna difícil a memorização pelo usuário;
3. a lista contém números e caracteres não alfanuméricos, o que também não é uma preferência dos usuários;
4. o comprimento da senha pode ser utilizado como indício para facilitar a busca.

Tabela 1: Número de palavras com determinado comprimento no *Diceware*.

comprimento	número de palavras
1	51
2	784
3	853
4	2346
5	3111
6	631

Conhecer o comprimento da senha reduz a entropia de uma certa quantidade, variando de 2,67 a 36,12 bits no total, com valor esperado de 3,35 bits. Este vazamento de entropia corresponderia a uma perda de 0,12 a 7,22 bits por caractere, e uma perda esperada de 0,16 bits por caractere. O vazamento de entropia é definido como a diferença entre a entropia máxima ($\log_2$ do número de possíveis senhas geradas pelo método) e a entropia remanescente, dado que o comprimento da senha é conhecido ($\log_2$ do número de possíveis senhas com um dado comprimento geradas pelo método). A Fig. 5 ilustra como o número de senhas disponíveis no *Diceware* varia em função do comprimento final da senha (número de caracteres). Vamos representar o conjunto de palavras no *Diceware* por $\mathcal{W}$. Uma senha gerada pelo método *Diceware* é constituída por uma sequência de $n = 5$ palavras. Esta sequência será representada por $w_{1:n} = w_1, \ldots, w_n$, onde cada um dos $w_i \in \mathcal{W}$, para $i = 1, \ldots, n$. Iremos definir $N(w_{1:n})$ como o número existente de sequências de comprimento n. Definindo também $l(w_{1:n})$ como o comprimento (em caracteres) de uma determinada sequência, podemos escrever $N(w_{1:n} \mid l(w_{1:n}) = k)$, ou seja, o número de sequências formadas por n palavras, tais que o comprimento (em caracteres) da sequência é igual a um determinado valor k. O vazamento de entropia para sequências com um determinado comprimento (em caracteres) k será dado por

$$G_k = \log_2 \left(\frac{N(w_{1:n})}{N(w_{1:n} \mid l(w_{1:n}) = k)} \right). \qquad (11)$$

A Tabela 2 apresenta os valores de vazamento de entropia calculados para os diferentes comprimentos da senha final.

É importante ainda observar que, independente do método utilizado para criar uma senha, conhecer o comprimento de uma senha sempre resultará em um vazamento de entropia.

O método ainda propõe que sejam descartadas as senhas geradas com menos do que 14 caracteres, o que evitaria muita perda de entropia causada pelas *passphrases* curtas, restringindo pouco o espaço de senhas, uma vez que apenas 0,65% delas seriam eliminadas. Entretanto o método não propõe a exclusão das *passphrases* muito longas que também são responsáveis por grande parte do vazamento de entropia [3].

5. Base de dados

Para realização das análises aqui propostas é necessária a utilização de uma base de dados extensa. Dados textuais podem ser facilmente obtidos e manipulados; além disso, as unidades de processamento textuais são bem definidas em letras e palavras, o que não podemos afirmar quando desejamos analisar a língua enquanto fenômeno falado [45].

