

**UNIVERSIDADE DE BRASÍLIA  
FACULDADE DE TECNOLOGIA  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**UTILIZAÇÃO DE DICIONÁRIOS PROBABILÍSTICOS  
PERSONALIZADOS PARA DECIFRAR ARQUIVOS  
EM ANÁLISES PERICIAIS**

**LUCIANO LIMA KUPPENS**

**ORIENTADOR: ANDERSON CLAYTON ALVES NASCIMENTO, Dr.**

**DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA  
ÁREA DE CONCENTRAÇÃO INFORMÁTICA FORENSE E  
SEGURANÇA DA INFORMAÇÃO**

**PUBLICAÇÃO: PPGENE.DM - 090 A/12**

**BRASÍLIA/DF: JANEIRO/2012**

**UNIVERSIDADE DE BRASÍLIA  
FACULDADE DE TECNOLOGIA  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**UTILIZAÇÃO DE DICIONÁRIOS PROBABILÍSTICOS  
PERSONALIZADOS PARA DECIFRAR ARQUIVOS  
EM ANÁLISES PERICIAIS**

**LUCIANO LIMA KUPPENS**

DISSERTAÇÃO DE MESTRADO PROFISSIONALIZANTE SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

---

**ANDERSON CLAYTON ALVES NASCIMENTO, Dr., ENE/UNB  
(ORIENTADOR)**

---

**RAFAEL TIMÓTEO DE SOUSA JÚNIOR, Dr., ENE/UNB  
(EXAMINADOR INTERNO)**

---

**GEORGES DANIEL AMVAME NZÉ, Dr., FGA/UNB  
(EXAMINADOR EXTERNO)**

**BRASÍLIA/DF, 31 de JANEIRO DE 2012.**

## FICHA CATALOGRÁFICA

KUPPENS, LUCIANO LIMA

Utilização de Dicionários Probabilísticos Personalizados para decifrar arquivos em análises periciais [Distrito Federal] 2012. xiii, 99p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2012).

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Senha
2. Recuperação de senhas
3. Gramática Probabilística Especializada
4. Dicionário

I. ENE/FT/UnB. II. Título (série)

## REFERÊNCIA BIBLIOGRÁFICA

KUPPENS, L. L. (2012). Utilização de Dicionários Probabilísticos Personalizados para decifrar arquivos em análises periciais. Dissertação de Mestrado, Publicação PPGENE.DM – 090 A/12, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília/DF, 99p.

## CESSÃO DE DIREITOS

**NOME DO AUTOR: LUCIANO LIMA KUPPENS**

**TÍTULO DA DISSERTAÇÃO: Utilização de Dicionários Probabilísticos Personalizados para decifrar arquivos em análises periciais**

**GRAU/ANO: Mestre/2012**

É concedida à Universidade de Brasília permissão para reproduzir cópias desta Dissertação de Mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Do mesmo modo, a Universidade de Brasília tem permissão para divulgar este documento em biblioteca virtual, em formato que permita o acesso via redes de comunicação e a reprodução de cópias, desde que protegida a integridade do conteúdo dessas cópias e proibido o acesso a partes isoladas desse conteúdo. O autor reserva outros direitos de publicação e nenhuma parte deste documento pode ser reproduzida sem a autorização por escrito do autor.

---

Luciano Lima Kuppens

SQS 404 – D – 305

CEP: 70.238-040

Brasília/DF – Brasil

Aos meus pais e irmãos que sempre  
foram exemplos de dedicação e  
sabedoria a serem seguidos.

## **AGRADECIMENTOS**

Aos meus PAIS que sempre me incentivaram e que nunca pouparam esforços na educação de seus filhos.

Ao meu orientador, Prof. Dr. ANDERSON CLAYTON ALVES NASCIMENTO, pelo constante apoio, incentivo, paciência, dedicação e amizade, essenciais para o desenvolvimento deste trabalho.

Aos demais docentes do programa de mestrado do Departamento de Engenharia Elétrica da UnB, em especial, Prof. Dr. RAFAEL TIMÓTEO DE SOUSA JÚNIOR e Prof. Dr. FLÁVIO ELIAS GOMES DE DEUS, do Curso de Engenharia de Redes de Comunicação.

Aos Peritos Criminais Federais que tornaram este mestrado uma realidade, em especial, HÉLVIO PEREIRA PEIXOTO, CRIS AMON CAMINHA DA ROCHA e MARCOS VINÍCIUS GARCIA RODRIGUES DE LIMA.

Aos meus companheiros de baia, Peritos Criminais Federais MARCELO ANTONIO DA SILVA e JOSÉ HENRIQUE LOPES LINHARES DA SILVA, pela paciência diária, dicas e apoio imprescindíveis.

Ao meu irmão CRISTIANO LIMA KUPPENS pelo fornecimento de dados sem os quais essa obra não seria possível.

A todos, os meus sinceros agradecimentos.

O presente trabalho foi realizado com o apoio do Departamento de Polícia Federal com recursos do Programa Nacional de Segurança Pública com Cidadania – PRONASCI, do Ministério da Justiça.

## **RESUMO**

### **UTILIZAÇÃO DE DICIONÁRIOS PROBABILÍSTICOS PERSONALIZADOS PARA DECIFRAR ARQUIVOS EM ANÁLISES PERICIAIS**

**Autor: Luciano Lima Kuppens**

**Orientador: Anderson Clayton Alves Nascimento, Dr.**

**Programa de Pós-graduação em Engenharia Elétrica**

**Brasília/DF, janeiro de 2012**

O objetivo do presente trabalho é determinar se senhas geradas com gramáticas probabilísticas especializadas são mais eficazes para decifrar arquivos que os métodos normalmente utilizados durante uma análise pericial. Foi realizado um extenso estudo sobre utilização de senhas e uma análise comparativa dos métodos existentes para recuperá-las. O resultado da análise de cinquenta mil senhas de usuários brasileiros foi apresentado. Com essas senhas, coletadas durante a elaboração desta obra, foi criada uma gramática probabilística especializada brasileira que se mostrou bastante eficaz em relação a outras técnicas de recuperação de senhas, como regras de alteração, cadeias de Markov e força bruta. Essa gramática recuperou senhas mais complexas e mais rapidamente que as outras técnicas. No total, utilizando todos os métodos, foram descobertas 85% das senhas do grupo de controle. Finalmente, para utilização prática durante análises periciais, foi proposto o uso de Dicionários Probabilísticos Personalizados criados a partir da gramática brasileira e de dados biográficos e palavras-chave do caso sendo analisado.

# **ABSTRACT**

## **USE OF CUSTOM PROBABILISTIC DICTIONARIES TO DECIPHER FILES DURING A FORENSIC ANALYSIS**

**Author: Luciano Lima Kuppens**

**Supervisor: Prof. Anderson Clayton Alves Nascimento, PhD**

**Programa de Pós-graduação em Engenharia Elétrica**

**Brasília/DF, January 2012**

The objective of this study is to determine whether passwords generated with specialized probabilistic grammars are more effective to decipher files than the methods typically used during a forensic analysis. We conducted an extensive study on the use of passwords and a comparative analysis of existing methods to retrieve them. The result of the analysis of fifty thousand Brazilian user passwords was presented. With these passwords, collected during the preparation of this work, a Brazilian specialized probabilistic grammar was created. It proved quite effective in relation to other password recovery techniques, such as mangling rules, Markov chains and brute force. This grammar recovered faster and more complex passwords than the other techniques. In total, using all methods, 85% of passwords in the control group were discovered. Finally, for real application during a forensic analysis, we proposed the use of Custom Probabilistic Dictionaries created from the Brazilian grammar, biographical data and the wordlist of the case under analysis.

# SUMÁRIO

<b>1 - INTRODUÇÃO .....</b>	<b>1</b>
1.1 - JUSTIFICATIVA .....	2
1.2 - OBJETIVO .....	3
1.3 - METODOLOGIA.....	4
1.4 - RESULTADOS ESPERADOS .....	4
1.5 - LIMITAÇÕES .....	5
1.6 - ORGANIZAÇÃO .....	5
<b>2 - UTILIZAÇÃO DE SENHAS .....</b>	<b>7</b>
2.1 - SENHAS TRIVIAIS.....	7
2.2 - REUTILIZAÇÃO DE SENHAS .....	9
2.3 - MEMORIZAÇÃO DE SENHAS .....	10
2.4 - POLÍTICAS DE SENHAS .....	13
2.5 - SEGURANÇA E ENTROPIA DE SENHAS .....	15
2.6 - ALTERNATIVAS PARA O USO DE SENHAS .....	17
<b>3 - MÉTODOS DE RECUPERAÇÃO DE SENHAS .....</b>	<b>22</b>
3.1 - RECUPERAÇÃO DIRETA .....	22
3.2 - FORÇA BRUTA .....	23
3.3 - DICIONÁRIO .....	24
3.4 - REGRAS DE ALTERAÇÃO .....	25
3.5 - PROBABILÍSTICOS .....	26
3.5.1 - Cadeias de Markov.....	27
3.5.2 - Gramáticas Especializadas .....	28
3.6 - PRÉ-COMPUTADO .....	30
3.7 - HÍBRIDOS .....	31
<b>4 - FERRAMENTAS PARA RECUPERAÇÃO DE SENHAS .....</b>	<b>33</b>
4.1 - ACELERADORES.....	33
4.1.1 - Unidades de Processamento Gráfico.....	33
4.1.2 - Arranjo de Portas Programáveis .....	35
4.1.3 - Computação em Grade.....	36
4.1.4 - Computação na Nuvem .....	37
4.2 - APLICATIVOS .....	39
4.2.1 - Distributed Network Attack .....	39
4.2.2 - Elcomsoft Distributed Password Recovery.....	43
4.2.3 - Passware Kit Forensic.....	45
4.2.4 - John the Ripper .....	47



4.3 - ANÁLISE COMPARATIVA .....	49
4.3.1 - Métodos de Recuperação Disponíveis .....	49
4.3.2 - Aceleradores Suportados.....	50
4.3.3 - Algoritmos Criptográficos Compatíveis .....	50
4.3.4 - Resultados .....	53
<b>5 - DICIONÁRIO PROBABILÍSTICO PERSONALIZADO.....</b>	<b>55</b>
5.1 - OBTENÇÃO DAS SENHAS BRASILEIRAS.....	55
5.2 - ANÁLISE DAS SENHAS BRASILEIRAS .....	57
5.3 - TREINAMENTO DA GPE .....	60
5.4 - GERAÇÃO DO DICIONÁRIO PROBABILÍSTICO PERSONALIZADO .....	62
5.5 - LIMITAÇÕES .....	66
<b>6 - TESTES, RESULTADOS E ANÁLISE COMPARATIVA .....</b>	<b>67</b>
6.1 - TESTES REALIZADOS E RESULTADOS OBTIDOS .....	67
6.1.1 - Gramáticas Probabilísticas Especializadas .....	69
6.1.2 - John the Ripper .....	72
6.1.3 - Força Bruta.....	74
6.2 - ANÁLISE COMPARATIVA .....	74
6.2.1 - Treinamento da GPE (GPE <i>versus</i> GPE).....	74
6.2.2 - Regras de Alteração (GPE <i>versus</i> JtR-wordlist) .....	76
6.2.3 - Probabilísticos (GPE <i>versus</i> JtR-incremental).....	79
6.2.4 - Força Bruta (GPE <i>versus</i> Força Bruta) .....	82
6.3 - EXPERIMENTOS ADICIONAIS.....	83
6.3.1 - Realimentação .....	83
6.3.2 - Contínuo.....	85
<b>7 - CONCLUSÃO .....</b>	<b>87</b>
7.1 - CONTRIBUIÇÕES .....	90
7.2 - TRABALHOS FUTUROS .....	90
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>92</b>

## LISTA DE TABELAS

Tabela 3.1: Exemplos de alteração da palavra ‘sucesso’ .....	26
Tabela 4.1: Métodos de recuperação suportados pelos aplicativos analisados .....	49
Tabela 4.2: Ferramentas de aceleração suportadas pelos aplicativos analisados .....	50
Tabela 4.3: Sistemas e algoritmos compatíveis com os aplicativos analisados .....	50
Tabela 5.1: Relação das senhas mais utilizadas .....	58
Tabela 5.2: Relação das 15 estruturas gramaticas completas mais comuns .....	61
Tabela 5.3: Relação das 10 estruturas mais frequentes do tipo $D_1$ , $D_2$ , $D_3$ , $D_4$ e $D_5$ .....	61
Tabela 5.4: Relação das 10 estruturas mais frequentes do tipo $S_1$ , $S_2$ , $S_3$ , $S_4$ e $S_5$ .....	62
Tabela 6.1: GPE 50k e GPE 3k – 5 primeiras e 5 últimas sugestões geradas .....	71
Tabela 6.2: Dez primeiras regras de alteração do modo <i>wordlist</i> no JtR .....	73
Tabela 6.3: Senhas sugeridas e encontradas pelas GPEs com o dicionário “Completo” ....	75
Tabela 6.4: Primeiras senhas sugeridas e encontradas pela GPE e JtR-wordlist .....	78
Tabela 6.5: Primeiras senhas sugeridas e encontradas pela GPE e JtR-incremental.....	81

## LISTA DE FIGURAS

Figura 4.1: Interface de gerenciamento do DNA .....	41
Figura 4.2: Métodos de recuperação disponíveis para um arquivo .doc .....	41
Figura 4.3: Estatísticas de processamento em um arquivo sendo atacado .....	42
Figura 4.4: Estatísticas de senhas tentadas por segundo de um cliente .....	42
Figura 4.5: Interface de gerenciamento dos arquivos processados no EDPR .....	44
Figura 4.6: Interface de gerenciamento dos agentes no EDPR .....	44
Figura 4.7: Tela inicial do PKF .....	46
Figura 4.8: Escolha do método de recuperação a ser utilizado no PKF .....	46
Figura 4.9: Arquivo sendo processado no PKF .....	47
Figura 4.10: Opções disponíveis no JtR .....	48
Figura 5.1: Número de ocorrência de senhas com letras, números e símbolos .....	57
Figura 5.2: Número de senhas encontradas em relação ao seu tamanho .....	59
Figura 5.3: Proporção dos caracteres formadores em relação ao tamanho das senhas .....	59
Figura 5.4: Exemplo dos campos da ferramenta <i>Biographical Dictionary Generator</i> .....	65
Figura 6.1: Passos realizados no ataque por adivinhação .....	67
Figura 6.2: Número de senhas descobertas com o método GPE .....	72
Figura 6.3: Número de senhas descobertas com o método JtR .....	73
Figura 6.4: Número de senhas sugeridas <i>versus</i> senhas descobertas – GPEs .....	76
Figura 6.5: Número de senhas descobertas com os métodos GPE e JtR-wordlist .....	77
Figura 6.6: Número de senhas sugeridas <i>versus</i> senhas descobertas - Alteração .....	79
Figura 6.7: Número de senhas descobertas <i>versus</i> tamanho (utilizando GPE e JtR) .....	80
Figura 6.8: Proporção das senhas com sua composição (letras, dígitos e símbolos) .....	81
Figura 6.9: Número de senhas sugeridas <i>versus</i> senhas descobertas – diversos .....	82
Figura 6.10: Número de senhas descobertas com GPEs realimentadas .....	84
Figura 6.11: Senhas descobertas depois de mais de 24 dias de processamento .....	85

## **LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIACÕES**

AES – Advanced Encryption Standard

AWS – Amazon Web Services

CAPTCHA – Completely Automated Public Turing test to tell Computers and Humans Apart

CLOUD – Computação na Nuvem

COPACOBANA – Cost-Optimized Parallel Code Breaker

CPU – Central Processing Unit

DPP – Dicionário Probabilístico Personalizado

CUDA – Compute Unified Device Architecture

DES – Data Encryption Standard

DNA – Distributed Network Attack

DPE – Dicionário Probabilístico Especializado

EC2 – Amazon Elastic Compute Cloud

EDPR – Elcomsoft Distributed Password Recovery

FPGA – Field Programmable Gate Array

GPE – Gramática Probabilística Especializada

GPGPU – General purpose Computing on Graphics Processing units

GPU – Graphic Processing Unit

GRID – Computação em Grade

HMM – Hidden Markov Model

JtR – John the Ripper

LM – LAN Manager

MD4 – Message-Digest Algorithm 4

MD5 – Message-Digest Algorithm 5

MIMD – Multiple Instruction Multiple Data

NIST – National Institute of Standards and Technology

NTLM – NT LAN Manager

PCFG – Probabilistic Context-Free Grammar

PDF – Arquivo do tipo Portable Document Format

PGP – Pretty Good Privacy

PIN – Personal Identification Number

PKF – Passware Kit Forensics

PRTK – Password Recovery Toolkit

PSK – Pre-Shared Key

RAR – Tipo de arquivo compactado (Roshal ARchive)

RC4 – Rivest Cipher 4

RT – Rainbow Tables

SHA – Secure Hash Algorithm

SIMD – Single Instruction Multiple Data

TACC – Tableau Hardware Accelerator

TI – Tecnologia da Informação

WPA – Wi-Fi Protected Access

XOR – Função OU exclusivo

ZIP – Tipo de arquivo compactado

# 1 - INTRODUÇÃO

Métodos criptográficos robustos têm sido utilizados em grande variedade de situações: comércio eletrônico, assinatura digital, votação eletrônica, segurança nacional, liberdade de expressão, entre outras. Ferramentas para a proteção de arquivos e discos rígidos estão disponíveis gratuitamente na Internet, garantindo níveis seguros de confidencialidade a qualquer pessoa ou empresa.

A utilização desses métodos é desejada em uma série de ocasiões. Entretanto, organizações criminosas, integrantes de redes de pedofilia e grupos terroristas também podem usar métodos criptográficos robustos para impedir o acesso da polícia a atos criminosos.

Quanto mais pessoas passarem a utilizar ferramentas criptográficas em suas casas e empresas, mais difícil será o trabalho das entidades de Segurança Pública, em especial, dos setores técnico-científicos responsáveis pelo exame pericial nos equipamentos apreendidos.

Computadores e arquivos criptografados podem aumentar o custo e atrasar uma investigação criminal. Se bem empregada, a criptografia pode ser uma forma eficiente de obstrução à Justiça. Com ela, pode-se impossibilitar o acesso a dados essenciais para um caso e, conseqüentemente, impedir a materialização da prova e a responsabilização penal dos envolvidos. Esse problema pode se tornar tão grave que alguns governos já se precaveram e modificaram sua legislação.

No Reino Unido, a legislação obriga que o suspeito sob investigação decifre quaisquer dados criptografados ou que forneça suas chaves criptográficas à autoridade policial. Caso o investigado se negue a cooperar, ele pode ser sentenciado a até cinco anos de prisão (Someren, 2007).

Nos Estados Unidos, a quinta emenda à Constituição daquele país protege seus cidadãos de abusos do governo e assegura que um suspeito não precisa revelar informações auto incriminadoras. Apesar disso, (Lowman, 2010) relata alguns casos em que o investigado foi obrigado a fornecer suas senhas em juízo, permitindo a condenação dos envolvidos.

A legislação brasileira também não obriga que o suspeito produza prova contra si próprio. Mesmo que todas as evidências apreendidas em uma determinada investigação estejam

criptografadas, o suspeito não precisa fornecer suas senhas durante o inquérito policial ou processo judicial.

Entretanto, a ampla literatura científica relacionada à utilização de senhas afirma que, de modo geral, pessoas não utilizam senhas seguras. Isso ocorre em função de uma série de fatores. O principal deles é a limitação na capacidade humana de memorização.

Senhas seguras devem ser longas e imprevisíveis, compostas por letras minúsculas, maiúsculas, números e símbolos, não devem conter informações pessoais ou palavras comuns e nunca devem ser reutilizadas. No limite, uma senha com entropia máxima é aquela formada por uma sequência de caracteres aleatórios. Assim, quanto mais segura, mais difícil de ser lembrada.

Segundo (Zhang *et al.*, 2009), para serem efetivas, senhas precisam ser fortes, secretas e memorizáveis. Senhas fortes são aquelas difíceis de serem adivinhadas; senhas secretas são aquelas difíceis de serem obtidas por terceiros; e senhas memorizáveis são aquelas facilmente lembradas por usuários. Mas de acordo com (Narayanan & Shmatikov, 2005), enquanto usuários precisarem memorizar suas senhas, elas serão vulneráveis a ataques inteligentes.

Encontrar um ponto de equilíbrio entre usabilidade e segurança é um processo difícil. Diversos estudos comprovam que o usuário típico acaba optando pela usabilidade e conveniência. Isso normalmente se traduz em senhas inseguras, que podem ser exploradas pelas entidades de Segurança Pública.

## **1.1 - JUSTIFICATIVA**

Os peritos responsáveis pela análise de materiais apreendidos precisam obter acesso aos dados mesmo quando estes estão criptografados. Quando mecanismos criptográficos robustos são empregados, é inviável encontrar fragilidades nos algoritmos utilizados e, até com o uso de poderosos supercomputadores, é intratável testar todas as chaves possíveis.

A abordagem prática nesses casos é recuperar a senha utilizada para gerar a chave criptográfica. Ou seja, a tática é explorar o elemento mais fraco envolvido: a senha criada pelo usuário.

Geralmente, senhas inseguras são fáceis de serem lembradas. São sequências curtas ou simples, compostas por dados pessoais do usuário ou palavras encontradas em dicionários. Além disso, é muito comum a reutilização de senhas em locais diferentes. Assim, parte das senhas utilizadas na prática são previsíveis.

A recuperação dessas senhas normalmente está associada à criação de bons dicionários. Em geral, esses dicionários são personalizados, criados especificamente para o caso em análise. Dicionários personalizados devem conter dados biográficos do suspeito e listas de palavras recuperadas de outras mídias apreendidas.

Além de personalizados, bons dicionários podem ser criados por métodos probabilísticos, em especial, por meio de gramáticas especializadas e cadeias de Markov.

Assim, visando o atendimento célere de exames periciais de dados criptografados, o perito precisa ter à sua disposição dicionários mais eficazes. O presente trabalho propõe a utilização em conjunto de técnicas probabilísticas e personalizadas para a criação de dicionários, ou seja, a geração de um Dicionário Probabilístico Personalizado (DPP).

## **1.2 - OBJETIVO**

O objetivo deste trabalho é determinar se o método de gramáticas probabilísticas especializadas é mais eficaz para decifrar arquivos apreendidos do que os métodos normalmente utilizados durante uma análise pericial.

Estão à disposição do perito diversos métodos para recuperar ou prever uma senha. São eles: recuperação direta, força bruta, dicionários, técnicas de alteração, algoritmos probabilísticos e técnicas pré-computadas.

Esses métodos, por sua vez, podem ser acelerados por meio de quatro ferramentas diferentes: unidades de processamento gráfico, arranjo de portas programáveis, computação em grade e computação na nuvem. Todo esse processo pode ser automatizado com o uso de aplicativos desenvolvidos especificamente para esse fim.



O presente trabalho visa ainda indicar qual abordagem (método, ferramenta e aplicativo) é mais adequada para o tratamento de dados criptografados no contexto de uma análise pericial.

### **1.3 - METODOLOGIA**

Diversos métodos de geração de senhas serão analisados e comparados. Isso será feito utilizando uma base de controle com mais de 2,6 milhões de *hashes* de senhas de usuários brasileiros.

Para cada método de geração de senhas testado, será registrado o número de senhas obtidas e a velocidade com que elas são descobertas. Serão objeto de estudo: métodos probabilísticos (gramáticas especializadas e cadeias de Markov), regras de alteração e força bruta.

Será utilizado o método de Gramáticas Probabilísticas Especializadas (GPEs) proposto por (Glodek, 2008) e aperfeiçoado por (Weir *et al.*, 2009). Primeiramente, essa técnica aprende a estrutura de formação das senhas (frequência e posição das letras, dígitos e símbolos). Em seguida, palavras presentes em um dicionário auxiliar são utilizadas para gerar sugestões de senhas em ordem decrescente de probabilidade.

Para a confecção de uma GPE é necessária a obtenção de senhas em claro, que são utilizadas na fase de aprendizado da gramática. Para isso, foram obtidas 50 mil senhas de usuários brasileiros. Um quadro estatístico dessas senhas será apresentado (ocorrências mais frequentes, tamanho médio, frequência de determinados caracteres, taxa de reutilização e outros).

### **1.4 - RESULTADOS ESPERADOS**

A análise comparativa dos métodos de recuperação será feita apenas para senhas de usuários brasileiros. Dessa forma, por ser um método probabilístico e adaptativo, é esperado que as GPEs obtenham um melhor resultado em relação aos demais métodos a

serem testados. Se isso se confirmar, DPPs serão ótimos candidatos para decifrar arquivos sob análise.

## 1.5 - LIMITAÇÕES

Para a criação de um DPP, deve-se utilizar um dicionário personalizado (gerado com dados pessoais do suspeito e listas de palavras recuperadas de outras mídias apreendidas).

Um problema da proposta em questão é a dificuldade de se realizar testes, já que cada DPP só será válido para um caso em análise e, em última instância, para a recuperação de apenas uma senha. Diante dessa dificuldade, o presente trabalho apenas descreverá os passos para a criação de DPPs e os testes utilizarão DPEs (ou seja, dicionários probabilísticos especializados sem a personalização biográfica do caso).

Além disso, o trabalho é baseado no contexto de uma análise pericial *post-mortem* e *off-line*. Ou seja, os arquivos cifrados encontram-se em computadores desligados e nenhum sistema ativo será objeto de estudo. Assim, não serão abordadas técnicas de recuperação de senhas que utilizam sistemas ou sítios *on-line*, análise de memória com o computador ligado, captura de dados utilizando *rootkits*, capturadores de teclado ou análise de sons e campos eletromagnéticos emitidos por dispositivos, por exemplo.