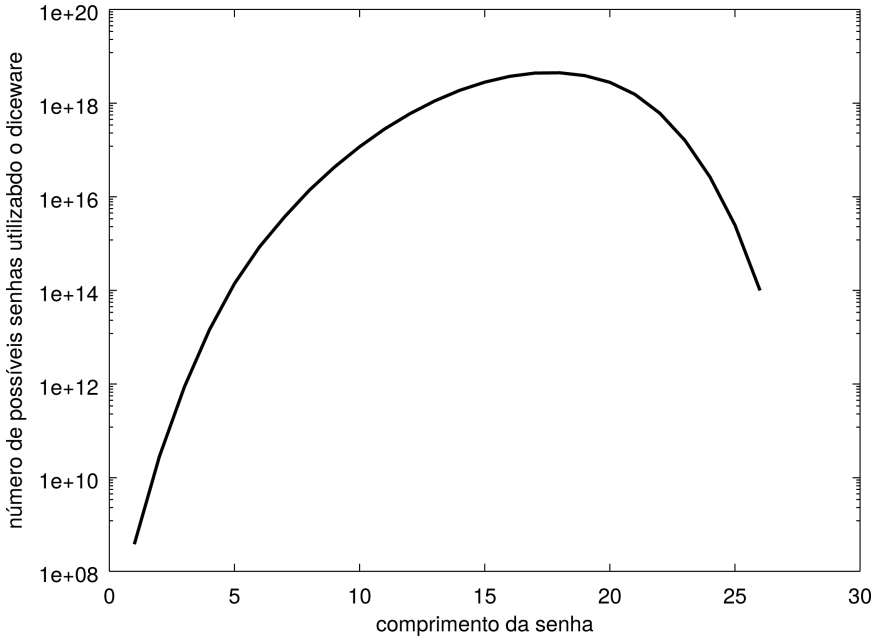

Figura 5: Número de senhas no *Diceware* para um determinado comprimento de senha.

 DOI: http://dx.doi.org/10.1590/S1806-11173812125

Tabela 2: Número de palavras com determinado comprimento no *Diceware*.

Comprimento	Vazamento de entropia (bits)
5	36,12
6	29,91
7	24,96
8	20,91
9	17,63
10	15,03
11	12,90
12	10,99
13	9,36
14	7,92
15	6,66
16	5,59
17	4,67
18	3,92
19	3,34
20	2,93
21	2,70
22	2,67
23	2,88
24	3,35
25	4,20
26	5,55
27	7,47
28	10,06
29	13,48
30	18,12

Neste trabalho utilizamos a base de dados *Open American National Corpus* (OANC)[5]. Esta trata-se de um corpus com anotações constituído por 15 milhões de palavras e disponíveis livremente no site da ANC para qualquer utilização. O corpus contém textos de diversos gêneros e transcrições de fala produzidas a partir de 1990. Os mesmos dados também estão disponíveis em outros formatos, como por exemplo um formato XML, contendo anotações, tais como marcação estrutural, limite de sentenças, classe gramatical e identificação de constituintes, dentre outras. Todas as anotação utilizam o padrão ISO/TC37 SC4.

As razões pelas quais utiliza-se corpora diversos para efetuar análises, assim como aqui realizamos, são:

1. facilidade na obtenção dos dados, uma vez que estes já foram reunidos e organizados;
2. os corpus são criados de forma a possuírem uma boa representação de uma determinada língua, pois são uma composição balanceada de textos de diferentes estilos;
3. a utilização de corpus permite a comparação de resultados no meio acadêmico;
4. muitos corpus possuem anotações lexicais e gramaticais;
5. podem ser utilizados para analisar as alterações temporais sofrida por uma língua.

Existem diversos corpora para a língua portuguesa. Ao leitor interessado, recomendamos que consultem a Línguateca[6], um centro de recursos distribuídos para a língua portuguesa.

6. Resultados

Nas Figs. 1 e 2 podemos observar que a distribuição dos símbolos (palavras ou letras) no inglês não é uniforme. Existe uma evidente preferência pela utilização de alguns tipos, em detrimento de outros. Este efeito é ainda mais acentuado quando analisamos a frequência de ocorrência de palavras, ilustrada na Figura 1. Esta característica é comum às linguagens naturais e faz com que a aleatoriedade em uma língua seja diminuída. Considerando a língua enquanto meio de comunicação, redundância é importante pois permite que erros inseridos durante a transmissão ou problemas de recepção sejam corrigidos, garantindo assim a eficiência da transmissão de informação. Por outro lado, do ponto de vista da criptografia, redundância é algo indesejável.