## 1.6 - ORGANIZAÇÃO

O restante deste trabalho está organizado da seguinte forma:

- *Capítulo 2*: realiza um estudo minucioso na utilização de senhas, incluindo hábitos comuns, fragilidades, cálculo de entropia e métodos alternativos;
- *Capítulo 3*: descreve em detalhes os métodos de recuperação de senhas existentes, desde os lentos e ineficientes ataques por força bruta até as técnicas mais modernas de pré-computação;
- *Capítulo 4*: realiza um levantamento das ferramentas para a recuperação de senhas mais utilizadas no ambiente pericial, incluindo dispositivos e aplicativos para esse trabalho. Esse capítulo faz ainda uma análise

comparativa dessas ferramentas, relacionando os métodos de recuperação disponíveis, os aceleradores suportados e os algoritmos criptográficos compatíveis;

- *Capítulo 5*: aborda os métodos de GPEs e DPPs. Este capítulo apresenta também a análise estatística das senhas brasileiras coletadas;
- *Capítulo 6*: apresenta os testes realizados, resultados obtidos e faz uma análise comparativa entre os métodos empregados;
- *Capítulo 7*: são apresentadas as conclusões, contribuições e sugestões para trabalhos futuros.

## **2 - UTILIZAÇÃO DE SENHAS**

A utilização de senhas para proteger o acesso a serviços on-line, sistemas ou arquivos é um recurso amplamente utilizado e bastante conveniente. Porém, como será mostrado neste capítulo, esse método de autenticação, por basicamente depender da capacidade humana de memorizar caracteres, pode ser bastante frágil.

A autenticação por senhas é um método rápido, simples e barato de ser implementado. Em função disso, embora existam diversos mecanismos alternativos, a autenticação por senhas não deve ser, tão cedo, completamente substituída (Forget *et al.*, 2008).

A autenticação por senhas é baseada no simples conceito de segredo compartilhado (apenas o usuário e o sistema devem ter informação sobre a senha). Um dos elementos chave dessa solução é confiança na nossa habilidade cognitiva de nos lembrarmos desse segredo. Mas esse modelo era efetivo quando existiam poucos sistemas computacionais e seus usuários precisavam guardar apenas “alguns segredos”. Nos dias de hoje, esse método é bastante frágil, pois usuários precisam se lembrar de várias senhas de diversos sistemas diferentes (Conklin, Dietrich & Walz, 2004).

### **2.1 - SENHAS TRIVIAIS**

A criação e utilização de senhas têm sido um assunto muito pesquisado pela comunidade científica. Diversos estudos acadêmicos têm comprovado que usuários criam senhas fáceis de serem lembradas e, conseqüentemente, fáceis de serem adivinhadas. Em contrapartida, senhas difíceis de serem descobertas são, por sua vez, difíceis de serem recordadas. Encontrar o ponto de equilíbrio entre usabilidade e segurança não é uma tarefa simples.

(Morris & Thompson, 1979), ao desenvolverem um esquema de autenticação para o acesso remoto a um sistema compartilhado, realizaram uma das primeiras pesquisas nesse campo. Seus experimentos mostraram que 86% das senhas criadas, quando nenhuma restrição era imposta, eram extremamente fracas. Essas senhas eram muito curtas, continham apenas letras minúsculas e/ou dígitos ou eram facilmente encontradas em dicionários.

Esses pesquisadores chegaram a propor recomendações para tornar o processo mais seguro, incluindo a utilização de senhas mais longas e com símbolos ou simplesmente

deixar o sistema criar a senha para o usuário. Foi proposta também a utilização de *salts*, que é a combinação da senha com dados adicionais, evitando alguns tipos de ataques. Essas recomendações se tornaram a base das técnicas de prevenção utilizadas hoje (St. Clair *et al.*, 2006).

Entretanto, mais de 30 anos após esse trabalho e, apesar dos incontáveis avanços tecnológicos desde então, recentes pesquisas comprovam que o hábito de criar senhas inseguras pouco mudou.

(Florêncio & Herley, 2007) realizam um detalhado estudo sobre a utilização de senhas na Internet. Por meio da inclusão de um módulo de captura de dados na barra de ferramentas do *Windows Live*, os pesquisadores da *Microsoft* foram capazes de monitorar e coletar uma variedade de métricas relativas à utilização de senhas de mais de 500.000 usuários.

Comprovou-se que usuários utilizam senhas fracas e, quando o sistema não obrigava a utilização de dígitos ou símbolos, a grande maioria dessas senhas era composta apenas por letras minúsculas. Além disso, verificou-se que os usuários esquecem suas senhas com uma frequência muito alta (Florêncio & Herley, 2007).

(Brown *et al.*, 2004) constataram que dois terços das senhas são criadas utilizando dados biográficos do próprio usuário ou de pessoas próximas a ele. Os dados mais utilizados nessas senhas são nomes, seguido de datas de aniversário, números de identidade e números de telefone. Resultado semelhante foi obtido por (Riley, 2006) e (Shay *et al.*, 2010).

De um modo geral, diversos estudos englobando dados controlados e empíricos mostram que, caso não haja restrições impostas pelo sistema de autenticação, usuários utilizam senhas triviais. Como exemplo, pode-se citar os experimentos de (Vu *et al.*, 2007) que resultaram na quebra de 50% das senhas testadas em menos de 4 horas.

Porém, como mostram (St. Clair *et al.*, 2006) o usuário nem sempre tem toda parte da culpa. Sistemas baseados em implementações antigas da função `crypt()` no *Unix* limitam a senha a oito caracteres. Dessa forma, mesmo que o usuário deseje uma senha mais longa, ele é impedido de criá-la. Em função disso, alguns sistemas ainda hoje apresentam a restrição de oito caracteres. Assim, usuários frequentemente escolhem senhas de seis a oito caracteres sabendo que elas serão universalmente aceitas (St. Clair *et al.*, 2006).

Os resultados de (Riley, 2006) indicam algo ainda mais relevante: usuários não têm o hábito de variar a complexidade de suas senhas para contas mais importantes, ou seja, um serviço bancário on-line terá uma senha similar a um serviço de postagem de mensagens, por exemplo. Um agravante, como será abordado em seguida, é a comprovação de que a reutilização de senhas é um hábito bastante comum.

## **2.2 - REUTILIZAÇÃO DE SENHAS**

A proliferação de acessos protegidos por senha faz com que usuários reutilizem suas senhas em uma variedade de contas distintas. Com isso, caso uma senha seja comprometida, todas as demais contas e sistemas que compartilham aquela senha estão em risco (Conklin, Dietrich & Walz, 2004).

“Usuários que reutilizam senhas não percebem que sua conta mais bem protegida é tão segura quanto aquela com a proteção mais fraca [...]. Como dominós, quando um sistema vulnerável é comprometido, informações que ajudarão a infiltrar outras contas serão reveladas, podendo levar à queda de outros sistemas, incluindo aqueles muito mais seguros que o primeiro” (Ives, Walsh & Schneider, 2004).

(Bonneau & Preibusch, 2010) apontam uma série de falhas técnicas em sítios com baixos níveis de segurança que podem levar ao comprometimento de sistemas seguros.

Além disso, um sítio falso criado por um atacante pode ser utilizado para coletar senhas e comprometer diversas contas. (Schneier, 2000) sugere ainda que o sítio falso pode rejeitar todas as senhas fornecidas e, com isso, o usuário (sem saber qual a senha correta para aquele serviço) acaba fornecendo várias senhas que possui.

Esse comportamento foi mostrado por (Florêncio & Herley, 2007), indicando que a reutilização de senhas é tão frequente que o problema do usuário passa a ser lembrar qual daquelas 5 ou seis senhas que possui é a correta para o serviço sendo acessado. (Zhang *et al.*, 2009) identificaram que, de fato, usuários confundem qual a senha correta para

determinada conta. O que os pesquisadores chamaram de “interferência entre senhas”, seria uma das principais razões para as falhas durante o processo de autenticação.

Para evitar ataques de força bruta, alguns sistemas utilizam técnicas que bloqueiam o acesso a uma conta após determinado número de tentativas incorretas. O acesso é liberado após intervenção do administrador do sistema ou após determinado período de tempo.

Entretanto, (Gehring, 2002) afirma que esse método acaba se tornando mais um incentivo para que usuários passem a utilizar uma variedade menor de senhas, ou mesmo que passem a anotá-las. Além disso, a técnica em questão pode ser intencionalmente utilizada para provocar um ataque de negação de serviço, impedindo que determinado usuário acesse legitimamente sua conta.

(Florêncio & Herley, 2007) mostram ainda que, em média, usuários possuem 6,5 senhas, sendo que cada uma é compartilhada com 4 contas diferentes (cada usuário possui 25 contas) e oito senhas são digitadas por dia. Valores atualizados, mas similares a esses, podem ser encontrados em (Hayashi & Hong, 2011).

Usuários têm dificuldades em criar e lembrar-se de novas senhas. O pensamento “É muito mais fácil se lembrar de uma senha que já foi utilizada” foi verificado por (Gaw & Felten, 2006) em sua pesquisa. O processo envolvido na memorização de senhas será abordado em seguida.

### **2.3 - MEMORIZAÇÃO DE SENHAS**

O ser humano não tem dificuldades em criar e memorizar um número pequeno de senhas. Entretanto, se o número de senhas aumenta, a probabilidade de lembrar-se de uma senha específica diminui (Vu *et al.*, 2007).

Diversas pesquisas foram realizadas visando identificar o processo envolvido na memorização de senhas. (Adams & Sasse, 1999) concluíram que escolher uma senha segura e ao mesmo tempo fácil de ser lembrada, é uma tarefa difícil para muitos usuários. Resultado semelhante foi encontrado por (Yan *et al.*, 2004) com um grupo de estudantes em um experimento controlado.

Segundo (St. Clair *et al.*, 2006), a memória humana apresenta uma limitação significativa: lembrar-se de mais de 7 itens aleatórios é algo difícil. Dentre outras conclusões, os pesquisadores confirmaram que usuários têm dificuldades em se lembrar de senhas que não sigam um padrão, ou seja, sequências aleatórias são facilmente esquecidas.

A razão disso são três características bem conhecidas da memória humana: primeira, nossa capacidade de memorizar itens em sequência é limitada. Segunda, quando nos lembramos de uma sequência de itens, esses itens não surgem de um conjunto arbitrário qualquer, mas sim de palavras e símbolos familiares. Terceira, nossa memória necessita de fontes redundantes de informação (Yan *et al.*, 2000).

Visando produzir senhas ao mesmo tempo fortes e fáceis de serem lembradas, várias técnicas foram propostas. Uma delas é a utilização de senhas mnemônicas. Com essa técnica, o usuário precisa memorizar um texto-base e a senha é gerada pelas letras iniciais da sentença. Adicionalmente, letras podem ser substituídas por símbolos e elementos de pontuação podem ser utilizados. Um exemplo de uma senha mnemônica é *Le\$\$P!* gerada a partir do texto-base “Luke, eu sou seu Pai!”. Um bom resultado desta técnica foi obtido por (Vu *et al.*, 2007).

Por outro lado, já foi demonstrado que esse método não é tão seguro como esperado. Isso se deve ao fato de muitos usuários utilizarem como textos-base sentenças comuns, disponíveis na Internet (como refrões de músicas, citações famosas, frases de filmes ou livros). Assim, é possível prever esse comportamento e criar ataques de dicionário específicos para senhas mnemônicas. Entretanto, em função do grande número possível de textos-base e da baixa frequência com que esse método é utilizado, a criação de um dicionário eficiente é uma tarefa não trivial (Kuo *et al.*, 2006).

(Helkala & Snekenes, 2009) afirmaram que usuários são sim capazes de gerar boas senhas (que sejam ao mesmo tempo complexas e fáceis de lembrar), especialmente se puderem utilizar seus próprios processos de criação. Esses processos podem produzir senhas em três categorias:

1. *Palavras*: sequência de caracteres que possuem um termo encontrado em dicionário, ex.: *cachorro*;
2. *Mescladas*: sequência que contém termos de dicionários E elementos que não formam uma palavra, ex.: *cachorro123*;



3. *Pseudo-Palavras*: sequência que contém apenas termos de dicionários incluindo a possibilidade de algumas modificações, ex: *c@ch0rr0*.

Para cada categoria, (Helkala & Snekkenes, 2009) propuseram regras para a geração de boas senhas. Por exemplo, senhas devem ter mais de 12, 10 e 08 caracteres para as categorias 1, 2 e 3, respectivamente. Dessa forma, o usuário poderia escolher uma das categorias e, seguindo as regras propostas, criar uma senha complexa e fácil de ser lembrada.

Mas (Hart, 2008) mostra que até mesmo educar usuários a utilizar métodos mais seguros não é uma tarefa simples. (Riley, 2006) conclui que os participantes de sua pesquisa, apesar de conhecerem as boas práticas associadas à criação e utilização de senhas, simplesmente não as utilizavam.

Além disso, um estudo feito por (Forget *et al.*, 2007) com um pequeno grupo de estudantes instruídos a utilizarem senhas mnemônicas, sugere que a abordagem tradicional utilizada para orientar usuários a criar senhas relevantes é ineficaz. O estudo sugere que um conjunto de regras ou novos esquemas de criação de senhas não são a solução. A proposta é a utilização de métodos “persuasivos” para obter senhas seguras.

O método proposto foi desenvolvido e testado em (Forget *et al.*, 2008) utilizando “Tecnologias de Persuasão”, que consistem em modificar atitudes e comportamentos de um indivíduo por meio de persuasão ou influência social. A ideia da técnica é influenciar a escolha de senhas melhores inserindo caracteres aleatórios na senha criada pelo usuário. O usuário pode trocar a posição dos caracteres inseridos até que se sinta confortável com o resultado obtido. Segundo o estudo, esse método aumentou, de forma geral, a qualidade das senhas sem comprometer sua usabilidade.

Entretanto, o estudo mostrou que se forem inseridos mais de três caracteres aleatórios, os usuários passam a ter dificuldades em recordar sua senha, o que gera um limitador significativo para o método em questão.

Além de todos esses aspectos relacionados à (falta de) capacidade humana para memorizar senhas, políticas de segurança que obrigam a sua troca em intervalos determinados de tempo são mais um desafio para usuários comuns, como será abortado em seguida.

## 2.4 - POLÍTICAS DE SENHAS

Como mostrado, é fato que usuários utilizam senhas fracas (curtas ou facilmente encontradas em dicionários). Isso significa que o conjunto de senhas efetivamente utilizado é muito menor que o conjunto de senhas teoricamente disponível, o que aumenta drasticamente a possibilidade de um atacante descobrir determinada senha (Shay *et al.*, 2010).

Para tentar combater essa fragilidade, políticas de senhas são normalmente empregadas. Essas políticas são regras de criação que podem especificar, por exemplo, o número mínimo de caracteres permitidos e a obrigatoriedade de letras maiúsculas, dígitos e/ou símbolos. O propósito dessas regras é garantir o aumento do conjunto de possíveis senhas (Shay *et al.*, 2010).

A utilização de uma política de senhas adequada é vital para tornar o processo de autenticação mais seguro e amigável (Hayashi & Hong, 2011). Entretanto, exigir senhas com certo nível de complexidade ou frequentemente obrigar um usuário a trocá-las podem acabar criando subterfúgios indesejados. Mais uma vez, encontrar o equilíbrio adequado não é uma tarefa simples.

“Políticas de senhas parecem ser hoje mais um processo de adivinhação e heurística do que ciência e cálculo” (Shay *et al.*, 2007).

(Kuo *et al.*, 2006) e (Shay *et al.*, 2007) apontam que se uma política de criação de senhas não exige um nível suficiente de complexidade, usuários estarão em risco (pelo fato de criarem senhas triviais). Por outro lado, caso seja muito complexa, usuários tendem a reutilizar senhas ou simplesmente anotá-las.

(Inglesant & Sasse, 2010) observam que políticas de senhas muito restritivas podem impactar negativamente na produtividade de um funcionário. Além disso, por mais eficientes que forem essas políticas, elas não são garantia contra alguns tipos de ataques (como *phishing* e *key loggers*, por exemplo).

Segundo (Florêncio, Herley & Coskun, 2009), a utilização de nomes de usuário mais longos (e não as respectivas senhas) – desde que não sejam publicamente expostos – parece desempenhar um papel melhor na segurança daquela conta como um todo.

Já (Komanduri *et al.*, 2011) concluíram que senhas simples e mais longas (com apenas a exigência de ter no mínimo de 16 caracteres) são mais fáceis de serem criadas e lembradas, além de serem mais seguras se comparadas a senhas muito curtas e complexas (com símbolos, dígitos e letras maiúsculas, por exemplo).

Verifica-se ainda que a reutilização de senhas poderá acontecer se a política obrigar o usuário a trocar suas senhas com uma frequência muito alta (Grawemeyer & Johnson, 2009).

(Adams & Sasse, 1999) identificaram uma série de problemas nessas políticas com “data de validade”. Como são sempre surpreendidos e forçados a trocarem suas senhas, com o passar do tempo, usuários passam a utilizar senhas mais simples de serem lembradas e, conseqüentemente, mais inseguras. A pesquisa conclui que, apesar dessa política de troca de senhas procurar reduzir eventuais brechas em uma organização, o efeito pode ser justamente o oposto.

Já os experimentos de (Zhang, Monroe & Reiter, 2010) concluem que a política de expirar uma senha após determinado tempo é ineficaz porque usuários criam senhas muito parecidas com as anteriores. (Shay *et al.*, 2010) mostram que usuários tendem a criar novas senhas apenas modificando alguns poucos caracteres das senhas antigas. Além disso, como já dito, essa política só ajuda se o atacante for muito lento para explorar as credenciais obtidas (Florêncio, Herley & Coskun, 2009).

Outros usuários insatisfeitos a política podem “resolver o problema” de uma forma mais inusitada: sempre que são forçados a trocarem sua senha, efetuam esse procedimento repetidamente até que atingir o limite do histórico de senhas do sistema. Com isso, eles podem utilizar sua “senha padrão” novamente.

(Schechter, Herley & Mitzenmacher, 2010) sugerem a substituição de todas essas complexas políticas de senhas por apenas uma: o usuário pode ter qualquer senha que desejar, desde que ela não seja um alvo fácil de um ataque estatístico, ou seja, senhas populares não seriam permitidas.

Sem grandes surpresas, (Komanduri *et al.*, 2011) mostraram que senhas se tornam mais resistentes a ataques quando, no momento de sua criação, são rejeitadas aquelas presentes

em determinados conjuntos de listas públicas (contendo palavras de dicionários, senhas e frases comuns, nomes diversos, entre outras).

A utilização de algoritmos que determinam se uma senha é satisfatória ou não é uma alternativa levantada por (Gehring, 2002) e (Vu *et al.*, 2007) para aumentar a segurança de senhas criadas por usuários. Esse método permite que o usuário escolha sua senha, mas o sistema verifica se ela está dentro dos critérios estabelecidos. Esse método é conhecido como “Checagem Proativa de Senhas” e força ou recomenda que o usuário escolha senhas complexas o suficiente.

Porém, (Dell’Amico, Michiardi & Roudier, 2010) argumentam que, normalmente, esses sistemas são baseados em métricas muito simples (como o tamanho da senha, sua resistência a ataques de força-bruta ou dicionários), e não levam em conta técnicas avançadas de exploração.

Mas como garantir que uma senha tenha um nível de complexidade suficiente para ser considerada segura? Esse assunto será abordado em sequência.

## **2.5 - SEGURANÇA E ENTROPIA DE SENHAS**

Uma forma de medir a segurança de uma senha é calcular sua entropia. O conceito de entropia foi primeiramente formalizado por (Shannon, 1948), que se tornou a base da Teoria da Informação e um marco muito importante para o atual estágio tecnológico em que vivemos.

A entropia quantifica a incerteza envolvida ao estimarmos o valor de uma variável aleatória (Shay *et al.*, 2010). Mais especificamente, a entropia de uma senha é a medida (em bits) de sua incerteza. Portanto, a entropia de uma senha pode ser considerada uma estimativa para a quantidade média de trabalho necessário para adivinha-la (Burr, Dodson & Polk, 2006).

“Normalmente, quanto maior a entropia em uma dada distribuição de senhas, maior será a dificuldade em adivinhar uma senha selecionada dessa distribuição. [...] Senhas com valores mais

altos de entropia requerem um número maior de tentativas para serem descobertas, tornando a entropia uma medida útil para a segurança de uma senha” (Shay *et al.*, 2010).

Uma senha com a máxima entropia possível seria formada pela sequência de caracteres mais longa permitida, escolhida de forma totalmente aleatória (sem redundância) utilizando o conjunto de todos os caracteres disponíveis (Yan *et al.*, 2000).

O método mais utilizado pela comunidade acadêmica para estimar a entropia de uma senha foi definido pelo NIST<sup>1</sup> no *Electronic Authentication Guideline SP-800-63* (Burr, Dodson & Polk, 2006). Dentre outras recomendações, essa publicação fornece orientações técnicas para a implementação de mecanismos de autenticação por senha. A norma em questão influenciou as políticas de criação e uso de senhas como nenhuma outra e suas recomendações tornaram-se a base de diversas políticas na indústria e no governo (Weir *et al.*, 2010).

De acordo com o documento, a entropia ( $H$ , em bits) de uma senha pode ser estimada de duas formas:

a) Se a senha for escolhida aleatoriamente:

$$H = \log_2(b^l) \quad (2.1)$$

Onde  $b$  é o número de caracteres do alfabeto utilizado e  $l$  o comprimento da senha.

b) Se a senha for criada pelo usuário, a entropia será o somatório de:

- 4 bits para o primeiro caractere;
- 2 bits por caractere (do 2º ao 8º caractere);
- 1,5 bit por caractere (do 9º ao 20º caractere);
- 1 bit por caractere (a partir do 21º caractere);
- 6 bits adicionais se houver letras maiúsculas e caracteres não alfabéticos (que podem ser números e/ou símbolos);
- 6 bits adicionais se não for uma senha presente em dicionários (ou listas de nomes).

---

<sup>1</sup> National Institute of Standards and Technology

Por serem caracteres escolhidos aleatoriamente, a equação (2.1) apresentada em (a) é exatamente a fórmula de entropia definida em (Shannon, 1948).

Mas quando a senha é escolhida pelo usuário, sua entropia é muito mais difícil de ser estimada. Para tanto, (Burr, Dodson & Polk, 2006) levaram em consideração a frequência de caracteres e padrões de textos comuns do idioma inglês. Isso foi feito com base na previsibilidade de palavras, utilizando os resultados presentes em (Shannon, 1951). O cálculo considera ainda um universo de 94 caracteres, composto por letras maiúsculas, minúsculas, números e os seguintes símbolos: “”! ? @ # \$ % & ^ \* ~ ( ) { } [ ] < > \ / \_ - ' : ; , . + = | . O resultado dessa estimativa é o somatório apresentado em (b).

Portanto, a norma não cria políticas específicas (que ficam a cargo dos desenvolvedores), mas proporciona um método para calcular a entropia que uma senha precisa atingir para se enquadrar em um determinado nível de segurança (Bonneau & Preibusch, 2010).

Entretanto, apesar de sua grande aceitação, a entropia por si só pode não ser a melhor maneira de avaliar a segurança de uma senha (Weir *et al.*, 2010). (Komanduri *et al.*, 2011), por exemplo, avaliaram duas políticas de senha bem diferentes que proporcionavam níveis de entropia parecidos. Porém, os níveis de segurança e usabilidade dessas duas políticas foram muito distintos.

Além disso, é possível que em uma dada distribuição de senhas com alta entropia, haja um grupo pequeno de senhas que sejam fáceis de serem descobertas (Komanduri *et al.*, 2011). Diante das fragilidades apresentadas, uma série de alternativas para o uso de senhas foram propostas e estão sendo empregadas. As principais serão abordadas em seguida.

## **2.6 - ALTERNATIVAS PARA O USO DE SENHAS**

Apesar das diversas fragilidades associadas à utilização de senhas, sua facilidade de uso e presença quase universal têm impedido a substituição sistemática desse método de autenticação por outros mais seguros.

“Do ponto de vista da usabilidade, senhas e PINs chegaram ao fim de sua vida útil. Apesar de serem convenientes para desenvolvedores, para

os usuários são cada vez mais impraticáveis. Hoje, cada usuário precisa gerenciar dezenas de senhas. Com isso, as restrições impostas (senhas complexas, sempre diferentes e nunca anotadas) já não são razoáveis. Contudo, não podemos abandonar-las até que surja um método alternativo de autenticação que seja ao mesmo tempo prático e seguro” (Stajano, 2011).

Existem diversos esquemas alternativos de autenticação, como biometria, *smartcards*, *one-time passwords*, senhas gráficas e vários outros (Kuo *et al.*, 2006). Entretanto, todos esses métodos envolvem contrapartidas e diversas considerações importantes. Entre elas pode-se citar: o custo, o local de utilização, limitações físicas e até implicações de saúde pública (como leitores de impressões digitais em locais públicos) (Ives, Walsh & Schneider, 2004).

Apesar de uma série de considerações que devem ser realizadas, como as listadas em (Brostoff & Sasse, 2000), diversos métodos biométricos e chaves de hardware estão em pleno funcionamento. Em conjunto com a senha, esses elementos formam algo que *o usuário é* (biometria), algo que *o usuário tem* (*token* físico ou *smartcard*) e algo que *o usuário sabe* (senha). Unidos, esses três fatores formam um método de autenticação extremamente robusto.

O *Google* já disponibiliza uma opção de autenticação por dois fatores<sup>2</sup>, utilizando a senha e um código pseudo-aleatório gerado por meio de um aplicativo para *smartphones*. (Lach, 2010) apresenta esse conceito em mais detalhes.

Verifica-se ainda a disponibilização cada vez mais comum de *login* único (ou *single sign-on*) por empresas com uma ampla base de usuários, como o *Google*, *Microsoft* e *Facebook*. Isso permite que seus usuários utilizem as credenciais (*usuário* e *senha*) que já possuem para efetuar a autenticação em sítios de terceiros. Com isso, o usuário não tem a preocupação de criar uma nova senha específica para aquele sítio. A tecnologia mais utilizada nesses sistemas é o OpenID<sup>3</sup> (Recordon & Reed, 2006).

---

<sup>2</sup> Google 2-step verification

<sup>3</sup> OpenID Foundation: <http://openid.net/>

Entretanto, (Bonneau & Preibusch, 2010) observam que métodos de autenticação centralizada não são bem aceitos em alguns sítios, pois eles perdem o pretexto que tinham para coletar informações pessoais do usuário (e utiliza-las posteriormente em campanhas de marketing, por exemplo). Além disso, os pesquisadores afirmam que essa técnica diminui a capacidade do sítio de estabelecer uma relação de confiança com seu usuário.