Uma análise linguística pode ser feita sob duas perspectivas: sincrônica ou diacrônica. A investigação sincrônica lida com a língua como um fenômeno fixo em um determinado instante, enquanto a abordagem diacrônica investiga a evolução da língua no tempo Esta distinção foi introduzida por Ferdinand de Saussure no início do Século XX [46]. Embora seja incontestável que línguas mudam ao longo do tempo, iremos considerar que, em uma análise sincrônica, suas características não se alteram drasticamente e podem ser encontradas em qualquer amostra representativa de uma língua, em um determinado instante [47]. A Fig. 2 apresenta a probabilidade de ocorrência dos caracteres no inglês escrito. Podemos verificar que a variabilidade destas probabilidades ao longo do *corpus* é baixa, iremos assim considerar as probabilidades constantes.

A Tabela 3 apresenta as seguintes estatísticas com relação ao inglês escrito e ao inglês falado:

[5] `http://www.anc.org/OANC`

[6] `http://www.linguateca.pt/`

Tabela 3: Comparação realizada entre o Inglês escrito e falado.

	$\lvert\mathcal{X}\rvert$	$\log\lvert\mathcal{X}\rvert$	H_p	L_p	$H_\mathrm{p}/L_\mathrm{p}$	H_c	H_pc
Ingl. Esc.	41903	15,36	10,61	4,69	2,26	4,19	4,14
Ingl. Fal.	27496	14,75	8,55	3,65	2,34	4,19	4,15

1. o tamanho do vocabulário $|\mathcal{X}|$ (número de palavras);
2. a entropia máxima $\log|\mathcal{X}|$ (considerando distribuição uniforme);
3. a entropia quando consideramos palavras como símbolos H_p;
4. o comprimento esperado das palavras L_p;
5. a entropia por caractere $H_\mathrm{p}/L_\mathrm{p}$ (quando consideramos as palavras como símbolos);
6. a entropia quando consideramos os caracteres como símbolos H_c;
7. e, por fim, a entropia do primeiro caractere das palavras H_pc.

Para calcular H_p devemos contabilizar o número de ocorrências de cada palavra no corpus, para então estimar suas probabilidades, necessárias para calcular a entropia. Procedemos de forma semelhante para calcular H_c e H_pc, ou seja, devemos contabilizar a frequência de ocorrência dos caracteres (ou apenas o primeiro caractere de cada palavra, no caso do cálculo de H_pc), e então estimar as probabilidades. Para calcular L_p devemos obter o comprimento das palavras e então computar $L_\mathrm{p} = \sum_{i=1}^{|\mathcal{X}|} p_i l_i$, onde p_i e l_i representam a probabilidade e o comprimento da i-ésima palavra.

A utilização de palavras ou *passphrases* fornecerá aproximadamente 2,3 bits por caractere de entropia. Para o alfabeto de 26 caracteres, o valor máximo será de 4,7 bits por caractere. Com a utilização do *Diceware* teremos $\log 7776 = 12,92$ bits por palavra, como o comprimento esperado das palavras no *Diceware* é 5,24 caracteres, teríamos então 2,47 bits por caractere de entropia, entretanto a distribuição dos caracteres no *Diceware* não é uniforme e, desta forma, a real entropia por caractere é 4,39 bits. Todavia, devemos lembrar que o *Diceware* utiliza um alfabeto com 54 símbolos. Para melhor compararmos os resultados, devemos normalizar para um alfabeto com 26 símbolos. Realizando esta normalização obteremos uma incerteza de 3,58 bits por caractere agregados por cada letra na sequência.

Este valor não difere muito do valor anterior obtido para a incerteza agregada por cada caractere acrescido em uma sequência. Como podemos verificar na Tabela 3, o valor da entropia por caractere H_c é praticamente o dobro dos valores calculados nos dois casos anteriores. A melhor estratégia seria sortear letras utilizando uma distribuição uniforme e assegurar assim o máximo de 4,7 bits por caractere. Uma outra estratégia seria sortear letras de um texto, desta forma as letras serão escolhidas conforme a sua distribuição na língua. Esta estratégia fornecerá 4,19 bits de entropia por caractere, valor bem mais próximo do máximo; entretanto, as sequências formadas serão de difícil memorização. Uma ultima estratégia, visando maximizar a entropia por caractere, sem criar grandes dificuldades em sua memorização, consiste em utilizar acrônimos. Memorizar uma frase com 10, 15 ou 20 palavras não é tão difícil quanto memorizar uma sequência de 10, 15 ou 20 letras aleatórias. A título de exemplo, podemos criar a frase: "Não posso afirmar que a Margaret Thatcher ou sequer a Dilma sejam sexys como certa vez sugeriu Edward Snowden para exemplificar uma senha difícil de ser quebrada", que nos geraria a seguinte senha "npaqamtosadssccvsespeusddsq". Esta ultima estratégia garante um acréscimo de 4,14 bits de incerteza por caractere adicionado à sequência, sendo então, sob esta análise, a melhor estratégia quando existe uma relação de compromisso entre maximização da entropia e facilidade de memorização da senha.

7. Conclusões

Para dificultar o sucesso de um ataque de força bruta é importante que a escolha da chave seja feita de forma a maximizar o espaço de chaves que deverá ser percorrido na busca exaustiva realizada pelo agressor. Muitas vezes os sistemas restringem o comprimento da chave, por exemplo, a Microsoft restringe a 16 caracteres, muitas lojas de comércio eletrônico também restringem drasticamente o número de caracteres de uma senha. Lojas virtuais como Submarino e Americanas.com utilizam o máximo de 8 caracteres, enquanto Netshoes utiliza 15 e, ainda, Mercado Livre, 20. Mais grave ainda são os bancos que, além de restringir o tamanho de sua senha, restringem também o alfabeto, aceitando apenas

 DOI: http://dx.doi.org/10.1590/S1806-11173812125

dígitos de 0 a 9. Algumas empresas fornecem um limite muito maior, como é o caso da Amazon (128 caracteres) e Google (100 caracteres).

Um método para criar senhas deve ser pautado por uma relação de compromisso entre:

1. usabilidade por um leigo;
2. segurança computacional.

Obedecendo então às usuais e severas restrições no comprimento de uma senha, e como não desejamos utilizar senhas muito longas que demandariam esforço e tempo para digitá-las, devemos buscar maximizar a entropia por caractere. Sob esta perspectiva, foram analisadas 4 diferentes estratégias para se criar senhas, observando a quantidade de entropia por caractere que cada uma delas fornece. Os resultados obtidos foram os seguintes:

1. a utilização de simples palavras fornecem aproximadamente 2,3 bits por caractere;
2. o uso de *passphrases* não apresenta nenhum ganho sobre a anterior, uma vez de que trata-se de uma simples concatenação de palavras, não alterando, desta forma, a incerteza associada por caractere agregado na sequência;
3. o método *Diceware* fornecerá 3,8 bits por caractere;
4. a utilização de acrônimos fornecerá 4,19 bits por caractere, o melhor resultado dentre aqueles analisado neste trabalho.

Os resultados mostram que a melhor relação de compromisso entre maximização da entropia por caractere e facilidade de memorização será obtida ao utilizarmos a última estratégia, estaremos assim a apenas 0,56 bits da entropia máxima. Poderemos melhorar ainda esta estratégia se utilizarmos também algarismos e caracteres não-alfanuméricos.

Ao empregarmos palavras de uma língua para constituir uma senha, estaremos pagando um alto preço que incorre da redundância existente nas línguas. A lei de Zipf é observada nas linguagens naturais e, como consequência, teremos uma drástica redução da entropia. Aplicamos o método proposto em [39] para estimar a entropia para uma distribuição zipfiana. A partir das estimativas para diferentes expoentes característicos e diferentes tamanhos de alfabeto, foi possível traçar o gráfico ilustrado na Fig. 6. Este evidencia o fator de redução da incerteza ocasionada pela distribuição de Zipf. Observamos que em todas as situações, a entropia