Observa-se também o surgimento e aprimoramento de outras técnicas, como senhas gráficas (Jermyn *et al.*, 1999; Thorpe & Oorschot, 2004; Lin *et al.*, 2007; Komanduri & Hutchings, 2008; Chiasson *et al.*, 2008); senhas que utilizam a movimentação dos olhos (De Luca, Denzel & Hussmann, 2009; Kumar *et al.*, 2007); senhas baseadas no desenho de traços (Weiss & De Luca, 2008); mnemônicos reutilizáveis (Topkara, Atallah & Topkara, 2007); senhas tolerantes a erros (Frykholm & Juels, 2001); senhas quânticas (Weedbrook & Gu, 2005) e diversas outras.

Apesar dos métodos alternativos existentes, a utilização de senhas simples continuará, por algum tempo, sendo fundamental para verificar se uma pessoa é realmente quem diz ser (Hart, 2008). Em função disso, algumas soluções têm sido empregadas visando tornar mais robusto o par *usuário-senha*.

Um exemplo, proposto por (O’Gorman, Bagga & Bentley, 2005), é a utilização de perguntas dirigidas onde as respostas são conhecidas (e não memorizadas) pelo usuário. As respostas devem ser facilmente recordáveis pelos usuários, mas difíceis de serem deduzidas por impostores.

Entretanto, (Schechter, Brush & Egelman, 2009) e (Bonneau, Just & Matthews, 2010) apontam sérias limitações e fragilidades para métodos desse tipo (que também são utilizados para redefinir uma senha esquecida). Segundo suas pesquisas, além das chamadas “perguntas secretas” serem muito simples e de fácil dedução para um atacante ou uma pessoa próxima ao usuário, a maioria das “respostas secretas” são nomes próprios que, dependendo da origem ou etnia do usuário, podem ser estatisticamente deduzidos.

(Pinkas & Sander, 2002) e (Stubblebine & Oorschot, 2004) propuseram – em conjunto com as tradicionais senhas – a utilização de “Testes de Turing Reversos” (ou CAPTCHAs). Esses desafios precisam satisfazer às seguintes condições: serem gerados de forma automática; serem fáceis para seres humanos e difíceis para máquinas; e apresentarem baixa probabilidade de serem adivinhados. Com esse método, que já é



empregado em uma série de locais, pode-se evitar ataques de dicionários automatizados sem interferir muito na usabilidade do sistema.

Já, de acordo com (Vu *et al.*, 2007), os métodos para melhorar a segurança no uso de senhas podem ser divididos em apenas três iniciativas:

1. Aumentar a qualidade da senha inicialmente gerada, como a utilização de políticas de senhas, troca periódica e educação do usuário;
2. Aumentar a capacidade de recordação das senhas geradas, como a utilização de senhas gráficas e mnemônicos; e
3. Aumentar a complexidade do método utilizado para armazenar as senhas, como a utilização de funções de *hash* mais resistentes.

Entretanto, como já abordado neste capítulo, essas soluções não se provaram amplamente aceitas (Forget *et al.*, 2007).

(Hayashi & Hong, 2011) mostraram que poucos usuários utilizam alguma técnica para auxiliá-los com suas senhas (como autopreenchimento do navegador ou gerenciadores de senhas). Apesar disso, gerenciadores de senhas se mostram uma alternativa interessante: ao contrário de ajudar a se lembrar de suas senhas, eles simplesmente as escondem dos usuários (Gaw & Felten 2006).

Um gerenciador de senhas<sup>4</sup> é, basicamente, um repositório criptografado do par *usuário-senha* (Yee & Sitaker, 2006). Com isso, o usuário pode armazenar todas as suas senhas em um único local, necessitando memorizar apenas a senha do repositório (Gehring, 2002). Esse repositório pode ser embutido no próprio navegador de Internet, que armazena e preenche automaticamente as credenciais do usuário.

Em geral, gerenciadores de senhas não associados aos navegadores utilizam algoritmos robustos para criptografar o repositório e, além das funcionalidades básicas, também disponibilizam geradores de senhas aleatórias, funções de autopreenchimento e armazenam outras informações necessárias para o processo de autenticação.

Dessa forma, o usuário pode criar senhas aleatórias com quantos caracteres desejar e armazená-las no repositório sem a necessidade de lembrá-las no futuro. Quando precisar de

---

<sup>4</sup> Por exemplo: PasswordSafe (<http://passwordsafe.sourceforge.net>) ou KeePass (<http://keepass.info/>)

suas senhas, e não quiser utilizar o recurso de autopreenchimento, um simples clique copia a senha para área de armazenamento temporária do computador.

Com isso, o usuário precisa se preocupar apenas com uma senha e todas as demais (que podem ser tão complexas quanto desejado) ficam a cargo do gerenciador de senhas. (Komanduri *et al.*, 2011) mostraram que, quando o usuário utiliza alguma ferramenta desse tipo, suas senhas tendem a ser mais seguras.

Uma solução interessante é armazenar o arquivo contendo o repositório criptografado em um serviço de armazenamento na Internet<sup>5</sup>. Dessa forma, todas as modificações no conjunto de senhas estarão sempre sincronizadas. E, como a maioria dessas ferramentas possuem versões para *smartphones*, as senhas podem ser acessadas de qualquer lugar.

Apesar desses métodos paliativos, (St. Clair *et al.*, 2006) sugerem descartar ou alterar radicalmente a autenticação baseada em senhas. (Zhang, Monroe & Reiter, 2010), por sua vez, sugerem seu total abandono. Já (Inglesant & Sasse, 2010) afirmam que o advento da computação na nuvem (que proporciona ataques por força-bruta com capacidades muito altas), pode ser – finalmente – a motivação para a migrarmos para outros métodos de autenticação mais seguros.

Segundo (Bonneau & Preibusch, 2010), esses métodos mais seguros utilizam variações do protocolo *Encrypted Key Exchange*. Dentre eles, o que mais facilmente se aplica à Internet é o protocolo *Secure Remote Password*, que impede satisfatoriamente diversos tipos de ataques. Entretanto, seu emprego ainda não se tornou popular. Está fora do escopo deste trabalho analisar as implicações do emprego em larga escala dessas soluções.

---

<sup>5</sup> Por exemplo: DropBox (<http://www.dropbox.com>) ou SugarSync (<http://www.sugarsync.com>)

### 3 - MÉTODOS DE RECUPERAÇÃO DE SENHAS

Com o conhecimento dos hábitos e fragilidades mais comuns na utilização de senhas, o presente capítulo abordará os principais métodos existentes para recupera-las dentro do contexto pericial. A análise pericial de arquivos cifrados normalmente é realizada pelos setores técnico-científicos das entidades de Segurança Pública no Brasil (entretanto, nada impede que os métodos aqui descritos sejam também utilizados por peritos cíveis e profissionais de segurança da informação).

Nesse contexto, o objetivo é recuperar uma senha específica (de um arquivo cifrado ou de um disco criptografado, por exemplo). Não é objeto de estudo buscar vulnerabilidades ou atacar os algoritmos criptográficos propriamente ditos. A ideia é recuperar as senhas utilizadas para gerar ou proteger as chaves criptográficas. Entretanto, como será visto, em alguns casos é mais produtivo calcular todo o espaço possível de chaves e efetuar a decifragem diretamente, sem a necessidade da senha.

Recuperar a senha de um arquivo ou sistema depende de uma série de fatores e varia de caso para caso. Dependendo da versão utilizada do compactador de arquivos, por exemplo, o método de cifragem pode ser o DES ou o AES. Alguns sistemas de autenticação armazenam a senha do usuário processada com o vulnerável MD4, outros utilizam o SHA com técnicas de *salt* (Blakstad *et al.*, 2009). Em alguns casos, a senha do usuário estará simplesmente codificada em base 64 e armazenada no registro do Windows.

#### 3.1 - RECUPERAÇÃO DIRETA

A recuperação direta de senhas ou arquivos cifrados pode ocorrer de três formas diferentes:

1. *Senha em claro*: o usuário simplesmente escreve suas senhas em um arquivo de texto (ou email) disponível no computador analisado;
2. *Senhas codificadas*: sistemas que armazenam as senhas codificadas em base 64 dentro do registro do Windows, por exemplo;
3. *Vulnerabilidade conhecida*: sistemas que utilizam algoritmos mal implementados ou com falhas conhecidas.

Em função da enorme quantidade de senhas que precisamos lidar diariamente, não é raro encontrarmos as senhas em claro descritas em (1). Como visto anteriormente, a reutilização de senhas é comum e o uso de gerenciadores ainda é tímido.

Diversos softwares gratuitos são capazes de recuperar as senhas descritas em (2). Uma variedade de sistemas está susceptível a falhas desse tipo. Por exemplo: senhas armazenadas em navegadores de Internet, senhas de leitores de email, senhas de mensageiros instantâneos, senhas de protetores de tela, senhas de pontos de acesso *Wi-Fi*, entre outras. Algumas aplicações (incluindo versões antigas do *Microsoft Office*) codificam as senhas do usuário utilizando um simples *XOR* ou cifras de substituição (Casey, 2002).

Já o método descrito em (3) ocorre quando sistemas de proteção são mal desenvolvidos. Existem softwares que cifram a senha do usuário utilizando uma chave conhecida e armazenam o criptograma da senha concatenado ao arquivo cifrado. Nesses casos, a chave utilizada para cifrar a senha é sempre a mesma e a localização da senha criptografada dentro do arquivo é conhecida. Existem ainda sistemas que usam duas chaves associadas a um arquivo cifrado: uma que cifra a senha e outra que efetivamente cifra o arquivo. Nesses casos, é possível sobrescrever as chaves em questão e obter acesso direto ao conteúdo do arquivo (Kuppens, 2010).

### **3.2 - FORÇA BRUTA**

A recuperação por força-bruta é um método de busca exaustiva em que todas as combinações possíveis de determinado grupo de caracteres são testadas. Teoricamente, esse método pode encontrar qualquer senha. Entretanto, por ser muito ineficiente, só funciona de forma satisfatória para senhas curtas.

Se o grupo de caracteres escolhido for apenas letras minúsculas, por exemplo, e uma senha de 5 caracteres for explorada, a primeira tentativa desse método será *aaaaa*, seguido por *aaaab* até finalizar em *zzzzz*.

O número total de possibilidades ( $n$ ) geradas em um ataque de força bruta é dado pela fórmula a seguir:

$$n = \sum_{i=1}^c A^i \quad (3.1)$$

Onde  $A$  é o número de caracteres disponíveis no alfabeto e  $c$  o comprimento máximo da senha.

Por exemplo: em um ataque que utilize todas as 26 letras minúsculas, 26 letras maiúsculas e 10 dígitos e tente senhas de até oito caracteres, serão geradas quase 222 trilhões possibilidades diferentes ( $\sim 2.22 \times 10^{14}$ ).

Existe ainda a possibilidade de ser realizado um ataque de força bruta sobre as chaves do algoritmo criptográfico (e não sobre as senhas de usuário). Entretanto, esse método só é efetivo em determinados cenários, normalmente em aplicações que usam chaves de até 40 bits. Nessas condições, a recuperação da chave é garantida e o arquivo pode ser diretamente decifrado (mas a senha utilizada para cifrar o arquivo não é recuperada) (Kuppens, 2010).

Obviamente, algoritmos criptográficos robustos não são susceptíveis a esse ataque já que podem possuir um espaço de chaves computacionalmente intratável, como por exemplo, a cifra de bloco AES com um espaço de chaves igual a  $2^{256}$ .

### 3.3 - DICIONÁRIO

A recuperação utilizando listas de palavras-chave ou dicionários é bastante eficaz. Nesse método, todas as ocorrências presentes em um determinado conjunto são testadas como possíveis senhas. O ataque em questão baseia-se no fato de muitas pessoas escolherem uma palavra simples do seu cotidiano como senha.

As listas de palavras-chave não se restringem a expressões comuns encontradas em dicionários. Elas podem ser geradas de diversas maneiras e fontes diferentes. Por exemplo:

- a) Relação das senhas mais utilizadas (disponível na Internet);
- b) Relação de nomes comuns de pessoas, cidades ou lugares famosos;
- c) Palavras e expressões de um filme, música ou esporte;
- d) Palavras e expressões contidas em um sítio na Internet;
- e) Todas as expressões alfanuméricas encontradas em um disco rígido;

f) Informações pessoais (nomes de parentes, números de identificação, etc.).

A maioria dos softwares para recuperar senhas disponíveis no mercado já trazem dicionários pré-configurados, como será mostrado no capítulo 4. Além disso, existe uma série de repositórios na Internet contendo essas listagens. Esses repositórios, além de dicionários em vários idiomas, incluem os itens (a), (b) e (c). O item (d) pode ser obtido por meio de métodos automatizados de coleta.

Já os itens (e) e (f) são típicos do ambiente pericial. Utilizar todas as expressões alfanuméricas de um disco rígido para tentar obter a senha de um arquivo é um método bastante utilizado pela perícia. Essas palavras-chave são coletadas de todo o disco rígido, incluindo espaço não alocado, espaço não particionado, conteúdo dos arquivos de paginação e hibernação, registro do Windows, bem como todos demais arquivos de usuário. Nesse caso, a lista de palavras-chave ou dicionário é a relação dessas expressões (Fragkos & Tryfonas, 2007).

O item (f) é conhecido como dicionário biográfico. Nele são listados os nomes, datas, números de identificação, hobbies e expressões de alguma forma relacionadas ao suspeito (incluindo parentes próximos e até animais de estimação). Pode-se ainda utilizar métodos de permutação e combinação entre essas expressões, aumentando o tamanho do dicionário.

Entretanto, se a senha utilizada for uma expressão comum seguida de um símbolo, por exemplo, essa senha deixa de estar contida no dicionário e o método em questão falha. Mas como será visto, esse método evoluiu para prever essas modificações, além de poder ser utilizado de forma híbrida com outras técnicas.

Segundo (Weir, 2010), encontrar e criar dicionários eficazes é um problema não trivial e talvez a melhor fonte sejam senhas previamente expostas ou divulgadas.

### **3.4 - REGRAS DE ALTERAÇÃO**

A utilização de senhas um pouco mais fortes e o emprego de políticas mais restritivas levaram à evolução do ataque de dicionário. A recuperação de senhas com regras de alteração utiliza dicionários como base e, para cada ocorrência presente no dicionário, é

efetuada a substituição ou adição de caracteres de acordo com as regras definidas. As regras de alteração incluem, por exemplo:

- a) *Capitalização*: modificar todas as letras para maiúsculas, alternar entre letras maiúsculas e minúsculas, colocar a primeira letra maiúscula, etc.;
- b) *Substituição*: modificar determinadas letras por símbolos, como *a* por *@*; *s* por *\$* e *l* por *1*;
- c) *Adição*: inclusão de dígitos ou símbolos no início, meio ou fim de cada ocorrência;
- d) *Concatenação*: cada ocorrência é concatenada com outra presente no mesmo dicionário ou em outra listagem qualquer;
- e) *Inversão*: cada ocorrência do dicionário é invertida.

Pode-se ainda mesclar diversas regras de alteração em um mesmo ataque. A tabela 3.1 mostra alguns exemplos de alteração da palavra *sucesso*.

Tabela 3.1: Exemplos de alteração da palavra ‘sucesso’

Capitalização	Substituição	Adição	Concatenação	Inversão
<i>SUCCESSO</i>	<i>\$uce\$\$o</i>	<i>sucesso123</i>	<i>sucessosucesso</i>	<i>ossecus</i>
<i>SuCeSsO</i>	<i>Suc3ssO</i>	<i>@sucesso#</i>	<i>sucessorápido</i>	
<i>Sucesso</i>	<i>\$u(3SSO</i>	<i>1suce2sso3</i>	<i>sucessolonge</i>	

Ressalta-se que o método pode aumentar de maneira significativa o número de senhas testadas. Para cada expressão presente no dicionário utilizado, uma ou mais regras de alteração podem ser utilizadas, aumentando bastante o tempo e esforço para finalizar o ataque. (Dell’Amico, Michiardi & Roudier, 2010) apontam esse efeito como a “Lei dos Rendimentos Decrescentes”: a probabilidade de recuperar uma senha diminui rapidamente com o aumento do dicionário utilizado.

### 3.5 - PROBABILÍSTICOS

Quando métodos baseados em dicionário (direto e alterados) falham, ainda existem alternativas ao ataque exaustivo por força-bruta. Dois métodos de recuperação de senhas probabilísticos serão analisados: Cadeias de Markov e Gramáticas Especializadas.

### 3.5.1 - Cadeias de Markov

(Dell’Amico, Michiardi & Roudier, 2010) apontam que algumas senhas são mais prováveis de serem escolhidas, mesmo não sendo baseadas em expressões comuns. Segundo os pesquisadores, essas senhas normalmente são formadas por vocábulos pronunciáveis ou sequências de teclas próximas em um teclado. Dessa forma, é improvável que senhas criadas por seres humanos sejam uniformemente distribuídas. Cadeias de Markov<sup>6</sup> são utilizadas para prever senhas desse tipo.

Essa técnica assume que pessoas utilizam senhas cuja sequência de caracteres segue o modelo oculto de Markov (ou *Hidden Markov Model* – HMM) (Rabiner, 1989). Isso significa que a probabilidade do  $n$ ésimo caractere de uma senha é uma função dos caracteres anteriores (Marechal, 2007).

No modelo de Markov de primeiro nível, a probabilidade de uma expressão é o produto das probabilidades de cada um de seus caracteres, calculadas em função da frequência em que aparecem no idioma. A probabilidade da ocorrência de um caractere é condicionada apenas pelo caractere imediatamente anterior. Portanto, o modelo de Markov de primeiro nível pode ser expresso da seguinte forma (Narayanan & Shmatikov, 2005):

$$P(x_1x_2 \cdots x_n) = v(x_1) \prod_{i=1}^{n-1} v(x_{i+1}|x_i) \quad (3.2)$$

Onde  $P(x_1x_2 \cdots x_n)$  é a probabilidade de Markov da sequência de caracteres  $x_1x_2 \cdots x_n$  e  $v(x_i)$  é a probabilidade do aparecimento de  $x$  em um determinado idioma. Os valores de cada probabilidade condicional  $v(x_{i+1}|x_i)$  são calculados por meio da análise de frequência de um dicionário grande o suficiente (Marechal, 2007).

Dessa forma, utilizando as probabilidades condicionais, é possível gerar uma sequência de expressões pronunciáveis e que se pareçam com palavras, mas sem nenhum significado. Essas expressões são então utilizadas como possíveis senhas. Cadeias de Markov podem ainda ser utilizadas em conjunto com técnicas de alteração. Atualmente, as ferramentas de recuperação de senhas disponíveis só possuem registros das probabilidades condicionais do idioma inglês (AccessData Corp., 2010). O software *John the Ripper* utiliza cadeias de Markov no modo de operação incremental (mais detalhes no Capítulo 6).

---

<sup>6</sup> Andrey Andreyevich Markov (14 de junho de 1856 à 20 de julho de 1922) foi um matemático russo.



### 3.5.2 - Gramáticas Especializadas

(Glodek, 2008) propôs a utilização de Gramáticas Probabilísticas Especializadas (GPEs) para gerar listas de senhas prováveis. As listas de senhas geradas por esse método são probabilísticas e utilizam padrões de senhas conhecidas e dicionários auxiliares.

O método em questão é inspirado no conceito de Gramáticas Livres de Contexto, que visa modelar a criação de sentenças gramaticalmente corretas em determinado idioma. Sentenças arbitrárias em uma linguagem podem ser criadas por meio da aplicação de regras de produção. Caso sejam associadas probabilidades a essas regras, pode-se criar sentenças com maior chance de aparecerem no idioma analisado, gerando, portanto, uma Gramática Probabilística Livre de Contexto (*Probabilistic Context-Free Grammar – PCFG*) (Glodek, 2008).

Para as GPEs, (Glodek, 2008) emprega uma técnica similar à PCFG, analisando regras de formação de senhas, ao invés de sentenças em um idioma. O método analisa a estrutura de um conjunto de senhas em claro e determina quais dessas estruturas aparecem com mais frequência.

O método de GPEs é composto por duas fases. A primeira fase consiste no treinamento da gramática onde são criadas estatísticas da localização e do número de letras, dígitos e símbolos que formam cada senha do conjunto de treinamento. Esse treinamento é feito utilizando senhas em claro de um conjunto conhecido. A segunda fase consiste na efetiva geração de senhas prováveis, por meio de um dicionário auxiliar. A seguir, as duas fases são descritas com mais detalhes.

- a) *Fase 01*: Dado um conjunto de senhas conhecidas, a primeira etapa do método é transformar cada senha em uma versão simplificada, utilizando os símbolos L, D e S, para letras, dígitos e símbolos, respectivamente. Por exemplo, a senha *minhasenha123##* é alterada para LDS. Esse processo é repetido para todas as senhas do conjunto de treinamento. Com todas as senhas em sua versão simplificada, pode-se determinar estatisticamente como esses três elementos (L/D/S) são combinados para formar uma senha. Em seguida, o número de ocorrências de cada elemento é calculado, formando a estrutura completa de cada senha. Para o exemplo

*minhasenha123##* a estrutura gramatical completa é  $L_{10}D_3S_2$  (10 letras, seguido de 3 dígitos e 2 símbolos).

Como os dígitos normalmente são utilizados em sequência ou representam uma data (e não números aleatórios), eles não são analisados individualmente, mas em grupos. Para cada estrutura ( $D_1, D_2, D_3, \dots, D_n$ ) a frequência de cada ocorrência é registrada. Por exemplo, no conjunto analisado por (Glodek, 2008) a estrutura  $D_3$  mais comum era *123* e a estrutura  $D_1$  mais comum era *1*. Os símbolos são tratados da mesma forma que os dígitos. Nesse ponto, o método de (Glodek, 2008) possui a relação e a frequência das senhas (nas versões simples e completa), bem como a relação e a frequência das estruturas de dígitos e símbolos.

- b) *Fase 02*: A segunda fase é geração propriamente dita das senhas mais prováveis por meio de um dicionário auxiliar. Durante a fase anterior, foram geradas tabelas com as frequências das estruturas, dígitos e símbolos. Utilizando um dicionário (ou lista de expressões) auxiliar, as senhas prováveis são formadas por meio dessas tabelas e do comprimento das palavras do dicionário. Pode-se ainda estabelecer uma probabilidade mínima a ser atingida, ou seja, pode-se descartar senhas improváveis.

Por exemplo: supondo que a estrutura gramatical  $S_1L_4D_2$  seja a de maior probabilidade e que *@* e *01* sejam as ocorrências mais frequentes das subestruturas  $S_1$  e  $D_2$ , a composição final irá utilizar palavras do dicionário compostas por 4 letras, gerando as senhas *@casa01 @bala01 @mesa01* etc.

O dicionário auxiliar utilizado na segunda fase do método é independente das senhas de treinamento utilizadas para criar as estruturas na primeira fase. Com isso, pode-se utilizar dicionários específicos para um caso sob análise.

(Glodek, 2008) apresentou resultados significativos com esse método, em especial quando o dicionário auxiliar era formado por palavras relacionadas ao contexto das senhas a sendo atacadas. Por exemplo, senhas de um sítio de bandas populares podem ser recuperadas mais facilmente por meio de um dicionário auxiliar gerado a partir de letras de músicas.

O método de GPEs também foi utilizado com sucesso por (Weir *et al.*, 2009), como será visto em detalhes no Capítulo 5.

### 3.6 - PRÉ-COMPUTADO

Métodos pré-computados de recuperação de senhas utilizam tabelas previamente geradas contendo criptogramas e seus respectivos textos em claro. Essa técnica é capaz de decifrar e recuperar determinados tipos de senhas por meio de uma simples consulta a essa base de dados.

“Criptoanálise baseada em busca exaustiva requer muito poder computacional e tempo. Quando o mesmo ataque precisa ser repetido diversas vezes, é possível executar previamente a busca exaustiva e armazenar os resultados em memória. Depois que esse processo prévio de computação é efetuado, o ataque pode ser realizado quase que instantaneamente” (Oechslin, 2003).

O resultado desse cálculo prévio é uma tabela indexada contendo a saída de uma função ligada à sua respectiva entrada. Por exemplo, pode ser criada uma tabela contendo pares de todas as combinações de letras minúsculas com até 7 caracteres e seus respectivos *hashes*. Em um segundo momento, por meio de uma simples busca, pode-se encontrar o texto em claro que gerou determinado *hash*.

Porém, a grande quantidade de entradas que precisam ser armazenadas em memória inviabiliza a aplicação direta dessa técnica. A solução desse problema foi proposta por (Hellman, 1980) ao introduzir a técnica de *time-memory trade-off*. Com esse artifício, uma quantidade significativa de memória é economizada ao custo de tempo de processamento. Ou seja, troca-se memória por tempo.

Como consequência, a quantidade de dados que precisam ser armazenados é reduzida, o tempo total do processo aumenta e a taxa de sucesso deixa de ser 100% (Oechslin, 2003).

A redução na quantidade de memória foi conseguida por (Hellman, 1980) utilizando o que ele chamou de *cadeia de hashes*. Por meio de funções de redução sucessivas, apenas os valores iniciais e finais de uma sequência encadeada são armazenados, e os valores intermediários são processados durante a busca. Esse processamento adicional é o responsável pelo aumento do tempo total da técnica (Blakstad *et al.*, 2009).

(Oechslin, 2003) conseguiu aumentar significativamente o desempenho dessa técnica modificando as funções de redução. Ele chamou as cadeias de *hash* modificadas de *Rainbow Chains*, e as tabelas que as contêm de *Rainbow Tables* (Blakstad *et al.*, 2009).

O termo Rainbow Tables (RT) é comumente utilizado como sinônimo de técnicas pré-computadas de recuperação de senhas.

Diversas ferramentas de RT encontram-se disponíveis, inclusive algumas on-line<sup>7</sup>. Porém, dependendo do algoritmo criptográfico utilizado, não é possível obter uma RT eficaz. Soluções comerciais existem para arquivos cifrados com chaves de até 40 bits (como versões antigas do Microsoft Office e Adobe PDF) (Kuppens, 2010). Além disso, existem RT para *hashes* nos formatos MD5, SHA, Ripemd-160, bem como senhas do Windows armazenadas nos formato LM Hash e NTLM (Thing, 2010).

Utilizando RT, (Blakstad *et al.*, 2009) concluem que todas as senhas armazenadas no formato LM Hash (utilizado por versões do Sistema Operacional Windows anteriores ao Vista) podem ser descobertas em, no máximo, poucas horas. (Hanawal & Sundaresan, 2010) explicam em detalhes o funcionamento e as falhas desse método de autenticação.

Melhorias estatísticas (Narayanan & Shmatikov, 2005), modificações nas cadeias de *hash* e nas funções de redução (Thing & Ying, 2009) foram propostas para tornar essa técnica ainda mais eficiente. Além disso, (Thing, 2010) obteve bons resultados utilizando uma abordagem de expansões virtuais para as tradicionais RT.

### 3.7 - HÍBRIDOS

Métodos híbridos são aqueles que utilizam mais de uma das técnicas descritas anteriormente em conjunto. A maioria das ferramentas disponíveis para recuperação de senhas utilizam regras bastante sofisticadas que mesclam diversos métodos.

(Dell'Amico, Michiardi & Roudier, 2010) concluíram que métodos de recuperação diferentes são mais eficazes dependendo do espaço de senhas que o atacante está disposto a explorar. Dessa forma, levando em consideração o espaço de busca, nenhum método de recuperação prevalece sobre os demais: o método de força bruta é eficiente para senhas

---

<sup>7</sup> Por exemplo: <http://md5.rednoize.com>; <http://www.objectif-securite.ch>; <http://www.lmcrack.com>

curtas, o dicionário para senhas fracas, as regras de alteração quando os dicionários não funcionarem e os probabilísticos quando as senhas forem mais complexas. O estudo propõe a utilização de técnicas híbridas e métodos de recuperação em sequência, o que aumenta a probabilidade de sucesso.

Um ataque de força bruta pode utilizar regras probabilísticas para priorizar uma sequência aleatória em detrimento de outra. Todas as possibilidades de senhas continuam sendo testadas, mas em uma ordem diferente. Por exemplo, a combinação *Locw1* pode ser testada antes de *A[@0(*. Ambas as sequências possuem 5 caracteres, mas a primeira é muito mais provável de ser escolhida como senha que a segunda (St. Clair *et al.*, 2006).

Ataques de força bruta podem ainda utilizar tabelas de frequência de caracteres, explorando uma característica típica da linguagem: algumas letras aparecem com mais frequência que outras. E, como usuários raramente criam senhas aleatórias, a frequência de distribuição de letras permanece. Entretanto, pode ser preciso utilizar tabelas especialmente desenvolvidas para o propósito de recuperar senhas excluindo, por exemplo, artigos e preposições (que ocorrem com grande frequência mas não são normalmente utilizados em senhas) (Kuo *et al.*, 2006).

(Narayanan & Shmatikov, 2005) foram capazes de combinar métodos probabilísticos e regras de alteração com a técnica de pré-computação. A técnica criada funciona da seguinte forma: primeiramente o espaço possível de senhas de determinado tamanho é reduzido utilizando filtros baseados em cadeias de Markov (1). Em seguida, o espaço de senhas restante é novamente reduzido utilizando autômatos finitos, simulando a utilização de regras de alteração por seres humanos (2). Todas as sequências de caracteres de determinado tamanho que satisfazem (1) e (2) são enumeradas. Finalmente, a listagem produzida (que contém senhas prováveis), é utilizada para efetuar um ataque pré-computado por meio de técnicas do tipo *time-space trade-off*. Dessa forma, (Narayanan & Shmatikov, 2005) conseguem aumentar a eficiência de uma RT limitando o espaço possível de senhas a apenas aquelas com alta probabilidade de serem criadas por usuários comuns.

O próximo capítulo abordará os dispositivos e aplicativos utilizados na prática para a recuperação de senhas.

## **4 - FERRAMENTAS PARA RECUPERAÇÃO DE SENHAS**

O presente capítulo é dividido nos tópicos “Aceleradores”, “Aplicativos” e “Análise comparativa”. No primeiro item, serão abordados os dispositivos de hardware utilizados para acelerar os métodos de recuperação de senhas já descritos. Como será visto, nem todos esses métodos podem ter seu desempenho melhorado por hardware.

No segundo item, serão detalhados os aplicativos (comerciais e gratuitos) mais utilizados no ambiente pericial para a recuperação de senhas. Esses aplicativos empregam parte dos métodos anteriormente descritos e podem dar suporte total ou parcial às tecnologias de aceleração que serão apresentadas.

O capítulo é finalizado com quadros comparativos relacionando os aplicativos analisados com os algoritmos criptográficos por eles suportados, incluindo os métodos de recuperação disponíveis e o suporte a aceleração por hardware.

Este capítulo foi baseado no trabalho desenvolvido em (Kuppens, 2010).

### **4.1 - ACELERADORES**

Este tópico fará uma breve análise dos quatro dispositivos utilizados para acelerar o processo de recuperação de senhas: Unidades de Processamento Gráfico, Arranjo de Portas Programáveis, Computação em Grade e Computação na Nuvem.

#### **4.1.1 - Unidades de Processamento Gráfico**

Unidades de processamento gráfico (*Graphic Processing Unit* – GPU) são microprocessadores especializados no tratamento de imagens que podem vir embutidos nas placas-mães ou em placas dedicadas. As GPUs realizam o processamento paralelo de operações do tipo ponto flutuante utilizadas em renderização de imagens e em gráficos de três dimensões. Com isso, a unidade central de processamento (*Central Processing Unit* – CPU) não gasta tempo com essa tarefa, o que aumenta o desempenho do sistema como um todo (Owens *et al.*, 2008).

“As GPUs têm se popularizado e vêm se tornando cada vez mais baratas. Além disso, elas estão ficando bastante versáteis, incorporando cada vez mais funções não só relativas a processamento gráfico, mas também de uso geral. Com isso, surgiu o conceito de computação de propósito geral em GPUs ou GPGPU (*General-purpose computing on graphics processing units*)<sup>8</sup>, que vem sendo utilizado nas mais diversas áreas da ciência, inclusive para criptoanálise” (Kuppens, 2010).

A Arquitetura CUDA (*Compute Unified Device Architecture*)<sup>9</sup> da empresa NVidia se destacou como a principal ferramenta para desenvolvimento de soluções em GPUs. Essa arquitetura permite que algumas funções escritas na linguagem C sejam diretamente processadas por GPUs compatíveis. Com isso, um programa especialmente desenvolvido pode utilizar o processamento normal da CPU e, quando precisar operar grandes quantidades de dados, utilizar o processamento em paralelo da GPU (Ryoo *et al.*, 2008).

Segundo a taxonomia de (Flynn, 1972), GPUs enquadram-se na arquitetura SIMD (*Single Instruction, Multiple Data*), ou seja, diferentes processadores operam simultaneamente a mesma instrução em um conjunto diferente de dados. Com isso, GPUs se tornam uma alternativa interessante para ataque a diversos algoritmos e *hashes* criptográficos.

(Kedem & Ishihara, 1999) realizaram uma das primeiras tentativas de ataque por força bruta nas cifras DES e RC4, utilizando uma arquitetura SIMD personalizada.

(Hu, Ma & Huang, 2009) apresentam um método eficiente para melhorar o desempenho na recuperação de senhas de arquivos do tipo RAR (cifrados com AES) utilizando GPUs. Já (Phong *et al.*, 2010) apresentam uma solução para aumentar a velocidade de ataques por dicionário a arquivos ZIP cifrados.

---

<sup>8</sup> GPGPU Discussion: <http://gpgpu.org>

<sup>9</sup> NVIDIA Corporation: <http://developer.nvidia.com/cuda>

Diversos outros estudos têm mostrado o grande potencial de GPUs para criptoanálise. (Zonenberg, 2009), por exemplo, apresenta um sistema para atacar *hashes* MD5 utilizando GPUs em uma rede de computadores.

#### 4.1.2 - Arranjo de Portas Programáveis

Arranjo de portas programáveis (*Field Programmable Gate Array* – FPGA) são dispositivos de hardware que podem ser configuráveis de acordo com o uso desejado. A utilização de FPGAs é bastante ampla, englobando soluções para processamento de sinais, emulação de hardware, criptoanálise e muitas outras.

Um FPGA é um circuito contendo um grande número de elementos lógicos genéricos que podem ser interconectados e configurados para gerarem um circuito arbitrário. Com um *chip* FPGA é possível simular o comportamento de um circuito digital qualquer. Dessa forma, pode-se obter de forma dinâmica um hardware que desempenhe uma função específica qualquer (Herbordt *et al.*, 2007).

Diversas soluções que utilizam FPGAs já foram propostas e desenvolvidas para acelerar o processamento de funções criptográficas (simétricas e assimétricas) e funções de *hash* criptográficos (Wollinger, Guajardo & Paar, 2004).

Em criptoanálise, (Güneysu *et al.*, 2008) e (Güneysu, 2009) desenvolveram um hardware composto de 120 FPGAs, chamado de “COPACOBANA” (*Cost-Optimized Parallel Code Breaker*) capaz de realizar ataques eficientes ao DES e a outros algoritmos criptográficos. Já (Dandass, 2008) mostra a utilização de FPGAs para acelerar a recuperação de senhas em determinados tipos de ataques por dicionário. Além disso, a utilização de FPGAs em conjunto com técnicas de *time-space trade-off*, incluindo RT, foi abordada com sucesso em (Jean-Jacques *et al.*, 2002), (Mentens *et al.* 2006) e (Theoharoulis, Manifavas & Papaefstathiou, 2009).

Uma ferramenta comercial que utiliza FPGAs para criptoanálise é o *Tableau TACC1441 Hardware Accelerator* (TACC)<sup>10</sup> da empresa *Guidance Software Inc.* Esse dispositivo é capaz de realizar ataques de força bruta e dicionário em diversos tipos de arquivos,

---

<sup>10</sup> Guidance Software Inc.: <http://www.tableau.com>



incluindo: ZIP, RAR, Office 2007 e PGP. Além disso, o TACC pode ser paralelizado<sup>11</sup>. Pode-se colocar até quatro dispositivos em um computador e utilizar tantos computadores quanto forem necessários para processar um ou mais arquivos criptografados (Kuppens, 2010).

#### **4.1.3 - Computação em Grade**

A computação em grade (*Grid Computing*) é o emprego de um conjunto de computadores interconectados para a obtenção de um objetivo único. O tempo total de processamento é reduzido distribuindo-se o processamento entre as máquinas participantes (Jacob *et al.*, 2005).

Segundo a taxonomia de (Flynn, 1972), computação em grade enquadra-se na arquitetura MIMD (*Multiple Instruction, Multiple Data*), ou seja, diferentes processadores operam de forma assíncrona e independente em diferentes conjuntos de instruções em um conjunto diferente de dados. Essa arquitetura, que engloba a maioria dos sistemas paralelos existentes, é utilizada em uma grande variedade de aplicações.

Esse modelo de processamento paralelo pode ser subdividido em duas arquiteturas: (1) Memória Compartilhada e (2) Memória Distribuída. Em (1), os processadores compartilham os recursos de uma memória em comum. Em (2), os processadores utilizam suas próprias memórias de forma independente dos demais. A comunicação entre os processadores em (2) é realizada por meio da passagem de mensagens utilizando um hardware especializado ou rede interconectada (Almasi & Gottlieb, 1994).

Segundo (Krauter, Buyya & Maheswaran, 2002), a computação em grade utiliza uma arquitetura de memória distribuída, composta por um conjunto de máquinas fracamente acopladas, heterogêneas e geograficamente dispersas. Os nós se comunicam pela rede (intranet ou Internet) e um deles, denominado mestre (ou servidor), é responsável por enviar a carga de processamento para os demais (os escravos ou clientes).

Criar e gerenciar grades é uma tarefa relativamente barata em comparação a outros modelos de processamento distribuído. Além disso, a escalabilidade é bastante simples.

---

<sup>11</sup> Digital Intelligence: <http://www.digitalintelligence.com/products/rack-a-tacc/>

Com isso, a computação em grade é uma solução atrativa para diversas aplicações, incluindo a criptoanálise (Bengtsson, 2007).

(Candia, 2008) apresenta um sistema distribuído que utiliza passagem de mensagens para acelerar ataques de dicionário e força bruta: “[...] os resultados dos testes realizados mostram que a [computação em grade] oferece uma alternativa simples, eficaz, eficiente e de baixo custo para uso em ambiente forense”.

(Fragkos, Xyno & Blyth, 2005) utilizaram o tempo ocioso de computadores em uma rede para realizar ataques (dicionário, força bruta e híbrido) a senhas de usuários.

Já (Kleinhans, Butts & Sheno, 2009) utilizam uma rede composta por videogames (*Sony PlayStation 3*) para realizar ataques de força bruta, apresentando um bom custo-benefício e boa escalabilidade.

Diversos outros estudos confirmaram que a computação em grade apresenta uma série de vantagens quando aplicada à criptoanálise. Como será visto à frente, diversas soluções comerciais para recuperação de senhas empregam com grande sucesso esse recurso, em especial os softwares Distributed Network Attack (DNA), Elcomsoft Distributed Password Recovery (EDPR) e Passware Kit Forensic (PKF).

#### **4.1.4 - Computação na Nuvem**

A computação na nuvem (*Cloud Computing*) tem recebido muito destaque recentemente. Muitos autores consideram esse modelo como um novo paradigma na computação. A flexibilidade e o alto poder de processamento têm atraído a atenção de diversos setores da academia e da indústria. Como será visto, já existem soluções de criptoanálise que utilizam esse modelo de computação.

Muitas definições surgiram para o conceito de computação na nuvem. (Vaquero *et al.*, 2009) apontam que existe confusão entre os conceitos de computação em grade e computação na nuvem. Esses autores argumentam que a distinção entre esses dois modelos não é clara, pois ambos compartilham características chave: redução dos custos de processamento e aumento em flexibilidade e confiança. Para chegar a um consenso,

(Vaquero *et al.*, 2009) apresentam uma relação com todas as diferenças e semelhanças entre esses dois modelos.

Já (Böhm *et al.*, 2010), fazem uma revisão da literatura existente, enumerando e comparando as definições para computação na nuvem que já foram sugeridas. Segundo os autores, a melhor definição é:

“Computação na nuvem é um modelo de implementação de TI baseado em virtualização, onde os recursos, em termos de infraestrutura, aplicações e dados, são implementados via Internet como um serviço distribuído por um ou vários provedores de serviço. Esses serviços são escaláveis sob demanda e podem ser cobrados pela modalidade *pay-per-use*” (Böhm *et al.*, 2010).

Em função de todas essas características, a computação na nuvem se torna uma alternativa interessante para a criptoanálise. Com ela, é possível obter uma capacidade de processamento tão grande quanto desejado a preços relativamente baixos (se comparado a um *grid* com a mesma capacidade). Essa flexibilidade permite que um analista faça seus ataques sob demanda.

(Roth, 2011) demonstrou com sucesso como acelerar ataques de força bruta, dicionário e pré-computados utilizando computação na nuvem. Por meio do serviço disponível pela empresa *Amazon.com*, o pesquisador desenvolveu um aplicativo chamado *Cloud Cracking Suite* capaz de atacar senhas de redes sem fio do tipo WPA/WPA2-PSK, bem como senhas armazenadas nos formatos SHA1 e MD5. O aplicativo utiliza o poder de processamento dos computadores disponíveis na nuvem, incluindo suas placas GPU (*Cluster GPU instances*), por meio do *Amazon Elastic Compute Cloud (EC2)*<sup>12</sup>, que é um dos serviços de Internet disponíveis na *Amazon Web Services (AWS)*<sup>13</sup>.

O AWS é um conjunto de serviços disponíveis na Internet que formam a plataforma de computação na nuvem da empresa *Amazon.com*. O EC2 é o módulo que disponibiliza

---

<sup>12</sup> Amazon Elastic Compute Cloud: <http://aws.amazon.com/ec2/>

<sup>13</sup> Amazon Web Services: <http://aws.amazon.com/>

computadores para serem alugados. Segundo a *Amazon.com*, o serviço permite controle completo dos computadores alugados e dispõe de uma interface de gerenciamento simples. O usuário pode escolher o Sistema Operacional que desejar e rodar seus próprios aplicativos. A capacidade de processamento pode ser aumentada ou diminuída de acordo com sua utilização e o usuário só paga pelo processamento efetivamente utilizado.

As soluções para recuperação de senhas existentes no mercado podem ser adaptadas para utilizarem o poder de processamento na nuvem. Os softwares DNA e EDPR que nativamente suportam computação distribuída podem ter seus clientes instalados em computadores localizados na nuvem, por exemplo. Já a versão mais recente do PKF possui acesso nativo a computadores disponibilizados na plataforma EC2.

Com isso, é possível aumentar significativamente o poder de processamento dessas ferramentas de forma rápida, escalável e barata, sem a necessidade de adquirir e manter um conjunto dedicado de computadores.

## **4.2 - APLICATIVOS**

Este tópico fará uma breve descrição dos quatro softwares mais utilizados para a recuperação de senhas e decifragem de arquivos no contexto pericial: DNA, EDPR, PKF e JtR.

### **4.2.1 - Distributed Network Attack**

O *Distributed Network Attack* (DNA) é um software comercial para decifragem de arquivos e recuperação de senhas que utiliza computação em grade. Ele é a versão distribuída do software *Password Recovery Toolkit* (PRTK), ambos produzidos pela empresa *AccessData*<sup>14</sup>.

Esses dois softwares possuem suporte aos mesmos algoritmos criptográficos e a única característica que os distingue é a possibilidade de distribuição do processamento. O presente trabalho será focado apenas no DNA.

---

<sup>14</sup> AccessData Corp.: <http://accessdata.com/>

A principal característica do DNA é a utilização do tempo ocioso de computadores para o processamento de arquivos por meio de uma arquitetura cliente-servidor (denominada *worker-supervisor*) (Kuppens, 2010).

O DNA suporta a recuperação de senhas de uma série de aplicações (veja Tabela 4.3 para mais detalhes) e é compatível com diversos sistemas operacionais, incluindo os mais comuns: Windows, Linux e Mac-OS. Além de suportar nativamente a aceleração por computação em grade, o DNA dá suporte também a hardwares do tipo FPGA. Entretanto, o DNA não é compatível com aceleradores baseados em GPU e não suporta nativamente processamento na nuvem.

Quando um arquivo criptografado é adicionado ao DNA, primeiramente sua assinatura é analisada e, se o arquivo for suportado pela ferramenta, o módulo de decifragem adequado é acionado. Dependendo do arquivo sendo processado, o usuário pode escolher qual método de recuperação deseja utilizar. O DNA é compatível (total ou parcialmente) com todos os métodos aqui descritos (recuperação direta, força bruta, dicionário, regras de alteração, probabilísticos, pré-computados e híbridos).

Em seguida, o arquivo (ou parte dele) é distribuído para os clientes selecionados, juntamente com a carga de processamento a ser realizada. Periodicamente, os clientes informam ao supervisor o estado de seu processamento e o alerta quando finalizado. O processamento é finalizado de duas formas: quando o cliente consegue decifrar o arquivo ou quando a carga de trabalho sob sua responsabilidade termina (AccessData, 2010).

O DNA oferece outras funcionalidades interessantes, dando ao perito diversas opções e flexibilidade durante o processo de decifragem. Por exemplo, pode-se gerar dicionários personalizados, criar regras de alteração e definir métodos de recuperação em um ataque híbrido.

Além disso, são apresentadas estatísticas detalhadas dos arquivos em processamento e dos clientes conectados, como número de senhas tentadas por segundo, tempo de disponibilidade, número e capacidade de processamento, entre outras (Kuppens, 2010).

As figuras de 4.1 a 4.4 apresentam, respectivamente, as seguintes telas do DNA: interface de gerenciamento, um exemplo dos métodos de recuperação disponíveis, estatísticas de processamento e estatísticas de senhas tentadas por segundo.