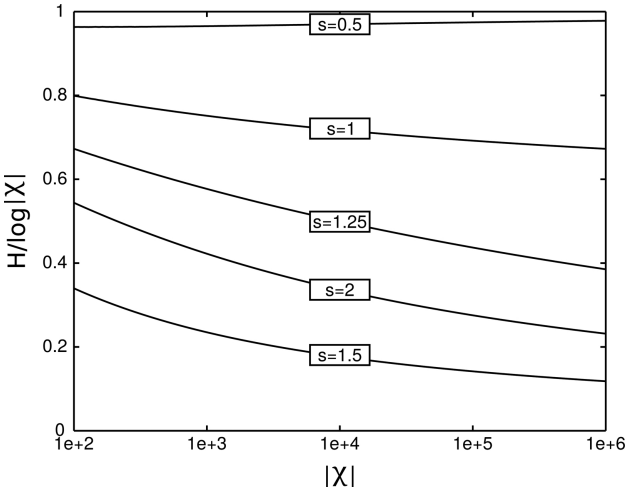

Figura 6: Fator de redução da entropia causado pela lei de Zipf.

é reduzida quando existe uma distribuição zipfiana. Para as distribuições apresentadas com $s \geq 1$, verifica-se que a redução é mais acentuada com o crescimento do tamanho do conjunto de símbolos $|\mathcal{X}|$ e com o crescimento do expoente s. As línguas naturais apresentam $s \approx 1$ e assim experienciamos uma diminuição de, no mínimo, 20% na entropia. Esta diminuição é grande o suficiente para assegurar que teremos um conjunto típico que será constituído por uma fração exponencialmente ínfima do todo, quando n for suficientemente grande.

Referências

[1] Dictionary.com Unabridged, Online Dictionary (2015). disponivel em `http://dictionary.reference.com/browse/cryptography`.

[2] D. Kahn, *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet* (Macmillan, New York, 1967).

[3] M.A. Carnut e E.C. Hora, in: *Anais do Simpósio de Segurança em Informática*, editado por P.S.M. Pires e C.T. Fernandes, São José dos Campos, 2005.

[4] K.-W. Lee and H.-T. Ewe, in: *International Conference on Computational Intelligence and Security*, Guangzhou, edited by Y. Cheung, Y. Wang and H. Liu (The Institute of Electrical and Electronics Engineers, Piscataway, 2006).

[5] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, (John Wiley & Sons, New York, 1996), 2nd ed.

[6] J. Oliver, *Last Week Tonight with John Oliver: Government Surveillance* (HBO, 2015), disponivel em `https://youtu.be/XEVlyP4_11M`.