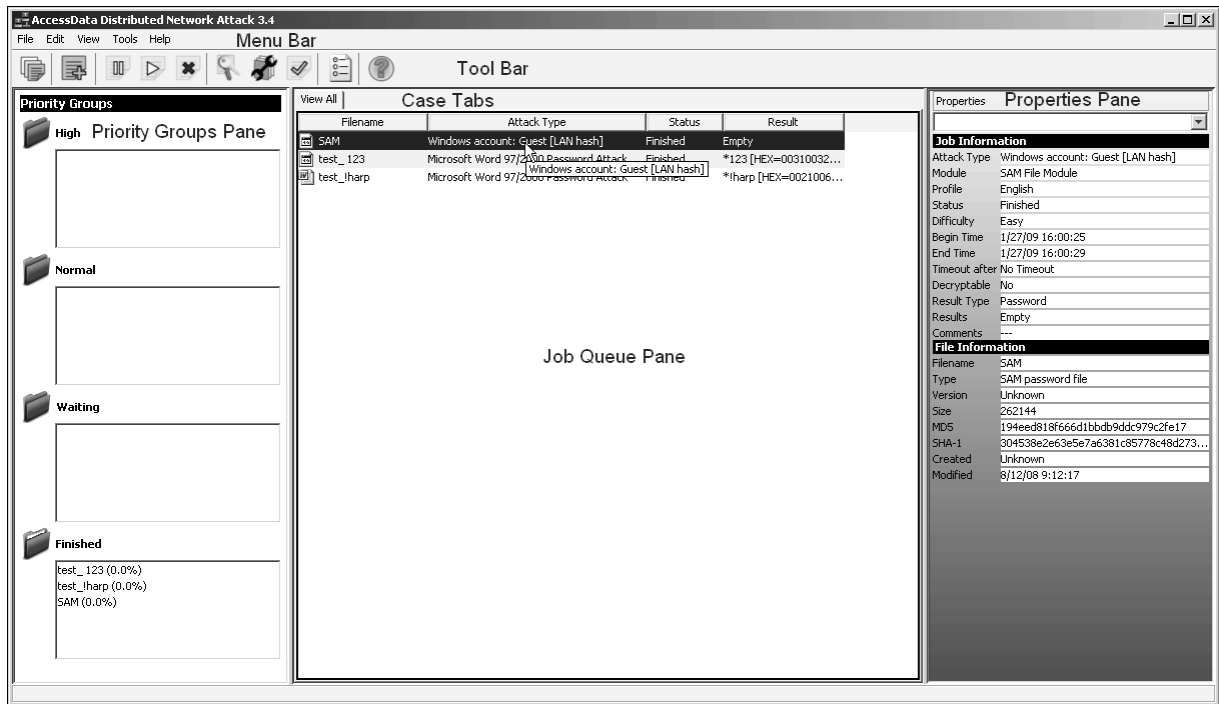


Figura 4.1: Interface de gerenciamento do DNA

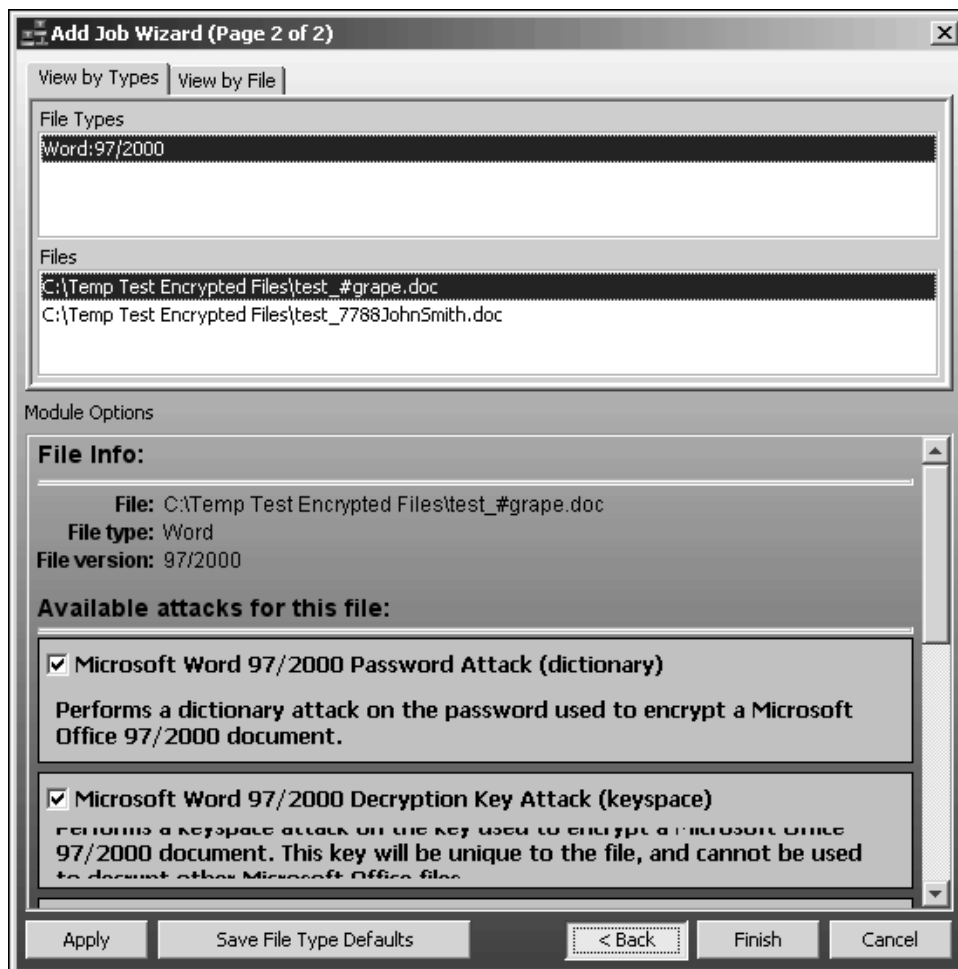


Figura 4.2: Métodos de recuperação disponíveis para um arquivo .doc

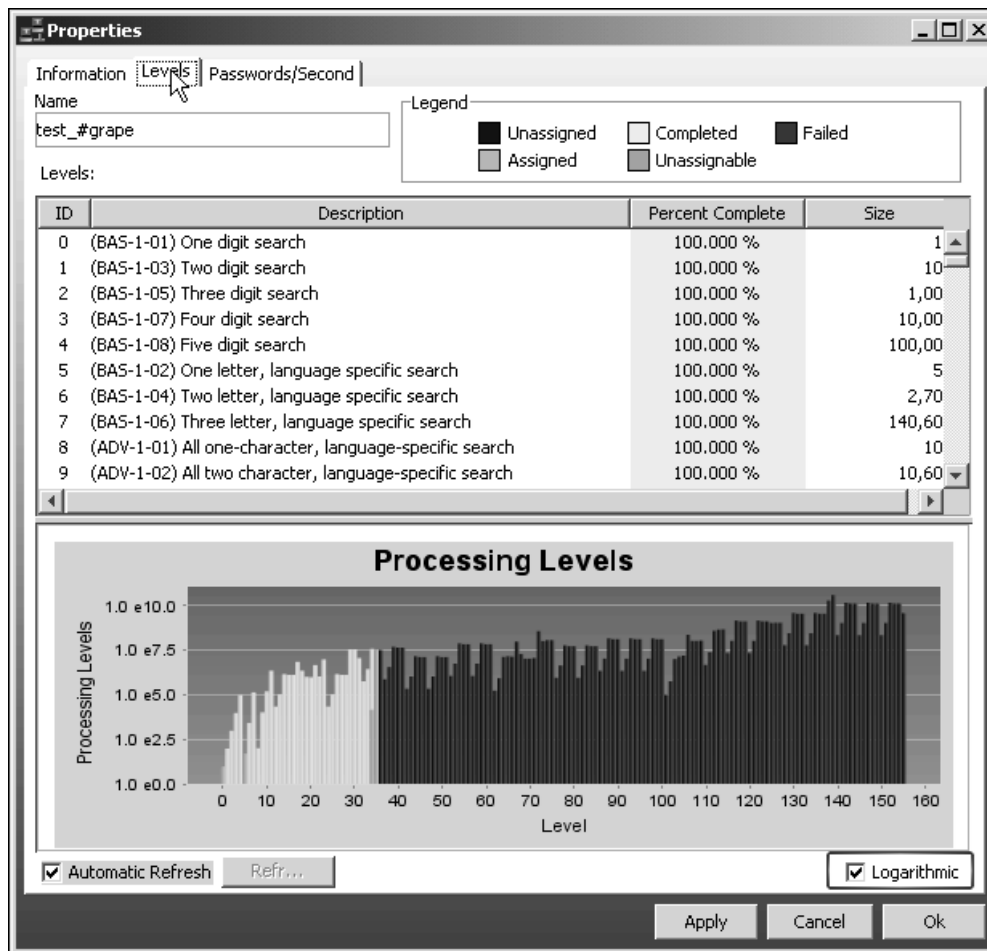


Figura 4.3: Estatísticas de processamento em um arquivo sendo atacado

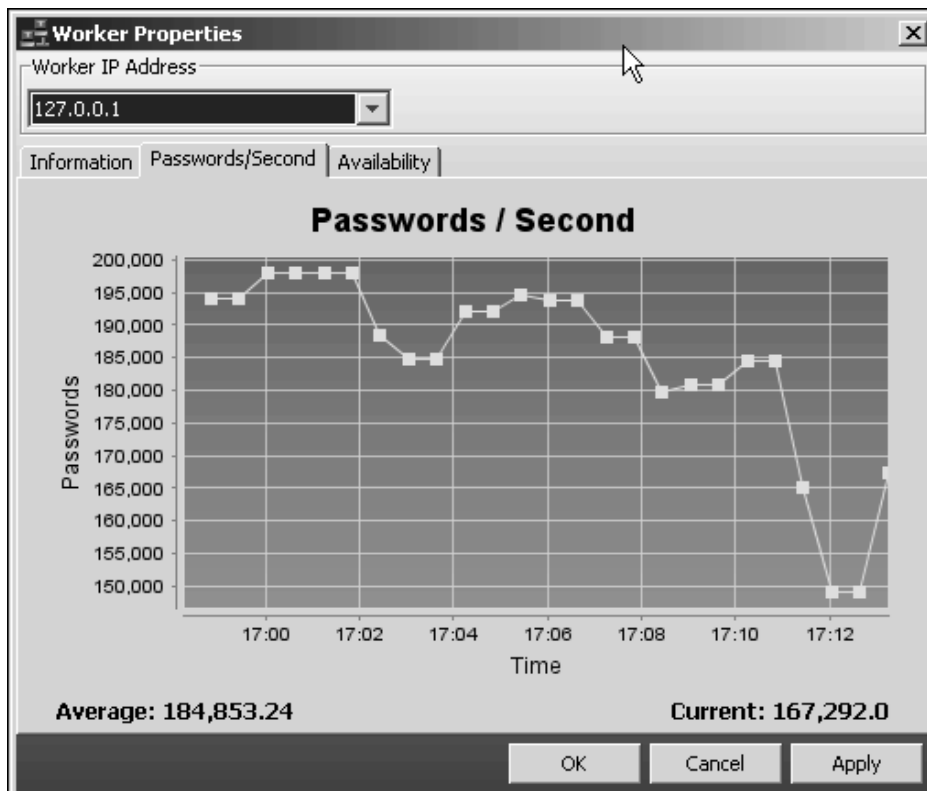


Figura 4.4: Estatísticas de senhas tentadas por segundo de um cliente

#### 4.2.2 - Elcomsoft Distributed Password Recovery

O *Elcomsoft Distributed Password Recovery* (EDPR) é um software comercial para decifragem de arquivos e recuperação de senhas que utiliza computação em grade produzido pela empresa *Elcomsoft*<sup>15</sup>.

O EDPR suporta a recuperação de senhas das aplicações mais comuns (veja Tabela 4.3 para mais detalhes) e é compatível apenas com o sistema operacional Windows. Assim como o DNA, ele distribui a carga de processamento aos computadores ociosos conectados à grade (Kuppens, 2010).

Além de suportar nativamente a aceleração por computação em grade, o EDPR é capaz de utilizar, para alguns algoritmos, o processamento de unidades gráficas (GPUs) e aceleradores de hardware (FPGAs), mas não suporta nativamente processamento na nuvem. A arquitetura utilizada é do tipo cliente-servidor (Elcomsoft, 2011).

O EDPR é composto por três módulos: *console*, *servidor* e *agente*. Esses módulos podem operar de maneira independente em diferentes computadores. O módulo *console* é a interface gráfica responsável pela: criação dos processos de recuperação de arquivos, definição dos métodos de ataque, controle dos agentes e demais tarefas de gerenciamento. O módulo *console* conecta-se ao módulo *servidor* (que pode estar instalado em outro computador). O módulo *servidor* não possui interface gráfica e é responsável por receber as tarefas criadas no *console* e distribuí-las aos *agentes*. O módulo *servidor* conecta-se aos outros dois módulos. Já o módulo *agente* é o responsável pelo processamento propriamente dito. Ele se conecta periodicamente ao *servidor*, informa seu estado, entrega o trabalho realizado (se houver) e recebe mais tarefas (Elcomsoft, 2011).

Os métodos de recuperação de senhas disponíveis no EDPR são: recuperação direta, força bruta e dicionário. A recuperação direta funciona para arquivos de versões antigas do Microsoft Office (97 e 2000) e Adobe PDF (com até 40 bits). No método de força bruta, o usuário pode definir apenas os tamanhos mínimo e máximo das senhas a serem testadas ou uma máscara de tamanho fixo e caracteres curinga. A ferramenta dá suporte também à busca exaustiva pela chave de alguns algoritmos. O método de dicionário é muito restritivo: só existem dicionários em inglês, alemão e russo e a ferramenta não permite a criação ou importação de novos dicionários (Kuppens, 2010).

---

<sup>15</sup> Elcomsoft Co. Ltd.: <http://www.elcomsoft.com/>



As Figuras 4.5 e 4.6 apresentam a interface de gerenciamento (console) do EDPR.

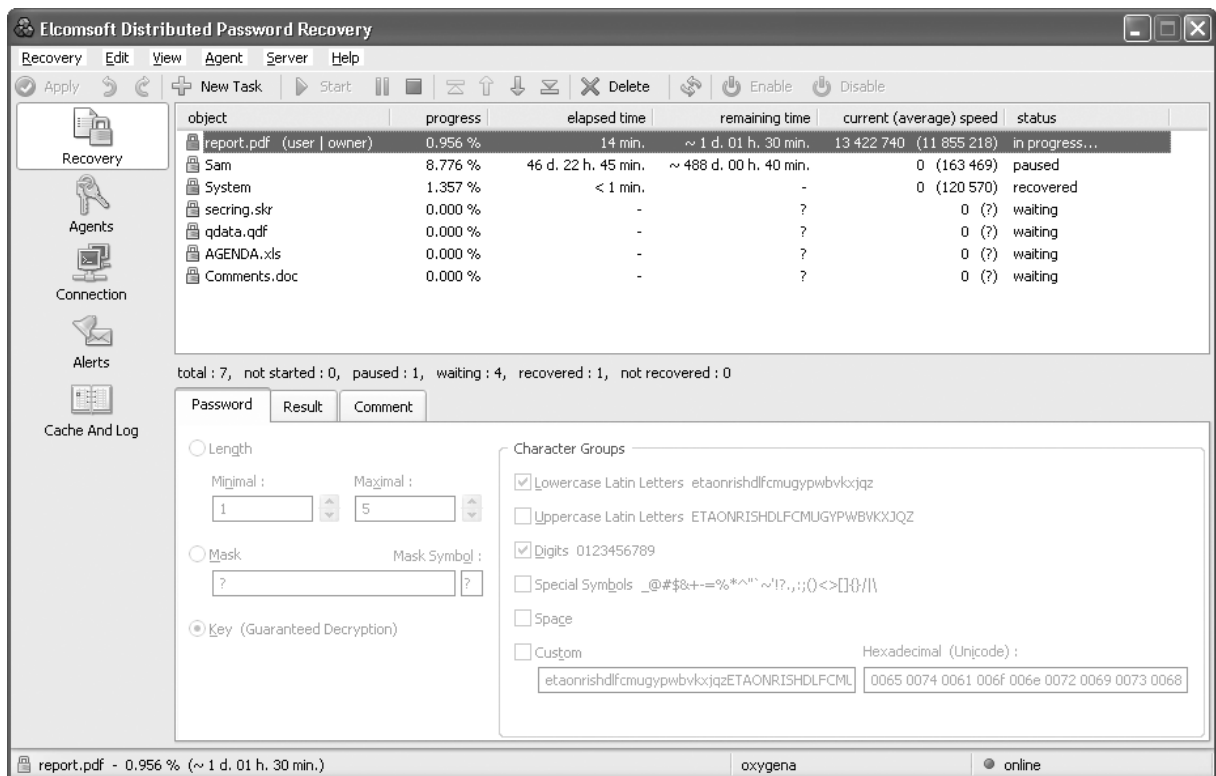


Figura 4.5: Interface de gerenciamento dos arquivos processados no EDPR

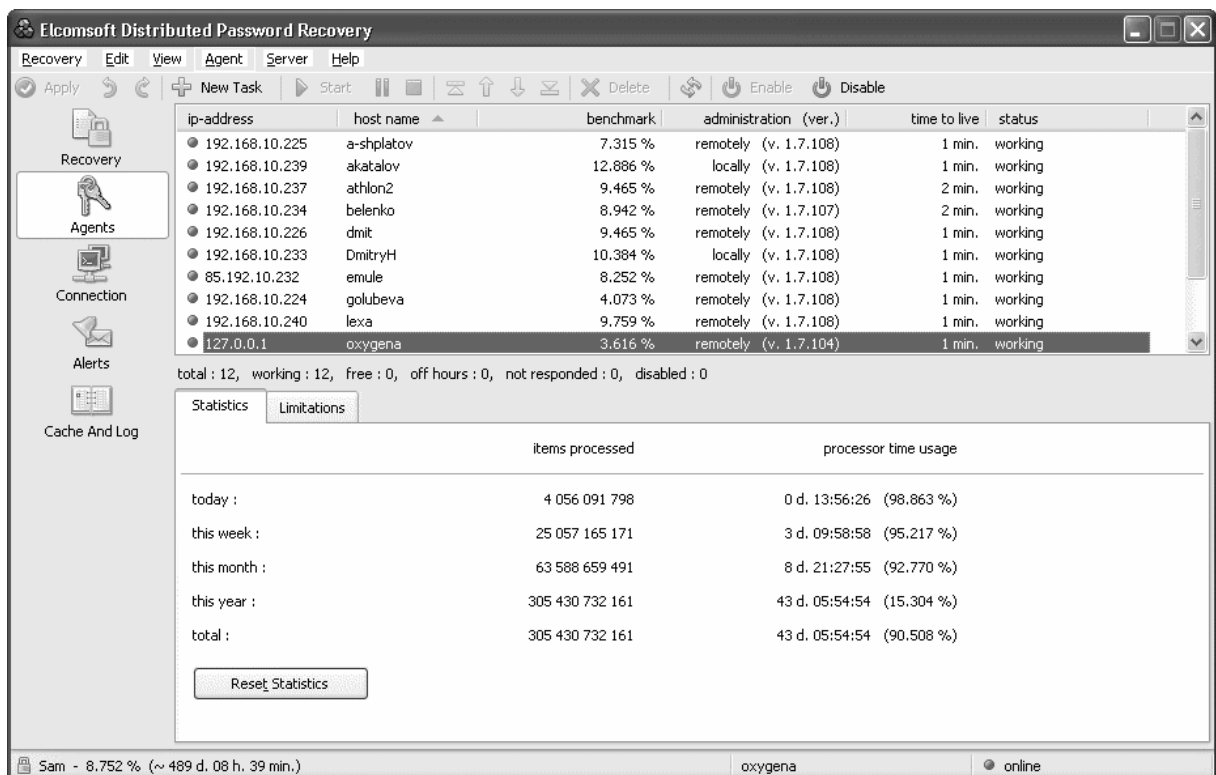


Figura 4.6: Interface de gerenciamento dos agentes no EDPR

### 4.2.3 - Passware Kit Forensic

O *Passware Kit Forensic* (PKF) é um conjunto de aplicativos comerciais produzidos pela empresa *Passware*<sup>16</sup> cujo principal objetivo é o acesso a dados cifrados.

O software possui suporte nativo a todas as ferramentas de aceleração (GPU, FPGA, GRID e CLOUD). O PKF suporta a recuperação de senhas de uma série de aplicações (veja Tabela 4.3 para mais detalhes) e é compatível apenas com o sistema operacional Windows. Além disso, o PKF possui uma versão portátil, que pode ser utilizada em campo por meio de um *pendrive* (Passware, 2011).

O PKF possui ainda suporte a dois mecanismos criptográficos robustos: *TrueCrypt* e *BitLocker*. O PKF é capaz de recuperar a chave criptográfica desses dois sistemas caso seja realizada uma cópia física da memória com os volumes criptografados ainda montados. Caso máquina esteja desligada ou os volumes desmontados, o PKF recorre ao ataque por força bruta. Além disso, o PKF é a única ferramenta que possui suporte nativo ao processamento na nuvem por meio da plataforma EC2 (Passware, 2011).

Os métodos de recuperação de senhas disponíveis no PKF são: recuperação direta, força bruta, dicionário, regras de alteração e pré-computado. A recuperação direta limita-se a alguns sistemas vulneráveis. Para o método de força bruta, o software permite determinar o grupo de símbolos a serem utilizados por meio de uma listagem personalizada de caracteres. Além disso, pode-se delimitar o tamanho das senhas a serem utilizadas e definir determinadas regras de alteração. Existe ainda a possibilidade de excluir do conjunto de teste senhas cuja formação seja uma combinação fora dos padrões usuais. No método de dicionário, a ferramenta disponibiliza palavras em nove idiomas (incluindo português) e permite a inclusão de dicionários personalizados. Esse método permite ainda a criação de padrões de senha, limitação de tamanho, capitalização e busca reversa. O método pré-computado funciona por meio de um serviço online de RT da própria empresa (Passware, 2011).

As Figuras de 4.7 a 4.9 apresentam, respectivamente, a tela inicial do PFK, um exemplo da escolha do método de recuperação e o andamento do processo de recuperação.

---

<sup>16</sup> Passware Inc.: <http://www.lostpassword.com>

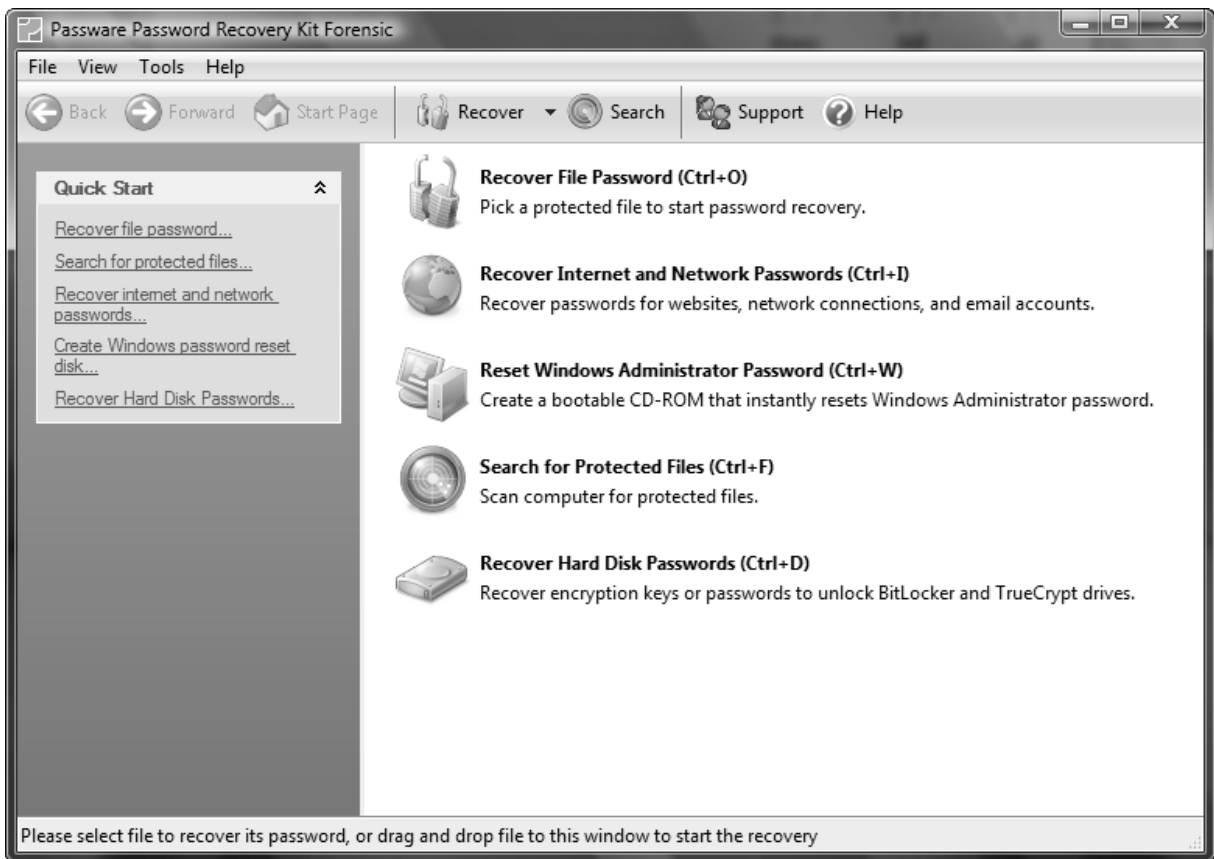


Figura 4.7: Tela inicial do PKF

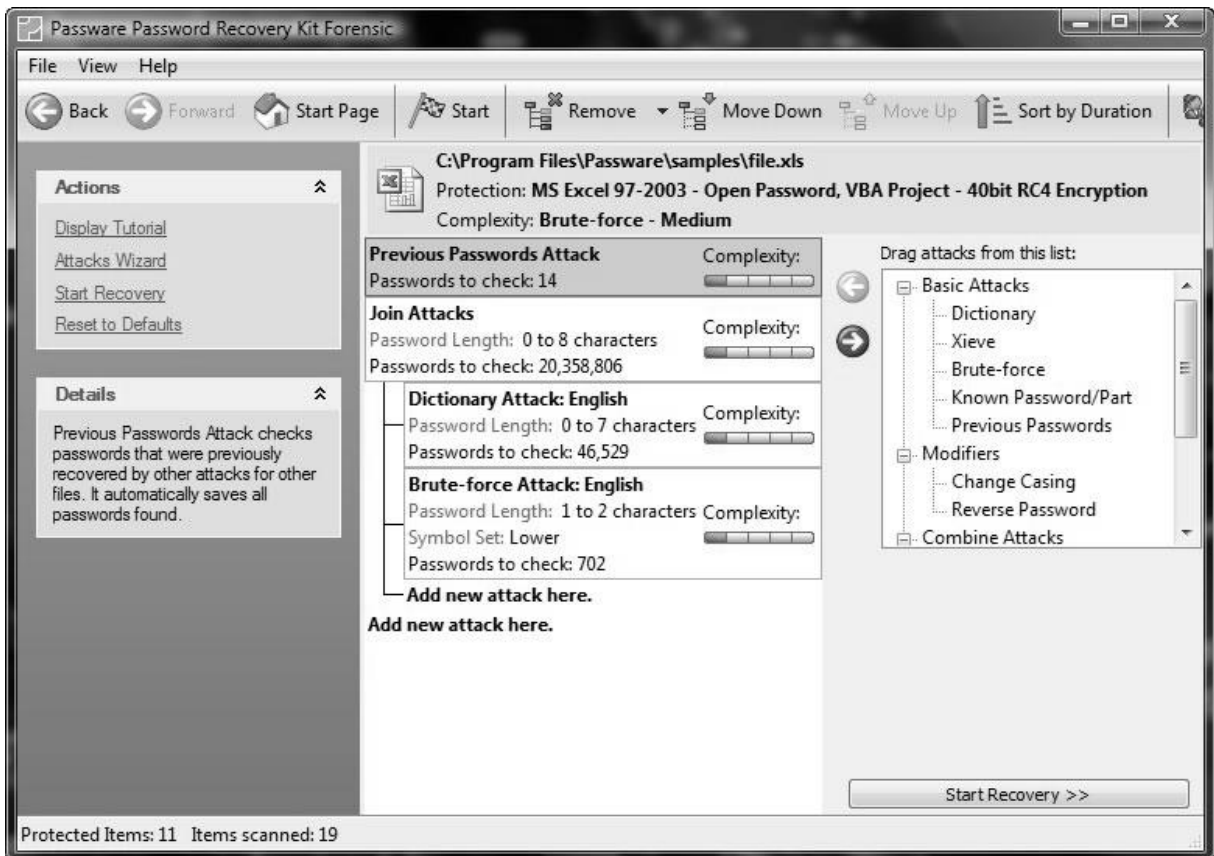


Figura 4.8: Escolha do método de recuperação a ser utilizado no PKF

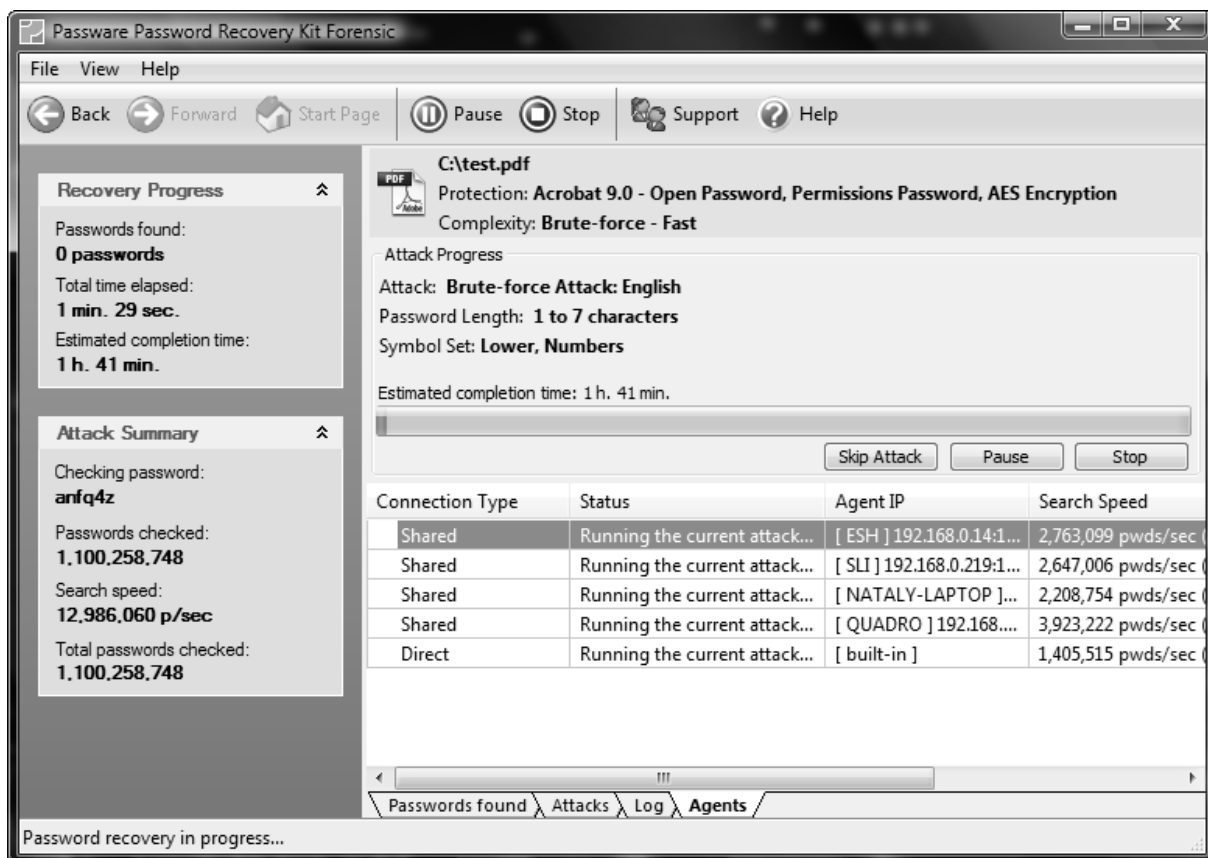


Figura 4.9: Arquivo sendo processado no PKF

#### 4.2.4 - John the Ripper

O *John the Ripper* (JtR) é um software livre e de código fonte aberto utilizado principalmente para a recuperação e auditoria de senhas no ambiente Unix. O JtR é mantido pelo *Openwall Project*<sup>17</sup> e é muito utilizado pela comunidade acadêmica, servindo como parâmetro para testes de novos algoritmos e métodos de recuperação de senhas.

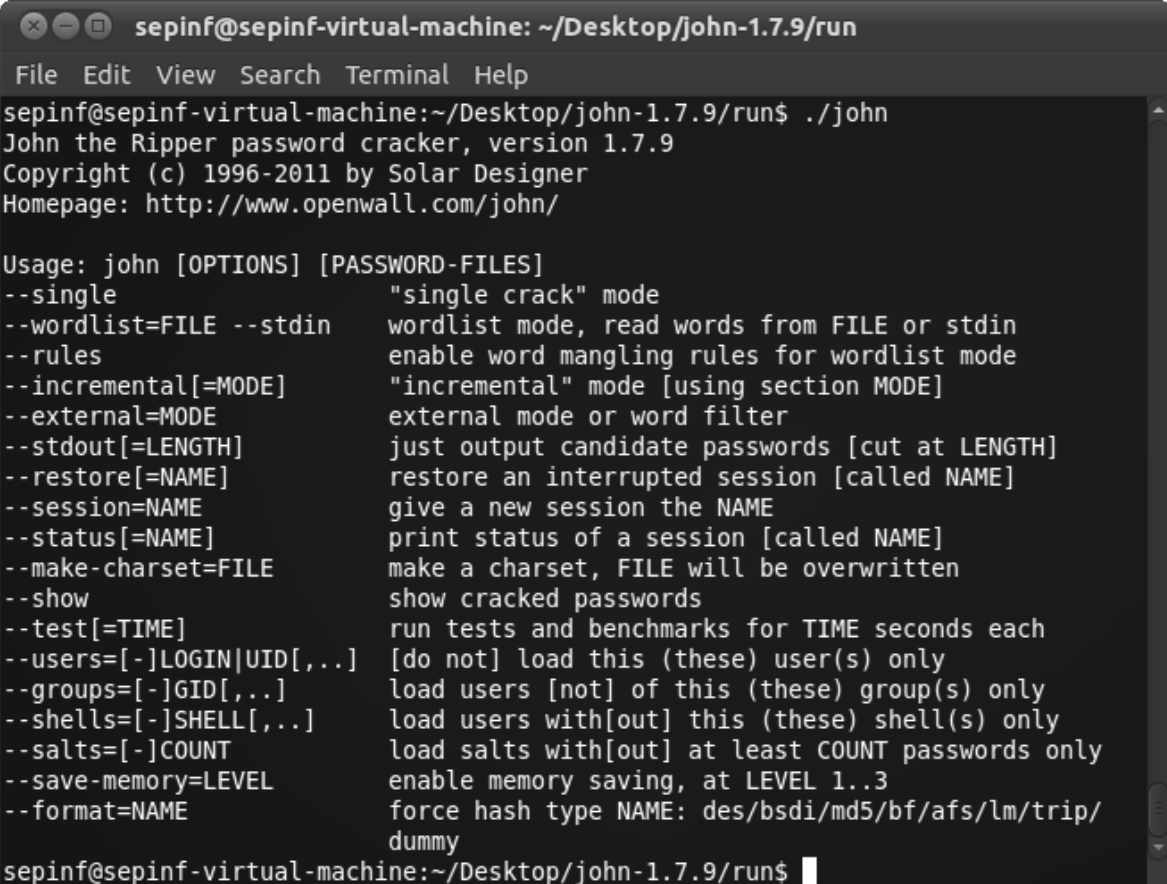
Além de suportar algoritmos de *hash* típicos do Unix (funções `crypt`), o JtR também é compatível com o formato utilizado por versões antigas do Windows (*LM Hash*) e diversos outros algoritmos de *hash*. Utilizando códigos não oficiais desenvolvidos por terceiros, o JtR pode ser utilizado para recuperar senhas de outros aplicativos e sistemas, como PDF, ZIP e RAR (Openwall, 2011).

<sup>17</sup> Openwall project: <http://www.openwall.com/john>

Nativamente, o JtR não suporta nenhuma ferramenta para acelerar o processo de recuperação de senhas. Entretanto, colaboradores independentes já desenvolveram versões paralelas com suporte a computação em grade e GPU.

Basicamente, os métodos de recuperação de senhas disponíveis no JtR são: força bruta, dicionário, regras de alteração, probabilísticos (cadeias de Markov) e híbrido, incluindo diversas opções de personalização. O software, por exemplo, utiliza de forma automática informações da conta sendo atacada (como nome do usuário e números de identificação) para criar um dicionário biográfico e aplicar regras de alteração pré-estabelecidas. O aplicativo permite ainda que o usuário gere suas próprias regras de alteração e crie métodos de ataque utilizado comandos na linguagem C (Openwall, 2011).

O *Openwall Project* mantém listas de dicionários contendo palavras de 21 idiomas diferentes (português não incluso) e senhas mais frequentemente utilizadas. Em conjunto, os dicionários chegam a quase quatro milhões de palavras (Dell'Amico, Michiardi & Roudier, 2010). A Figura 4.10 apresenta as opções disponíveis do JtR (versão 1.7.9). Não existe uma interface gráfica oficial.



```
sepinf@sepinf-virtual-machine: ~/Desktop/john-1.7.9/run
File Edit View Search Terminal Help
sepinf@sepinf-virtual-machine:~/Desktop/john-1.7.9/run$ ./john
John the Ripper password cracker, version 1.7.9
Copyright (c) 1996-2011 by Solar Designer
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single                "single crack" mode
--wordlist=FILE --stdin  wordlist mode, read words from FILE or stdin
--rules                enable word mangling rules for wordlist mode
--incremental[=MODE]    "incremental" mode [using section MODE]
--external=MODE         external mode or word filter
--stdout[=LENGTH]      just output candidate passwords [cut at LENGTH]
--restore[=NAME]        restore an interrupted session [called NAME]
--session=NAME          give a new session the NAME
--status[=NAME]         print status of a session [called NAME]
--make-charset=FILE     make a charset, FILE will be overwritten
--show                  show cracked passwords
--test[=TIME]           run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,..] [do not] load this (these) user(s) only
--groups=[-]GID[,..]    load users [not] of this (these) group(s) only
--shells=[-]SHELL[,..] load users with[out] this (these) shell(s) only
--salts=[-]COUNT      load salts with[out] at least COUNT passwords only
--save-memory=LEVEL     enable memory saving, at LEVEL 1..3
--format=NAME           force hash type NAME: des/bsdi/md5/bf/afs/lm/trip/
                        dummy
sepinf@sepinf-virtual-machine:~/Desktop/john-1.7.9/run$
```

Figura 4.10: Opções disponíveis no JtR

### 4.3 - ANÁLISE COMPARATIVA

Este tópico apresenta uma análise comparativa dos softwares comerciais para recuperação de senhas abordados, incluindo os métodos disponíveis, suporte a aceleração por hardware, algoritmos e sistemas compatíveis.

O software JtR foi excluído da análise por não suportar, em sua versão oficial, nenhuma tecnologia de aceleração e por ser compatível com poucos algoritmos criptográficos.

As tabelas 4.2 e 4.3 foram adaptadas e atualizadas da versão original em (Kuppens, 2010).

A análise foi feita considerando as documentações em (AccessData, 2010), (Elcomsoft, 2011) e (Passware, 2011) e as seguintes versões dos softwares:

- DNA: 3.5.2
- EDPR: 2.96
- PKF: 11.1

#### 4.3.1 - Métodos de Recuperação Disponíveis

Os métodos de recuperação disponíveis pelos aplicativos analisados foram relacionados na Tabela 4.1. A compatibilidade dos métodos apresentados depende do algoritmo ou sistema sendo processado.

Tabela 4.1: Métodos de recuperação suportados pelos aplicativos analisados

	<b>DNA</b>	<b>EDPR</b>	<b>PKF</b>
<b>Recuperação Direta</b>	SIM	SIM	SIM
<b>Força Bruta</b>	SIM	SIM	SIM
<b>Dicionário</b>	SIM	SIM	SIM
<b>Regras de Alteração</b>	SIM	NÃO	SIM
<b>Probabilístico</b>	SIM	NÃO	NÃO
<b>Pré-computado</b>	SIM	NÃO	SIM
<b>Híbrido</b>	SIM	NÃO	NÃO

### 4.3.2 - Aceleradores Suportados

As ferramentas de aceleração suportadas pelos aplicativos analisados foram relacionadas na Tabela 4.2. O suporte depende do algoritmo ou sistema sendo processado.

Tabela 4.2: Ferramentas de aceleração suportadas pelos aplicativos analisados

	DNA	EDPR	PKF
<b>GPU</b>	Não suporta	Para alguns algoritmos	Para alguns algoritmos
<b>FPGA</b>	Para alguns algoritmos	Para alguns algoritmos	Para alguns algoritmos
<b>GRID</b>	Para todos os algoritmos	Para todos os algoritmos	Para alguns algoritmos
<b>CLOUD</b>	Pode ser adaptado	Pode ser adaptado	Suporte nativo

### 4.3.3 - Algoritmos Criptográficos Compatíveis

Os sistemas e algoritmos criptográficos compatíveis pelos aplicativos analisados são apresentados na Tabela 4.3.

Tabela 4.3: Sistemas e algoritmos compatíveis com os aplicativos analisados

Sistema ou Algoritmo	Versões suportadas	Aplicativo		
		DNA	EDPR	PKF
7-Zip	Até 4.65; 9.4 até 9.10	✓		
ABI Coder	3.5.7.4 até 3.6.1.4	✓		
ACT!	Várias versões	✓		✓
Adobe PDF	Várias versões	✓	✓	✓
Advanced File Lock	7.1	✓		
AIM	Várias versões	✓		
A-Lock	Até 7.4	✓		
Ami Pro	-	✓		
AOL (diversos sistemas)	Várias versões	✓		
Apple iTunes	iPhone; iPad		✓	
ARJ	Várias versões	✓		
Ascend	-	✓		
Ashampoo (diversos sistemas)	Várias versões	✓		
BestCrypt	Várias versões	✓		✓
BlackBerry Backup	-		✓	
BulletProof FTP	1.03 até 2.45	✓		
CD-Lock	Até 9.50 5.08 - 7.06	✓		
CheckWriter	5.x	✓		
CodedDrag	2.4	✓		
CruzerLock	1.0 atpe 2.1	✓		
Crypta (diversos sistemas)	Várias versões	✓		
Cryptainer	5 até 7.2	✓		
Cryptext	2.3 até 3.4	✓		
CryptoForge	3.3.0	✓		
CuteFTP	2 até 5; 7.x	✓		

Sistema ou Algoritmo	Versões suportadas	Aplicativo		
		DNA	EDPR	PKF
DataPerfect	-	✓		
dBASE	2.x até 3.x	✓		
DiskCryptor	0.4 até 0.8	✓		
DriveCrypt	Várias versões	✓		
EasyCrypto	5.5	✓		
Encrypted Magic Folders	3.x; 7.x até 9.06	✓		
FileMaker	Várias versões	✓		✓
FileVault	10.3 até 10.5	✓		
Geli	1 até 3; 8	✓		
Gifshuffle	1; 2	✓		
GnuPG	Até 1.4.0	✓		✓
Google Chrome (ProtectedData)	-	✓		✓
HandyBits EasyCrypto Deluxe	5.5	✓		
Hashed Password	Macintosh Hash files	✓		
Hashed Password	MD5 Hashed Password	✓	✓	
Hashed Password	Oracle Hashed Password		✓	
Hashed Password	Unix fcrypt (SHA, Blowfish)	✓	✓	
Hashed Password	Unix htpasswd	✓	✓	
Hashed Password	Unix passwd (MD5, SHA)	✓	✓	
Hashed Password	Windows Active Directory	✓		✓
Hashed Password	Windows LM e NTLM	✓	✓	✓
Hashed Password	Windows PWL Files	✓		
Hashed Password	WPA-PSK Export Hashes		✓	
Hello	1.0	✓		
Hide and Seek	4.0; 5.0; 95	✓		
Icon Lock-IT	XP	✓		
ICQ	Várias versões	✓		✓
Invisible Secrets	4.3 até 4.6	✓		
JPHide	0.51	✓		
Justsystem	Várias versões	✓		
KeePass Password Safe	Várias versões	✓		
KeyChain	OSX	✓		
Kremlin	Várias versões	✓		
LockNote	-	✓		
Lotus (diversos sistemas)	Várias versões	✓	✓	✓
MaxCrypt	1.0 até 1.10	✓		
Microsoft Backup	-	✓		✓
Microsoft Bitlocker	Windows Vista; Windows 7			✓
Microsoft EFS	Windows 2000 até Windows 7	✓		✓
Microsoft Internet Explorer (ProtectedData)	5.0 até 8.0	✓		✓
Microsoft Mail	-	✓		✓
Microsoft Money	Várias versões	✓	✓	✓
Microsoft MSN, Messenger e Live	Várias versões	✓		
Microsoft Office	Várias versões	✓	✓	✓
Microsoft OneNote	2003 até 2007	✓	✓	✓
Microsoft Outlook Express (SMTP)	5.0 até 6.0	✓		✓
Microsoft PFX	P12 Private Key Format	✓		
Microsoft Project	98 até 2007	✓		✓
Microsoft Protected Registry	Até Windows 7	✓		✓
Microsoft Remote Desktop Connections	-			✓
Microsoft Schedule+	7.x	✓		✓
Microsoft Screen Saver	Windows 95	✓		
Microsoft SourceSafe	6.x	✓		
Microsoft SQL	Várias versões		✓	✓
Microsoft Visual Basic for Applications	-	✓		✓



Sistema ou Algoritmo	Versões suportadas	Aplicativo		
		DNA	EDPR	PKF
Microsoft Windows Credential Files	Windows XP	✓	✓	
Mozilla Firefox (ProtectedData)	Várias versões	✓	✓	✓
MYOB (diversos sistemas)	Várias versões	✓		✓
MySQL	5.0 até 5.5	✓		
Netscape (diversos sistemas)	Várias versões	✓		
Norton Backup	-			✓
Norton Ghost	10; 15	✓		
Norton Secret Stuff	1.0	✓		
Omziff	1.0 até 3.0.4	✓		
OpenDocument (OpenOffice, StarOffice, etc)	Várias versões	✓	✓	
Palm Pilot User File	-	✓		
Paradox	Várias versões	✓		✓
Password Pal	Até 2.0	✓		
Password Safe	1 até 3	✓		
PC-Encrypt	Até 9.11	✓		
Peachtree	2002 até 2008			✓
PGP (diversos módulos)	Várias versões	✓	✓	✓
PGP Whole Disk Encryption	9.0	✓	✓	
PKCS #12	-	✓	✓	
PKZIP	Várias versões	✓		
PowerDesk	4; 6; 7	✓		
Pretty Good Envelope	-	✓		
ProWrite	-	✓		
Quattro Pro	Várias versões	✓		✓
Quickbooks	Até 2010	✓		✓
Quicken	Várias versões	✓	✓	✓
RAR	Várias versões	✓		✓
RoboForm	6.8; 6.9	✓		
SafeHouse	Até 3.04; 3.06	✓		
SecureIT	3.1 até 4.0	✓		
SecureZIP	-	✓		
Security Suite	-	✓		
SiFEU File Encryptor	0.9	✓		
Simp	2.2	✓		
Snow	-	✓		
Steganos (diversos sistemas)	Várias versões	✓		
S-Tools	1 até 4	✓		
Super File Encryption	Até 4.0	✓		
Symantec Q&A	4.x até 5.x	✓		
Tally	5.4 até 7.2	✓		
TheBat!	-		✓	
Trillian	Até 4.1	✓		
TrueCrypt (diversos módulos)	Até 6.3a	✓		✓
VersaCheck	Várias versões	✓		
wbStego	2.x; 4.x	✓		
Whisper	Até 1.16	✓		
WinZip	Várias versões	✓		✓
WNStorm	-	✓		
WordPerfect	5.0 até 12; X3	✓		✓
WS FTP	5.0x; 2006	✓		
Yahoo! Messenger	3.0 até 7.0	✓		
Yayoi Kaikei	Até 5.0	✓		

#### 4.3.4 - Resultados

Foram comparados os métodos de recuperação de senhas (recuperação direta, força bruta, dicionário, regras de alteração, probabilísticos, pré-computado e híbridos) e os dispositivos para acelerar esse processo (GPU, FPGA, GRID e CLOUD) implementados, total ou parcialmente, em três ferramentas comerciais utilizadas em laboratórios periciais (DNA, EDPR e PKF). Pode-se constatar que nenhuma dessas ferramentas é completa a ponto de ser a solução única para o tratamento pericial de dados criptografados (Kuppens, 2010).

Verificou-se que o DNA é a ferramenta mais consolidada, compatível com todos os métodos de recuperação disponíveis, além de possuir o maior número de algoritmos e sistemas suportados, todos nativamente distribuídos para a grade. Entretanto, o DNA não suporta a utilização de GPUs.

O EDPR é compatível com os sistemas e algoritmos criptográficos mais comuns. Porém, os métodos de recuperação disponíveis são apenas recuperação direta, força bruta e dicionário. O EDPR suporta aceleração por meio de GPU, FPGA e GRID para alguns algoritmos.

Já o PKF é compatível com quase todos os métodos de recuperação e suporta nativamente todos os aceleradores, incluindo computação na nuvem. Entretanto, poucos sistemas e algoritmos são compatíveis com essas funcionalidades. O PKF é a única ferramenta analisada capaz de processar sistemas criptografados com o *BitLocker* da *Microsoft*.

Todas as três ferramentas suportam o uso do FPGA, em especial o dispositivo *TACC*. Porém, os algoritmos suportados por esse equipamento são poucos e dependem da criação de módulos especializados por parte de cada ferramenta utilizada.

O DNA e o PKF integram-se a bases de dados do tipo RT e são capazes de, quase instantaneamente, decifrar alguns tipos de arquivos cifrados com chaves de até 40 bits.

“Portanto, conclui-se que as ferramentas apresentadas se completam e, dependendo do arquivo a ser processado, uma ou mais deverão ser utilizadas para o acesso tempestivo aos dados. Em uma situação comum onde apenas ataques por força bruta e dicionários são possíveis, pode-

se criar um dicionário personalizado e coloca-lo para processar no DNA e, em paralelo, utilizar o poder das GPUs do EDPR para um ataque por força bruta. Em situações que necessitem a captura de chaves criptográficas da memória, deve-se utilizar o PKF” (Kuppens, 2010).

Apesar de todos esses métodos e ferramentas disponíveis, sistemas modernos e robustos utilizam algoritmos criptográficos susceptíveis apenas a ataques muito lentos (como força bruta, dicionário e regras de alteração). Portanto, a recuperação de senhas e a decifragem desses sistemas dependem da criação de bons dicionários. Para que esses dicionários sejam eficazes, eles devem ser personalizados, ou seja, criados especificamente para o caso em análise.

Bons dicionários personalizados são compostos de dados biográficos, listas de palavras recuperadas de outras mídias apreendidas, idiomas utilizados pelo suspeito, entre outros. Como será visto no próximo capítulo, métodos probabilísticos podem ser adaptados para gerar dicionários mais eficazes.

## 5 - DICIONÁRIO PROBABILÍSTICO PERSONALIZADO

O presente capítulo descreve as fases para a criação de um Dicionário Probabilístico Personalizado por meio de uma Gramática Probabilística Especializada. Para a geração do dicionário são necessários:

- a) Um conjunto de senhas conhecidas; e
- b) Um (ou mais) dicionário(s) auxiliar(es).

Preferencialmente, as senhas conhecidas devem ter a mesma complexidade daquelas sendo recuperadas e o dicionário auxiliar deve conter palavras do mesmo contexto sendo analisado. A criação do dicionário é dividida em duas fases: treinamento da gramática (fase 1) e geração das senhas prováveis (fase 2).

O treinamento da gramática será feito com senhas de usuários brasileiros. Essas senhas serão objeto de análise, incluindo a relação de ocorrências mais comuns, seus tamanhos, taxas de reutilização, emprego de dígitos e símbolos, entre outros.

### 5.1 - OBTENÇÃO DAS SENHAS BRASILEIRAS

Os testes e dicionários gerados no presente trabalho foram baseados em um conjunto de 50.000 senhas brasileiras. Esse conjunto de senhas foi obtido por meio de duas formas:

- a) *Divulgação pública em páginas de compartilhamento de arquivos*: durante a elaboração do levantamento bibliográfico e demais fases iniciais desta obra, a divulgação de listas de senhas por parte de grupos de *crackers* (incluindo o que ficou conhecido como *LulzSec*) foi monitorada e todos os arquivos tornados públicos foram coletados.
- b) *Laudos periciais*: durante a elaboração de laudos periciais de mídias de armazenamento computacional apreendidas no âmbito de fraudes bancárias, é comum a detecção de listas contendo emails e senhas capturadas por meio de *malwares* instalados nas máquinas das vítimas. Foram recuperados os arquivos desse tipo presentes em alguns laudos de operações ocorridas no Brasil entre 2009 e 2010. Os usuários e senhas desse conjunto não foram tornados públicos e não serão expostos nesta obra.

As senhas coletadas referem-se a diversos sítios e sistemas, como emails, páginas de relacionamento, áreas restritas de páginas, entre outros.

Todos esses dados foram analisados e apenas as senhas claramente criadas por usuários brasileiros foram mantidas. Dessa forma, senhas de serviços, sítios ou usuários estrangeiros foram descartadas. Foram eliminados também os registros com dados aparentemente inválidos (de usuários que detectaram que suas informações estavam sendo coletadas indevidamente).

Após a compilação de todos os dados e remoção das entradas duplicadas, foi obtido um total de 64.207 pares *usuário-senha*. Desse conjunto, foi feita uma seleção bastante criteriosa que manteve apenas os registros que satisfizessem as seguintes condições:

- Senhas com seis ou mais caracteres;
- Nomes de usuários únicos (independente do valor da senha);
- Nomes de usuários contendo um endereço de email válido dos seguintes domínios: hotmail.com; gmail.com; globo.com; bol.com.br; msn.com; msn.com.br; live.com; ig.com.br; ibest.com.br; yahoo.com.br; terra.com.br; uol.com.br; globomail.com; oi.com.br; brturbo.com.br; zipmail.com.br; superig.com.br; click21.com.br; r7.com; itelefonica.com.br e pop.com.br.

Assim, foram excluídos registros visivelmente inválidos, senhas muito curtas, senhas geradas por um mesmo usuário e registros de sítios não tipicamente utilizados no Brasil.

Depois da aplicação dessas regras, o conjunto de pares *usuário-senha* baixou de 64.207 para 52.912 registros. Em seguida, foram eliminados (de forma aleatória) outros 2.912 registros, atingindo o número total de 50.000 pares *usuário-senha*.

Dessa forma, procurou-se manter apenas registros válidos de senhas utilizadas no Brasil, evitando quaisquer desvios ou interpretações incorretas nas próximas etapas deste trabalho. Além disso, para preservar a privacidade dos usuários, não serão divulgados os emails coletados e apenas algumas senhas – consideradas de interesse estatístico – serão publicadas nesta obra. Todas as demais serão tratadas como confidenciais. Cabe ainda registrar que não foi realizada nenhuma tentativa de acesso a nenhum tipo de serviço, sítio ou computador utilizando as credenciais obtidas.

## 5.2 - ANÁLISE DAS SENHAS BRASILEIRAS

A análise das 50.000 senhas coletadas mostrou uma taxa de reuso (entre usuários diferentes) de 8%. Ou seja, do total de senhas, 92% eram únicas. O tamanho médio encontrado foi de 9,5 caracteres (foram excluídas senhas com menos de seis caracteres) e a maior senha continha 39 caracteres: “*ame a vida como se nao ouvesse o amanha*”.

A Figura 5.1 apresenta o número de ocorrência de senhas contendo letras, números e símbolos, sendo:

- D: Senhas contendo apenas dígitos, por exemplo: *123456*;
- LD: Senhas contendo letras e dígitos, por exemplo: *abcd123*;
- LDS: Senhas contendo símbolos, por exemplo: *abc123@*;
- L: Senhas contendo apenas letras, dividido em:
  - M/m: Maiúsculas e minúsculas, por exemplo: *aBcDeF* ou *ABCDF*;
  - m: Somente minúsculas, por exemplo: *abcdef*.

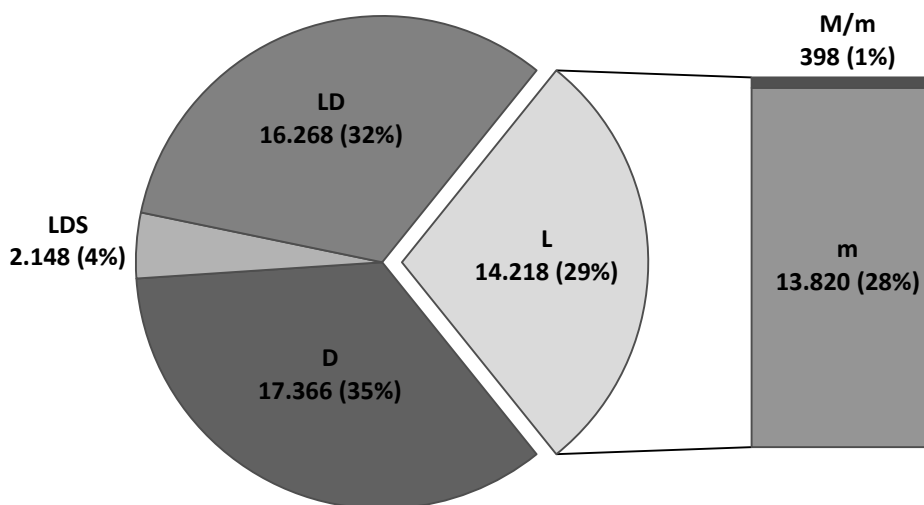


Figura 5.1: Número de ocorrência de senhas com letras, números e símbolos

Observa-se que aproximadamente um terço das senhas contém apenas números, um terço apenas letras minúsculas e um terço letras e números. A participação de senhas com símbolos ou letras maiúsculas é de apenas 5%.