[7] A.G. Reinhold, The Diceware Passphrase Home Page (1995), disponivel em http://world.std.com/~reinhold/diceware.html.
[8] J. Bonneau, C. Herley, P.C. van Oorschot and F. Stajano, in: *Proc. IEEE Symposium on Security and Privacy*, San Francisco, 2012, editado por P. Kellenberger (Institute of Electrical and Electronics Engineers Inc., Piscataway, 2012), p. 553.
[9] N. Lee, Yahoo mail drops passwords and adds third-party email support for new apps (2015), disponivel em http://www.engadget.com/2015/10/15/yahoo-mail-update/.
[10] E. Grosse and M. Upadhyay, IEEE Computer and Reliability Societies **11**, 1 (2013).
[11] Wikipedia, Data breach: Major incidents (2015), disponivel em https://en.wikipedia.org/wiki/Data_breach, (acessada em 20 de Outubro de 2015).
[12] J. Ryall, Social media goes wild over massive celebrity nude photo leak (2014), disponivel em http://mashable.com/2014/08/31.
[13] N. Kerris and T. Muller, *Update to Celebrity Photo Investigation* (2014), disponivel em https://www.apple.com/pr/library/2014/09/02Apple-Media-Advisory.html.
[14] M. Zviran and W.J. Haga, Journal of Management Information Systems **15**, 4 (1999).
[15] R. Munroe, Password strength (2011), disponivel em https://xkcd.com/936/.
[16] C.E. Shannon, Bell Systems Technical Journal **27**, 3 (1948).
[17] C.E. Shannon, Bell Systems Technical Journal **28**, 4 (1949).
[18] S.W. Golomb, E. Berlekamp, T.M. Cover, R.G. Gallager, J.L. Massey and A.J. Viterbi, Notices of the American Mathematical Society **49**, 1 (2002).
[19] A. Kerckhoffs, Journal des Sciences Militaires **9** (1883).
[20] R. Landauer, IBM J. Res. Dev. **5**, 3 (1961).
[21] A. Bérut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider and E. Lutz, Nature **483**, 7388 (2012).
[22] M.A. Montemurro and D.H. Zanette, PLoS ONE **6**, e19875 (2011).
[23] G.K. Zipf, *Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology* (Addison-Wesley Press Inc., Cambridge, 1949).
[24] S. Abe, N. Suzuki, Physica A: Statistical Mechanics and its Applications **350**, 2 (2005).
[25] X. Gabaix, Quarterly Journal of Economics **114**, 3 (1999).
[26] B. Mandelbrot, The Journal of Business **36**, 4 (1963).
[27] C. Furusawa and K. Kaneko, Physical Review Letters **90**, 8 (2003).
[28] G. Nicolis, C. Nicolis and J.S. Nicolis, Journal of Statistical Physics **54**, 3(1989).
[29] P. Bak, *How Nature Works: The Science of Self-organized Criticality* (Copernicus, New York, 1996).
[30] M.E. Crovella and A. Bestavros, in: *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems* Philadelphia, 1996, edited by B.D. Gaither (ACM, New York, 1996).
[31] D.J. de Solla Price, Science **149**, 3683 (1965).
[32] R. Kohli and R.K. Sah, Management Science **52**, 11 (2003).
[33] R.A.K. Cox, J.M. Felton and K.C. Chung, Journal of Cultural Economics **19** (1995).
[34] M.E.J. Newman, Contemporary Physics **46**, 5 (2005).
[35] D. Malone and K. Maher, in: *Proceedings of the 21st International Conference on World Wide Web*, Lyon, 2012 (ACM, New York, 2012).
[36] D. Wang, G. Jian, X. Huang and P. Wang, in: *Transactions on Information and System Security* (ACM, New York, 2015), volume 1.
[37] L.L. Gonçalves and L.B. Gonçalves, Physica A **360**, 2 (2006).
[38] R.G. Piotrovskii, V.E. Pashkovskii and V.R. Piotrovskii, Nauchno-Tekhnicheskaya Informatsiya **28**, 11 (1994).
[39] L.C. Araujo, H.C. Yehia and T. Cristófaro-Silva, Glottometrics **26**, 38 (2013).
[40] R. Ferrer-i-Cancho, Proceedings of the National Academy of Sciences of the United States of America **100**, 3 (2003).
[41] C.E. Shannon, Bell Systems Technical Journal **30**, 1 (1951).
[42] T.M. Cover and R.C. King, IEEE Transactions on Information Theory **24**, 4 (1978).
[43] W.J. Teahan and J.G. Cleary, in: *Data Compression Conference*, Snowbird, 1996, edited by J.A. Storer and M. Cohn (IEEE Computer Society Press, 1996), p. 53.
[44] J.A. Thomas and T.M. Cover, *Elements of Information Theory* (Wiley, Hoboken, 1991), 2nd edition.
[45] R. Port, New Ideas in Psychology **25** (2007).
[46] F. de Saussure, *Curso de linguística Geral* (Editora Cultrix, São Paulo, 1916).
[47] C. Bowern and B. Evans, *The Routledge Handbook of Historical Linguistics* (Taylor & Francis, New York, 2015).

 DOI: http://dx.doi.org/10.1590/S1806-11173812125