A Tabela 5.1 apresenta as dez senhas mais encontradas e o respectivo número de ocorrências. São apresentadas as senhas mais comuns de: todo o conjunto (coluna

“Geral”), alfabéticas (coluna “Somente Letras”) e alfanuméricas (coluna “Letras e Dígitos”).

Tabela 5.1: Relação das senhas mais utilizadas

	Geral	Número de Ocorrências	Somente Letras	Número de Ocorrências	Letras e Dígitos	Número de Ocorrências
01	123456	474	gatinho	13	123mudar	9
02	123456789	163	brasil	12	jesus123	6
03	102030	67	junior	11	diario123	6
04	101010	29	positivo	10	a1b2c3d4	6
05	654321	27	gostosao	10	diego123	5
06	010203	25	flamengo	10	abc123	5
07	242424	23	fernando	10	1q2w3e	5
08	1234567	23	eduardo	10	jesus1	4
09	12345678	22	ricardo	9	junior24	4
10	123123	18	matheus	9	flamengo2011	4

A senha que mais se repete de todo o conjunto é *123456*, representando aproximadamente 1% do total analisado. Nota-se ainda que as dez senhas mais utilizadas em geral são formadas apenas por dígitos. Além disso, não foram encontradas senhas contendo símbolos que se repetissem mais de duas vezes, como por exemplo a senha *P@sswOrd*.

A Figura 5.2 apresenta o número de ocorrências em relação ao tamanho de cada senha (quantidade de caracteres). Nota-se que mais de 50% das senhas tem entre seis e oito caracteres. Esses resultados são semelhantes a trabalhos anteriores. (St. Clair *et al.*, 2006), por exemplo, afirmam que usuários frequentemente escolhem senhas de seis a oito caracteres sabendo que elas serão universalmente aceitas.

Além disso, 20% do total de senhas analisadas representam uma data válida (nos formatos DDMMAAAA ou DDMMAA) ou um possível número de telefone de oito dígitos (começando com 2, 3, 7, 8 ou 9). Observa-se que (Brown *et al.*, 2004), (Riley, 2006) e (Shay *et al.*, 2010) constataram que os dados mais utilizados em senhas são nomes, seguido de datas de aniversário, números de identidade e números de telefone.

A Figura 5.3 apresenta a proporção de senhas compostas somente por letras (L), somente por dígitos (D), letras e dígitos (LD) e contendo símbolos (LDS) em relação ao seu tamanho. Nota-se que mais da metade das senhas de seis e oito caracteres são formadas apenas por

números. Já as senhas com sete e mais de oito caracteres apresentam proporções similares e uma parcela maior de senhas formadas apenas por letras e letras e dígitos.

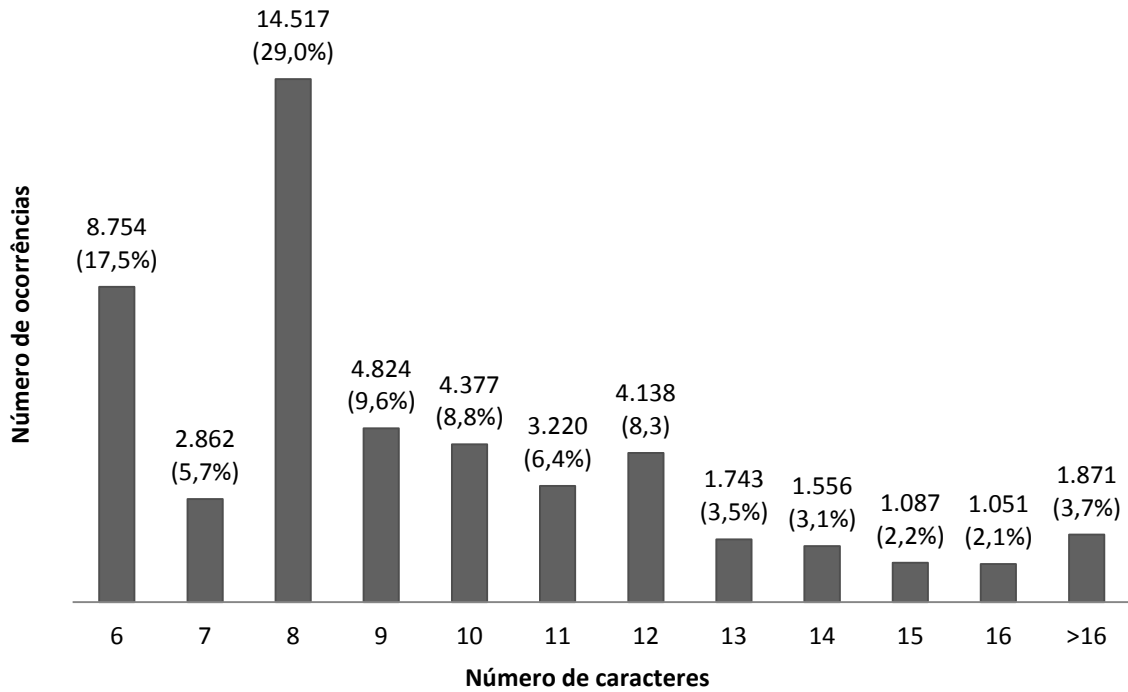


Figura 5.2: Número de senhas encontradas em relação ao seu tamanho

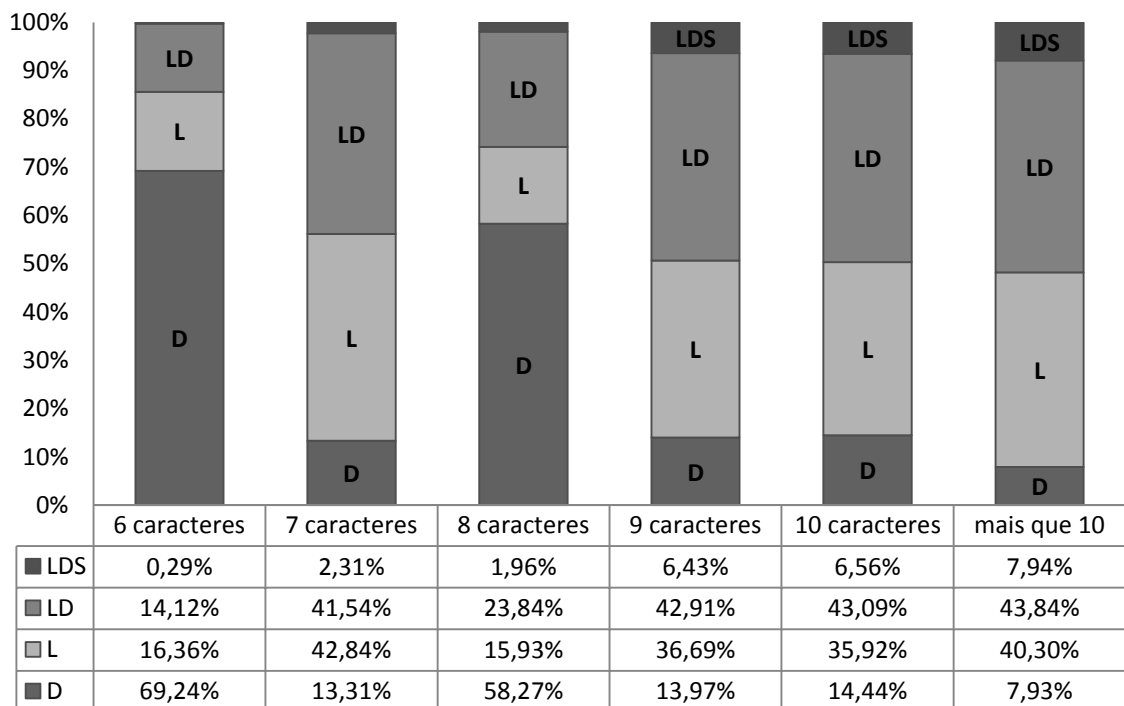


Figura 5.3: Proporção dos caracteres formadores em relação ao tamanho das senhas



### 5.3 - TREINAMENTO DA GRAMÁTICA PROBABILÍSTICA ESPECIALIZADA

O treinamento da GPE foi realizado por meio da ferramenta *Probabilistic Password Cracker*<sup>18</sup> desenvolvida no trabalho de (Weir *et al.*, 2009). Este treinamento consiste na Fase 1 descrita no Capítulo 3 (subitem 3.5.2).

O treinamento é realizado por meio do comando `process.py` e o resultado é a obtenção das estruturas e frequências de formação das senhas (por exemplo: *minhasenha123##* →  $L_{10}D_3S_2$ ), bem como as ocorrências de dígitos e símbolos. Conforme descrito em (Weir *et al.*, 2009), a frequência de dígitos e símbolos são registradas separadamente, independente da estrutura em que aparecem.

Nesta etapa, portanto, três conjuntos de dados são gerados:

- 1) Estrutura gramatical completa e respectivas frequências (veja Tabela 5.2);
- 2) Ocorrências e frequências de conjuntos de dígitos de todos os tamanhos encontrados (veja Tabela 5.3);
- 3) Ocorrências e frequências de conjuntos de símbolos de todos os tamanhos encontrados (veja Tabela 5.4).

O resultado do treinamento da gramática especializada, realizado no conjunto de 50.000 senhas anteriormente descrito, está parcialmente apresentando nas tabelas a seguir.

No total, foram identificadas 2.421 estruturas gramaticais diferentes. As 15 mais frequentes estão listadas na Tabela 5.2. Observa-se que quase 17% de todas as senhas analisadas são compostas por oito dígitos. A segunda estrutura mais comum são senhas de seis dígitos, que ocorrem 12% das vezes. A estrutura composta por letras e dígitos mais frequente só ocorre em 1,5% das senhas analisadas. E a estrutura contendo símbolos mais frequente só ocorre em 0,07% das vezes.

Foram encontradas ocorrências de conjuntos de 1 a 28 dígitos e de 1 a 7 símbolos. As Tabelas 5.3 e 5.4 apresentam as 10 primeiras ocorrências de dígitos e símbolos de 1 a 5 caracteres. Observa-se, por exemplo, que a sequência *123* ocorre em 41% das senhas que contém um conjunto de três dígitos (estrutura  $D_3$ ). Além disso, espaço e ponto estão presentes em mais de 40% das senhas que contém um símbolo (estrutura  $S_1$ ).

---

<sup>18</sup> [https://sites.google.com/site/reusablesec/Home/password-cracking-tools/probablistic\\_cracker](https://sites.google.com/site/reusablesec/Home/password-cracking-tools/probablistic_cracker)

Assim, uma estrutura completa da forma  $L_3S_1D_3S_1L_4$ , por exemplo, tem grande possibilidade de ser: LLL [espaço] 123 . LLLL. As letras que irão compor a senha final serão definidas na segunda fase do processo, que consiste na geração de senhas prováveis por meio de um dicionário auxiliar.

Tabela 5.2: Relação das 15 estruturas gramaticas completas mais comuns

	<b>Estrutura Gramatical Completa</b>	<b>Estrutura Simplificada</b>	<b>Percentual Encontrado</b>	<b>Número de Ocorrências</b>	<b>Tamanho da Estrutura</b>
01	DDDDDDDD	D <sub>8</sub>	16,92%	8.460	08
02	DDDDDD	D <sub>6</sub>	12,12%	6.062	06
03	LLLLLLLLL	L <sub>8</sub>	4,63%	2.313	08
04	LLLLLLLLL	L <sub>9</sub>	3,54%	1.769	09
05	LLLLLLLLLLLLL	L <sub>12</sub>	3,43%	1.716	12
06	LLLLLLLLLLLLL	L <sub>10</sub>	3,14%	1.571	10
07	LLLLLL	L <sub>6</sub>	2,86%	1.432	06
08	LLLLLLLLLLLLL	L <sub>11</sub>	2,66%	1.329	11
09	LLLLLLL	L <sub>7</sub>	2,45%	1.226	07
10	LLLLLLLLLLLLLLL	L <sub>13</sub>	1,52%	758	13
11	LLLLDDDD	L <sub>4</sub> D <sub>4</sub>	1,50%	751	08
12	LLLLLLDD	L <sub>6</sub> D <sub>2</sub>	1,49%	747	08
13	DDDDDDDD	D <sub>9</sub>	1,35%	673	09
14	LLLLLLLLLLLLLLL	L <sub>14</sub>	1,30%	649	14
15	DDDDDDDDDD	D <sub>10</sub>	1,26%	632	10

Tabela 5.3: Relação das 10 estruturas mais frequentes do tipo D<sub>1</sub>, D<sub>2</sub>, D<sub>3</sub>, D<sub>4</sub> e D<sub>5</sub>

<b>1 Dígito (D<sub>1</sub>)</b>		<b>2 Dígitos (D<sub>2</sub>)</b>		<b>3 Dígitos (D<sub>3</sub>)</b>		<b>4 Dígitos (D<sub>4</sub>)</b>		<b>5 Dígitos (D<sub>5</sub>)</b>	
<b>1</b>	25,32%	<b>10</b>	7,76%	<b>123</b>	40,59%	<b>2010</b>	6,07%	<b>12345</b>	21,63%
<b>2</b>	21,88%	<b>12</b>	5,57%	<b>100</b>	4,69%	<b>1234</b>	4,49%	<b>54321</b>	0,86%
<b>3</b>	12,14%	<b>22</b>	3,55%	<b>157</b>	3,95%	<b>2011</b>	3,90%	<b>12369</b>	0,86%
<b>4</b>	8,29%	<b>13</b>	3,18%	<b>007</b>	1,99%	<b>2009</b>	2,75%	<b>32364</b>	0,64%
<b>9</b>	6,33%	<b>11</b>	3,05%	<b>321</b>	1,72%	<b>2008</b>	2,75%	<b>20111</b>	0,64%
<b>5</b>	6,03%	<b>01</b>	3,00%	<b>456</b>	1,68%	<b>2007</b>	1,48%	<b>20010</b>	0,64%
<b>7</b>	5,33%	<b>20</b>	2,87%	<b>789</b>	1,17%	<b>1995</b>	1,25%	<b>14789</b>	0,64%
<b>8</b>	5,29%	<b>15</b>	2,83%	<b>200</b>	1,13%	<b>1993</b>	1,15%	<b>99898</b>	0,43%
<b>0</b>	5,15%	<b>16</b>	2,57%	<b>666</b>	1,10%	<b>1994</b>	1,11%	<b>92584</b>	0,43%
<b>6</b>	4,22%	<b>21</b>	2,52%	<b>159</b>	1,02%	<b>1996</b>	0,92%	<b>88866</b>	0,43%

Tabela 5.4: Relação das 10 estruturas mais frequentes do tipo  $S_1$ ,  $S_2$ ,  $S_3$ ,  $S_4$  e  $S_5$

1 Símbolo ( $S_1$ )		2 Símbolos ( $S_2$ )		3 Símbolos ( $S_3$ )		4 Símbolos ( $S_4$ )		5 Símbolos ( $S_5$ )	
<i>espaço</i>	23,24%	**	22,58%	***	22,02%	****	14,29%	*****	25,00%
.	21,82%	..	12,90%	...	20,18%	....	5,71%	%%%%%%%%	25,00%
*	9,64%	@@	7,26%	*_*	11,93%	...!	5,71%	=^.^=	12,50%
@	8,72%	.,	4,44%	!@#	4,59%	*_*	5,71%	/*-+	12,50%
/	6,47%	@#	4,03%	@@@	3,67%	*-+	5,71%	***/-	12,50%
_	6,05%	//	4,03%	/*-	3,67%	!@#	5,71%	!@#\$\$%	12,50%
-	5,01%	+-	3,63%	.+-	2,75%	@@!_	2,86%		
+	3,34%	--	3,23%	*-+	2,75%	@#..	2,86%		
#	2,88%	.,	3,23%	!!!	2,75%	?...	2,86%		
,	2,13%	/*	2,42%	*+-	1,83%	//**	2,86%		

Para obtermos um parâmetro de comparação nos testes do próximo capítulo, foi realizado o treinamento de uma segunda GPE utilizando o arquivo de senhas<sup>19</sup> que acompanha o JtR. Esse arquivo é formado por um total de 3.157 senhas, sendo 3.017 senhas únicas.

As dez primeiras senhas desse conjunto são: *12345*; *abc123*; *password*; *computer*; *123456*; *tigger*; *1234*; *a1b2c3*; *qwerty* e *123*.

Segundo a documentação do JtR, esse conjunto é baseado nas senhas mais comuns encontradas em ambientes *Unix* e inclui também as senhas mais frequentes que foram tornadas públicas de sistemas comprometidos entre 2006 e 2009.

Essa segunda GPE treinada com senhas genéricas servirá apenas para referência nos testes a serem realizados.

## 5.4 - GERAÇÃO DO DICIONÁRIO PROBABILÍSTICO PERSONALIZADO

No treinamento da gramática especializada, descrito no tópico anterior, foram calculadas as estruturas gramaticais de cada senha e suas frequências. Na próxima etapa serão geradas senhas prováveis por meio de dicionários personalizados, dando origem aos DPPs. Esse processo consiste na Fase 2 descrita no Capítulo 3 (subitem 3.5.2).

<sup>19</sup> Arquivo `password.lst`

Com a estrutura gramatical completa e ocorrências de dígitos e símbolos (listados parcialmente nas Tabelas de 5.2 a 5.4), pode-se gerar as sugestões de senhas em ordem decrescente de probabilidade.

Mas quando a estrutura gramatical completa é extensa, não é trivial calcular essa sequência decrescente. O método proposto precisa estabelecer qual estrutura gramatical é mais provável. Por exemplo, dadas as ocorrências e probabilidades dos elementos  $L_3$ ,  $D_3$ ,  $D_5$  e  $S_2$ , a sugestão final do tipo  $L_3D_3$  deve ser listada antes ou depois da sugestão  $D_5S_2$ ?

Para estabelecer essa sequência, a probabilidade de todas as estruturas existentes poderiam ser calculadas e posteriormente ordenadas. Entretanto, essa tarefa não é facilmente paralelizável e a quantidade de dados que precisariam ser armazenados para ordenação (antes mesmo de uma única sugestão ser gerada) é muito grande (Weir *et al.*, 2009).

A solução proposta por (Weir *et al.*, 2009) foi a utilização de uma fila de prioridades, parcialmente descrita a seguir. A fila é criada com o elemento que contém a estrutura gramatical (antes da utilização do dicionário auxiliar) mais provável no topo. Em seguida, a fila é preenchida com todas as demais estruturas gramaticais, ordenada pela probabilidade mais alta dos elementos  $D_x$  e  $S_y$  que compõe cada estrutura. Depois, novos elementos descendentes (com a segunda maior probabilidade de  $D_x$  e  $S_y$ ) são inseridos e o processo prossegue até que todas as estruturas gramaticais possíveis preencham a fila na ordem correta.

(Weir *et al.*, 2009) provaram que o algoritmo de ordenação descrito é correto, ou seja, ele sempre termina e as estruturas gramaticais na fila sempre estão em ordem não crescente de probabilidade. Os pesquisadores apresentaram também o pseudocódigo do algoritmo em questão.

Com a gramática ordenada, pode-se terminar a segunda fase do processo com a substituição dos elementos alfabéticos pelas palavras presentes em um dicionário auxiliar. A probabilidade desses elementos (letras,  $L$ ) é calculada levando-se em conta somente o dicionário auxiliar. Dessa, forma se o dicionário auxiliar contiver, por exemplo, 5 palavras com 4 letras, a probabilidade de cada elemento  $L_4$  será 20%.

Para que o dicionário final gerado nesse processo seja personalizado, o dicionário auxiliar deve ser criado com dados biográficos do caso em análise, ou seja: *Dicionário Personalizado = Dicionário Especializado com dados biográficos*.

Dicionários Personalizados são específicos para o caso em análise e devem conter:

- a) Dados biográficos do suspeito e seus familiares, abrangendo:
  - Nomes (incluindo de animais de estimação);
  - Endereços (incluindo cidade, CEP e país);
  - Números de telefone, identidade e CPF;
  - Datas relevantes (aniversário, casamento, nascimento dos filhos);
  - Palavras relacionadas a algum hobby ou esporte praticado;
  - Palavras de outros idiomas falados pelo suspeito.
- b) Expressões alfanuméricas recuperadas da(s) mídia(s) apreendida(s).

No contexto pericial, parte dos dados presentes em (a) pode ser facilmente obtida. Outros dados podem necessitar de algum trabalho investigativo.

O DNA contém a ferramenta *Biographical Dictionary Generator* que é bastante útil para a criação de dicionários biográficos a partir de dados pessoais. Trata-se de um software que realiza a permutação de expressões de acordo com sua relevância. Os campos disponíveis nessa ferramenta são: Nome, Endereço, Cidade, CEP, País, Telefone, Data, Número, Palavra e Frase.

Utilizando essa ferramenta e apenas quatro campos preenchidos (Nome: *Isaac*; Nome: *Newton*; País: *Inglaterra* e Data: *25/12/1642*) foram geradas 1.886 permutações, incluindo *isaacnewton*, *newton1642*, *25isaac12*. A imagem da ferramenta *Biographical Dictionary Generator* pode ser vista na Figura 5.4.

Já a listagem relacionada em (b) é obtidas diretamente das ferramentas periciais ou por meio de simples comandos de obtenção de *strings* em um ambiente *Linux*. Essas expressões são coletadas de todo o disco rígido, incluindo espaço não alocado, espaço não particionado, conteúdo dos arquivos de paginação e hibernação, registro do Windows, bem como todos demais arquivos ativos. Obviamente, as expressões em questão só poderão ser obtidas de discos rígidos que não estejam completamente criptografados, o que ocorre em apreensões típicas.

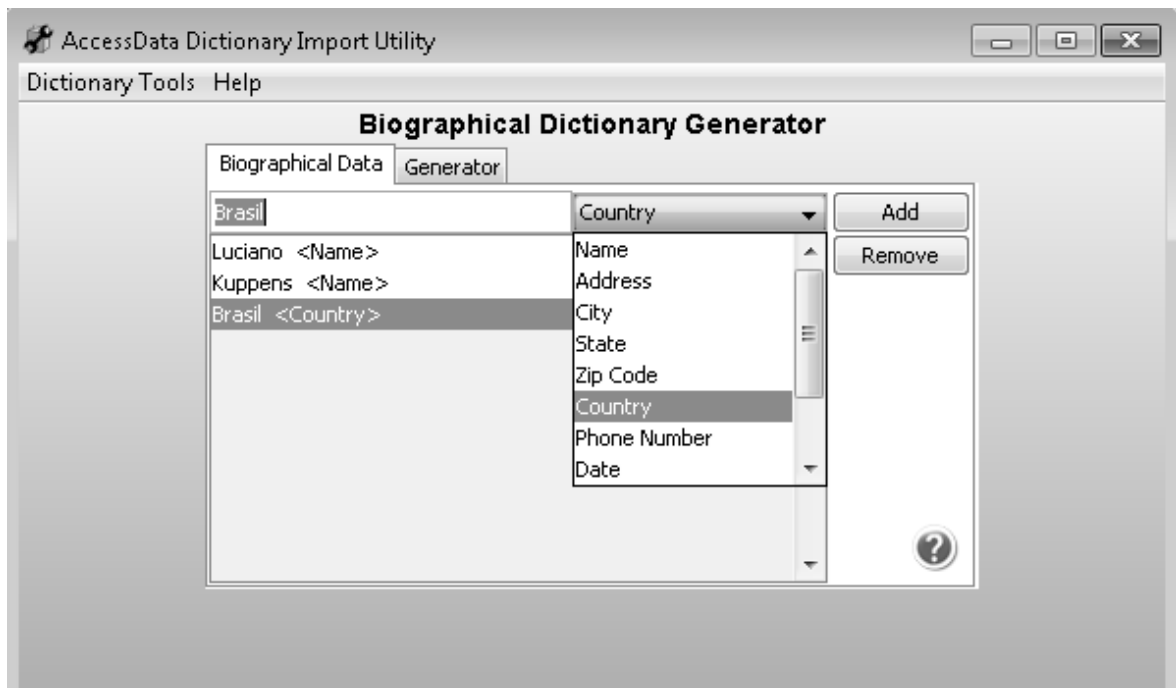


Figura 5.4: Exemplo dos campos da ferramenta *Biographical Dictionary Generator*

De posse dos dados personalizados (a) e (b) pode-se finalizar a segunda fase do método GPE e criar um DPP. A geração das senhas sugeridas também é feita pela ferramenta *Probabilistic Password Cracker* desenvolvida por (Weir *et al.*, 2009), por meio do comando `pcfg_manager`.

Essa ferramenta permite o uso de múltiplos dicionários auxiliares que, nesse caso, serão as listagens (a) e (b). Pode-se ainda determinar probabilidades diferentes para cada dicionário auxiliar. No caso em questão, sugere-se aumentar a probabilidade para o dicionário contendo a listagem (a) e reduzir para a listagem (b).

Com isso, dadas as estruturas gramaticais da Fase 1, e a substituição dos elementos alfabéticos ( $L_x$ ) pelos dados biográficos na Fase 2, a geração do DPP é concluída.

Caso existam senhas conhecidas do alvo sendo analisado, deve-se inclui-las na Fase 1 e adicioná-las na forma de um terceiro dicionário auxiliar na Fase 2, com probabilidade superior aos demais.

Parte dos procedimentos descritos neste capítulo foi inspirada nas sugestões de trabalhos futuros presentes em (Weir, 2010).

## 5.5 - LIMITAÇÕES

A criação de GPEs apresentam algumas limitações:

- a) *Capitalização*: O método descrito não faz distinção entre letras maiúsculas e minúsculas, ou seja, a fase de treinamento calcula a ocorrência de letras, independente de sua capitalização. Dessa forma, dicionários auxiliares que contenham apenas expressões em minúsculas, nunca serão capazes de encontrar senhas que contenham alguma letra maiúscula;
- b) *Números e símbolos*: Quaisquer números ou símbolos que não estejam nas senhas utilizadas durante o treinamento, só aparecerão na relação final de senhas se estiverem presentes no dicionário auxiliar (onde só são considerados os tamanhos de cada entrada, e não sua composição). Assim, caso seja desejado que números importantes para o caso façam parte do cálculo probabilístico e, dessa forma, permeiem todo o dicionário gerado, deve-se artificialmente adicionar esses números na relação de senhas de treinamento, criando assim, uma estrutura gramatical alternativa.
- c) *Similaridade*: Para que as GPEs funcionem de maneira adequada, o conjunto de senhas de treinamento deve ser o mais similar possível daquelas sendo atacadas. Por exemplo, se o conjunto de senhas de treinamento não contiver pelo menos uma senha de 10 caracteres, nenhuma senha desse tamanho será recuperada, independentemente das entradas do dicionário auxiliar. Isso acontece porque nenhuma estrutura gramatical com 10 caracteres será gerada e, conseqüentemente, nenhuma sugestão desse tamanho será produzida.

Além disso, DPPs são difíceis de serem testados e comparados com outros métodos. Um DPP só será válido para um caso em análise e, em última instância, para a recuperação de apenas uma senha. Diante dessa dificuldade, os testes realizados no presente trabalho não utilizarão DPPs, mas apenas Dicionários Probabilísticos Especializados (DPEs), ou seja, os dicionários auxiliares não utilizarão dados biográficos. Apesar dessa limitação, o método de Gramáticas Probabilísticas Especializadas poderá ser comparado com técnicas tradicionalmente utilizadas na recuperação de senhas e, dessa forma, os DPEs gerados poderão ser um ótimo indicador para a eficácia de DPPs em um caso real.

## 6 - TESTES, RESULTADOS E ANÁLISE COMPARATIVA

Os testes realizados neste capítulo têm o objetivo de determinar se Dicionários Probabilísticos Especializados são mais eficazes para a obtenção de senhas de usuários brasileiros do que os métodos tradicionalmente utilizados. Esses testes são parcialmente baseados no trabalho realizado por (Weir, 2010).

### 6.1 - TESTES REALIZADOS E RESULTADOS OBTIDOS

Para a realização dos testes propostos, foi obtido um conjunto composto por 2.614.857 *hashes* (no formato MD5) de senhas brasileiras, totalizando 1.501.096 ocorrências únicas. Os *hashes* em questão funcionarão como um conjunto de controle e foram obtidos de uma base de dados de usuários brasileiros composta por senhas com, no mínimo, seis caracteres (as senhas em claro não foram obtidas).

Ao se comparar senhas geradas por diversos métodos com esse conjunto de *hashes*, pode-se determinar qual método é mais eficaz. Essa comparação pode ser resumida a um simples ataque por adivinhação (Weir, 2010), composto pelos seguintes passos (veja Figura 6.1):

- 1) Gerar uma sugestão de senha, por exemplo: *senha123*;
- 2) Calcular o *hash* da sugestão gerada no passo anterior;
- 3) Comparar o *hash* gerado no passo anterior com os *hashes* sendo atacados.

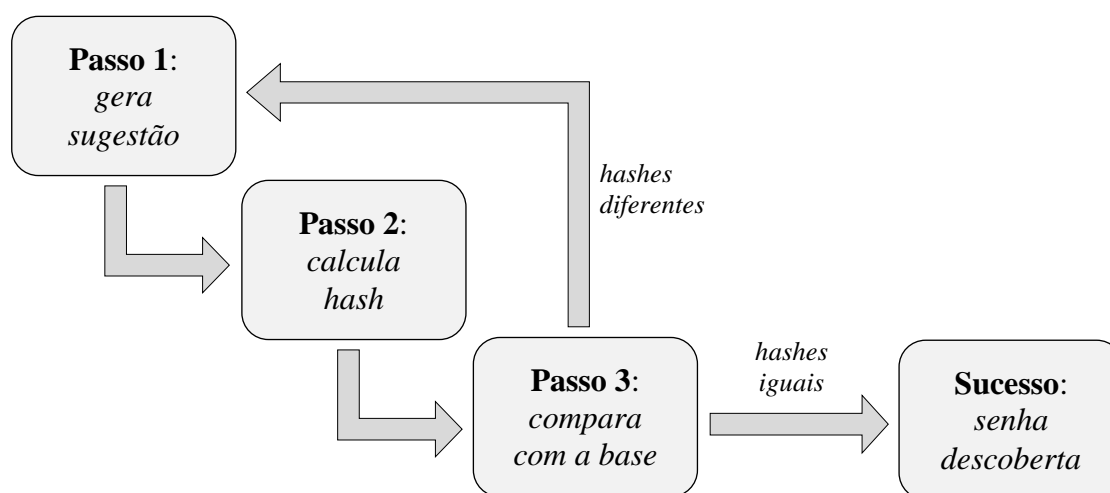


Figura 6.1: Passos realizados no ataque por adivinhação



Se o resultado do passo (3) forem *hashes* idênticos, a senha é considerada descoberta. Caso sejam resultados distintos, retorna-se ao passo (1) e uma nova sugestão de senha é testada. Neste trabalho, o passo (1) foi executado por meio da criação de dicionários especializados e por outros métodos que serão detalhados a seguir.

Para os passos (2) e (3), optou-se por utilizar o software *Jonh the Ripper*. Essa escolha se deve por diversos motivos, dentre eles: possibilidade de comparar os resultados com trabalhos anteriores; por ser um projeto de código aberto que permite saber exatamente qual tratamento está sendo dado para cada senha; pela possibilidade de importar e exportar sugestões de senhas para outros softwares; por ser um aplicativo bastante popular que implementa diversos métodos para recuperação de senhas. Foi utilizada a versão *1.7.8-jumbo-8* para *Linux*.

Ressalta-se que os *hashes* sendo atacados não têm nenhuma relação com as senhas em claro originalmente obtidas para o treinamento das gramáticas e geração dos dicionários especializados. Essa condição simula uma situação comum encontrada no processo de decifragem de arquivos apreendidos, uma vez que o perito às vezes não tem informações do material sendo analisado, como senhas anteriormente empregadas e sua complexidade.

A utilização desses conjuntos disjuntos é justificada por (Malone & Maher, 2011) ao mostrarem que senhas de um conjunto são boas candidatas para prever senhas de um conjunto diferente.

“Consequentemente, senhas de uma lista fornecem bons candidatos quando se está adivinhando ou quebrando senhas de outra lista. Um atacante que coleta senhas vazadas de um grupo de sítios tem um ótimo ponto de partida para a quebra de outras senhas” (Malone & Maher, 2011).

Para realizar os testes, foram utilizados três métodos diferentes de geração de senhas: Gramáticas Probabilísticas Especializadas, geração padrão de senhas do *John the Ripper* (*wordlist* e *incremental*) e força bruta.

Assim, foram geradas sugestões de senhas utilizando os três métodos descritos (passo 1) e para cada sugestão gerada, foi calculado seu *hash* (passo 2), cujo resultado foi comparado com o conjunto de controle (passo 3). Os testes realizados foram agrupados de acordo com o método e variáveis utilizadas (o detalhamento de cada método será feito a seguir).

Os valores apresentados do número de senhas descobertas referem-se ao total de *hashes* do conjunto de controle (e não às entradas únicas). Dessa forma, se a senha *123456* se repetir, por exemplo, 100 vezes no conjunto de controle, o valor apresentado aqui será 100 e não apenas 1.

Ressalta-se ainda que, na maioria dos casos, os testes foram limitados a 500 milhões de senhas sugeridas. Além disso, só foram geradas senhas com seis ou mais caracteres.

### **6.1.1 - Gramáticas Probabilísticas Especializadas**

Diante da dificuldade de se testar Dicionários Probabilísticos Personalizados (DPPs) já apresentada no capítulo anterior, optou-se por testar o método de Gramáticas Probabilísticas Especializadas (GPEs) utilizando Dicionários Probabilísticos Especializados (DPEs), ou seja, os dicionários auxiliares não utilizarão dados biográficos.

A primeira fase do método de GPE consiste no treinamento da gramática, que foi realizado no Capítulo 5. Foram treinadas duas gramáticas: a primeira com 50 mil senhas brasileiras (que será identificada como *GPE 50k*) e a segunda com pouco mais de 3 mil senhas genéricas (que será identificada como *GPE 3k*).

A segunda fase do método consiste na geração de senhas sugeridas, ou seja, na criação de DPEs. Para isso, é necessária a utilização de dicionários auxiliares.

O presente trabalho utiliza dois dicionários auxiliares:

- 1) *Português*: Dicionário contendo apenas palavras da língua portuguesa retiradas de um popular dicionário (também disponível em versão eletrônica). O total de entradas nesse dicionário é 131.877 expressões, porém, não foram utilizadas palavras com acentos, cedilha ou travessão,

reduzindo o conjunto para 80.595 palavras. Esse dicionário será identificado no restante do trabalho como “Português”;

- 2) *Completo*: Dicionário muito mais abrangente, gerado a partir de diversas fontes, incluindo: palavras em português e inglês, listagens encontradas na Internet contendo senhas e nomes diversos e as 50 mil senhas utilizadas na primeira fase do processo. O total de entradas desse dicionário era 585.550, porém, eliminando números, símbolos e entradas repetidas, o conjunto foi reduzido para 509.816 palavras. Esse dicionário será identificado no restante do trabalho como “Completo”.

Definidos os dicionários auxiliares, a criação dos DPEs seguiu o procedimento descrito no capítulo anterior, utilizando a ferramenta *Probabilistic Password Cracker* desenvolvida em (Weir *et al.*, 2009).

A primeira tentativa utilizou todas as 2.421 estruturas gramaticais da *GPE 50k*. Entretanto, o processo foi interrompido quando o arquivo contendo as senhas sugeridas atingiu 30GB. Estima-se que, utilizando todas as estruturas e palavras do dicionário auxiliar mais curto, seriam geradas  $3,3 \times 10^{36}$  sugestões de senhas. Dessa forma, optou-se por limitar os DPEs a 500 milhões de sugestões.

As cinco primeiras e cinco últimas sugestões geradas para cada dicionário auxiliar da *GPE 50k* e da *GPE 3k* foram listadas na Tabela 6.1. Como pode ser observado nessa tabela, as primeiras senhas sugeridas de cada gramática são as mesmas, independente do dicionário auxiliar utilizado. Isso ocorre porque essas senhas são as mais prováveis de ocorrerem, de acordo com o treinamento realizado.

Com isso, conclui-se a criação de quatro DPEs compostos por 500 milhões de senhas sugeridas em cada. Esses quatro dicionários foram identificados como: *GPE 50k – Português*, *GPE 50k – Completo*, *GPE 3k – Português* e *GPE 3k – Completo*, e contêm as seguintes regras de formação:

- a) *GPE 50k – Português*:
  - Treinamento da Gramática: 50.000 senhas brasileiras conhecidas;
  - Dicionário Auxiliar: mais de 80.000 palavras comuns em português;



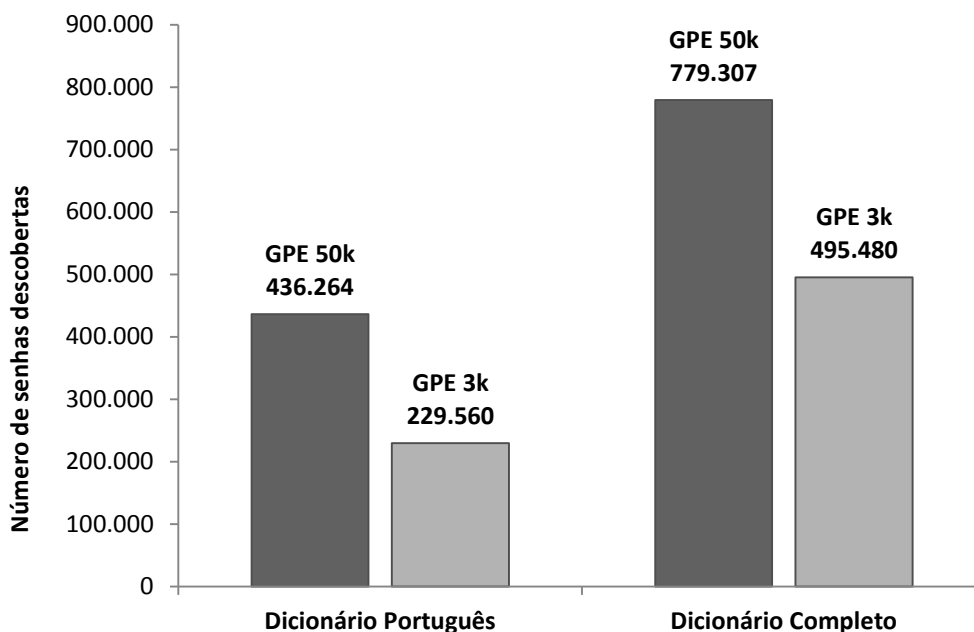


Figura 6.2: Número de senhas descobertas com o método GPE

### 6.1.2 - John the Ripper

Para o segundo teste foram utilizados dois modos de operação do software *John the Ripper* (JtR): *incremental* e *wordlist*.

O modo de operação *incremental* utiliza métodos probabilísticos para gerar as sugestões de senhas. Este modo utiliza cadeias de Markov de nível dois (descritas no Capítulo 3), ou seja, utiliza probabilidades condicionais para prever o aparecimento de uma sequência de três letras (Weir, 2010).

O modo de operação *wordlist* utiliza um conjunto de regras de alteração (veja Capítulo 3) a partir do dicionário fornecido. A Tabela 6.2 lista as 10 primeiras regras de alteração para o modo *wordlist* do JtR.

Para o modo de operação *incremental* foram geradas 500 milhões de sugestões de senhas com seis ou mais caracteres. E para o modo de operação *wordlist* foram geradas 3.775.778 e 24.041.869 sugestões de senhas com seis ou mais caracteres utilizando, respectivamente, os dicionários “Português” e “Completo” (descritos anteriormente). Esses valores são o número máximo de sugestões geradas pelo JtR utilizando as regras de alteração padrão presentes na ferramenta.

O número de comparações positivas entre esses dois modos de operação e o grupo de controle é apresentado na Figura 6.2. Esses valores representam o número total de senhas descobertas no teste em questão.

Tabela 6.2: Dez primeiras regras de alteração do modo *wordlist* no JtR

#	Descrição da Regra	Exemplo	
		Entrada	Sugestão
01	Nenhuma modificação (utiliza as entradas como elas aparecem no dicionário)	<i>aBcD123</i>	<i>aBcD123</i>
02	Coloca as entradas alfanuméricas em minúsculas	<i>aBcD123</i>	<i>abcd123</i>
03	Coloca a primeira letra das entradas alfanuméricas em maiúsculas	<i>aBcD123</i>	<i>ABcD123</i>
04	Coloca as entradas puramente alfabéticas em minúsculas e no plural	<i>CaSa</i>	<i>casas</i>
05	Coloca as entradas puramente alfabéticas em minúsculas e adiciona “1” ao final	<i>CaSa</i>	<i>casa1</i>
06	Coloca a primeira letra das entradas puramente alfabéticas em maiúsculas e adiciona “1” ao final	<i>casa</i>	<i>Casa1</i>
07	Duplica entradas curtas e puramente alfabéticas	<i>casa</i>	<i>casacasa</i>
08	Coloca as entradas puramente alfabéticas em minúsculas e em ordem inversa	<i>CaSa</i>	<i>asac</i>
09	Adiciona “1” no início de entradas puramente alfabéticas	<i>Casa</i>	<i>1Casa</i>
10	Coloca todas as letras das entradas puramente alfabéticas em maiúsculas	<i>CaSa</i>	<i>CASA</i>

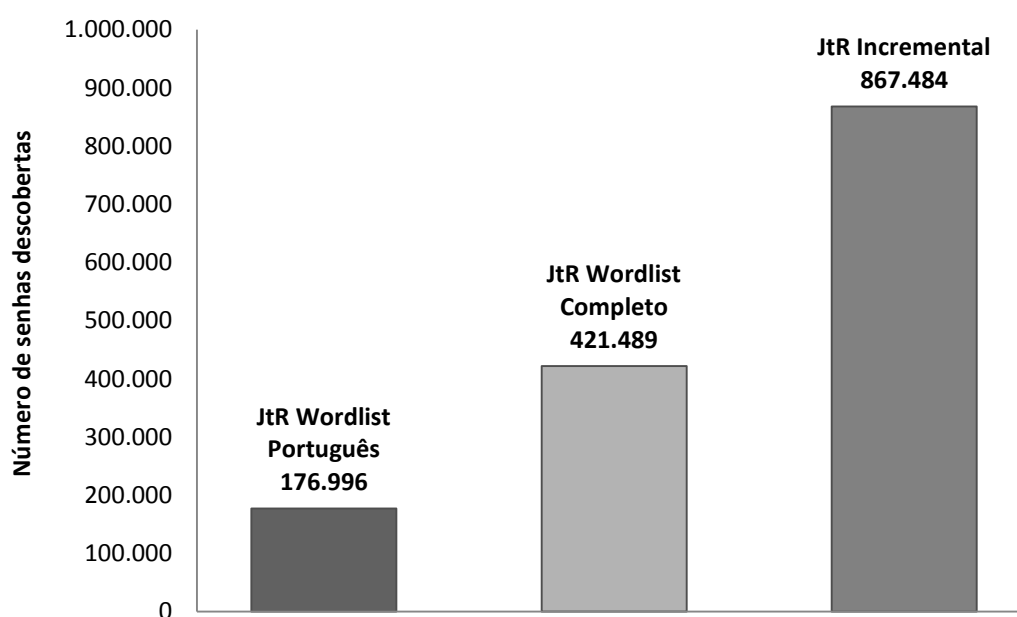


Figura 6.3: Número de senhas descobertas com o método JtR

### 6.1.3 - Força Bruta

Para o terceiro teste foi realizado um ataque de força bruta limitado. Foram geradas sugestões de senhas de seis caracteres, compostas apenas por dígitos e letras minúsculas (0-9 e a-z). A sequência de sugestões gerada foi 000000, 000001, 000002, [...], 89oqgu e 89oqgv, totalizando 500 milhões de entradas.

O número de comparações positivas entre o ataque por força bruta e o grupo de controle é 714.356. Esse valor representa o número total de senhas descobertas no teste em questão.

## 6.2 - ANÁLISE COMPARATIVA

O presente tópico fará uma análise comparativa entre os métodos de geração de senhas apresentados. As Gramáticas Probabilísticas Especializadas serão comparadas:

- a) Entre si (GPE *versus* GPE);
- b) Com o método de regras de alteração (GPE *versus* JtR-wordlist);
- c) Com outro método probabilístico (GPE *versus* JtR-incremental); e
- d) Com o método de força bruta (GPE *versus* Força Bruta).

Os dados utilizados nessas comparações estão presentes no tópico anterior. Ressalta-se que para a realização de uma análise adequada, os seguintes cuidados foram tomados:

- A comparação sempre é realizada com os mesmos dicionários de entrada;
- Utiliza-se sempre o mesmo número de senhas sugeridas (na maioria dos casos limitado a 500 milhões de entradas);
- Todos os métodos foram adaptados para fornecer sugestões de senhas com seis ou mais caracteres.

### 6.2.1 - Treinamento da Gramática Probabilística Especializada (GPE *versus* GPE)

Observa-se pela Figura 6.2 que o treinamento da gramática especializada com senhas brasileiras (GPE 50k) resulta em um número maior de senhas recuperadas em comparação

com o treinamento com senhas genéricas (*GPE 3k*). Os ganhos variam de 57% a 90%, dependendo do dicionário auxiliar utilizado.

Quando comparamos um mesmo tipo de treinamento utilizando dicionários auxiliares diferentes (“Português” *versus* “Completo”), verificam-se ganhos variando de 79% a 116%, dependendo do grupo de senhas utilizado para o treinamento da gramática.

A GPE treinada com 50 mil senhas brasileiras utilizando o dicionário auxiliar “Completo” obteve os melhores resultados, encontrando o equivalente a 30% das senhas do grupo de controle.

A Tabela 6.3 apresenta as dez primeiras senhas sugeridas pelas GPEs utilizando o dicionário “Completo” e quantas, de fato, foram encontradas no grupo de controle. A coluna “Posição” refere-se ao número de ocorrências em relação ao total de senhas encontradas. A senha 123456, por exemplo, foi a senha com o maior número de ocorrências encontradas e, dessa forma, encontra-se na posição 1.

Verifica-se que a *GPE 50k* sugeriu as três senhas com o maior número de ocorrências na primeira, terceira e sétima senhas geradas, respectivamente. Já a *GPE 3k* sugeriu a senha com o maior número de ocorrências apenas na décima senha gerada.

Tabela 6.3: Senhas sugeridas e encontradas pelas GPEs com o dicionário “Completo”

	GPE 50k – Completo			GPE 3k-Completo		
	Senha sugerida	Senhas encontradas		Senha sugerida	Senhas encontradas	
		Ocorrências	Posição		Ocorrências	Posição
01	123456	32703	1	888888	180	366
02	123456789	1040	28	789456	501	100
03	102030	2978	2	696969	297	203
04	010203	1749	7	666666	323	181
05	12345678	1066	25	654321	2104	2
06	101010	1374	14	222222	540	90
07	654321	2104	3	181818	311	189
08	159753	798	52	171717	355	159
09	1234567	596	83	131313	1453	9
10	123321	617	80	123456	32703	1



A Figura 6.4 apresenta o número de senhas sugeridas necessárias para encontrar um determinado número de senhas do conjunto de controle, ou seja, quantas senhas são recuperadas para um determinado número de tentativas. Verifica-se que, nas quatro GPEs testadas, o número de senhas descobertas cresce muito rapidamente e se estabiliza em seguida. Isso mostra que, de fato, as GPEs geram primeiramente as senhas mais prováveis de serem encontradas.

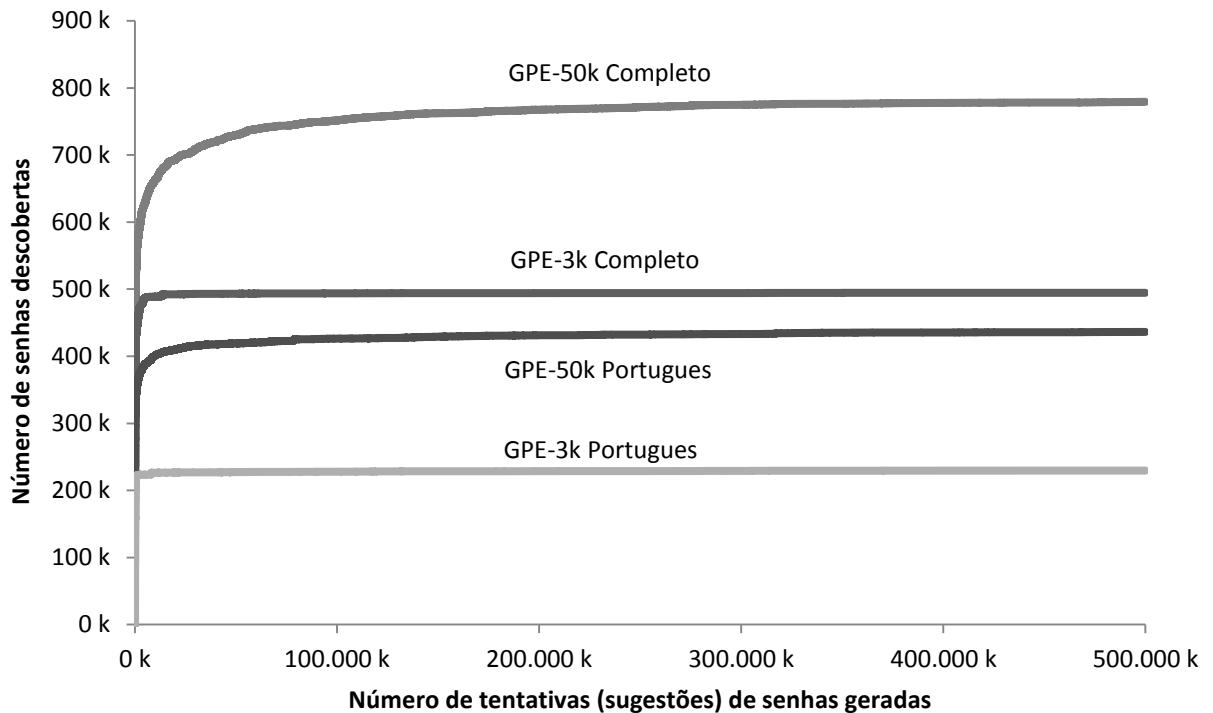


Figura 6.4: Número de senhas sugeridas *versus* senhas descobertas – GPEs

Verifica-se, portanto, que o treinamento da gramática é essencial para a obtenção de melhores resultados e que o conjunto de senhas utilizado para o treinamento deve ser o mais similar possível àquelas senhas sendo atacadas.

### 6.2.2 - Regras de Alteração (GPE *versus* JtR-wordlist)

A análise realizada agora procura comparar a criação de sugestões de senhas das GPEs com as regras de alteração padrões do JtR, apresentadas na seção 6.1.2. Quando (Weir, 2010) realizou essa mesma comparação, o ganho utilizando GPEs foi de 36% a 93% em relação às regras de alteração padrões do JtR.

Nos testes aqui realizados, uma vez que o número de senhas sugeridas pelo JtR no modo *wordlist* não atingiu 500 milhões de entradas, reduziu-se o número de tentativas geradas pelas GPEs de modo que os totais de sugestões sejam os mesmos, proporcionando uma comparação adequada. Dessa forma, as entradas das *GPE 50k* e *GPE 3k* foram reduzidas para 3.775.778 no dicionário “Português” e 24.041.869 no dicionário “Completo”.

Em todos os casos, o número de senhas recuperadas pelas GPEs foi superior ao JtR. Os ganhos são 26% e 117% para o dicionário “Português” e 17% e 66% para o dicionário “Completo”.

O número de comparações positivas entre os métodos descritos e o grupo de controle é apresentado na Figura 6.5. Esses valores representam o número total de senhas descobertas no teste em questão. Ressalta-se que o número de tentativas utilizadas foi limitado ao número de sugestões geradas pelo JtR.

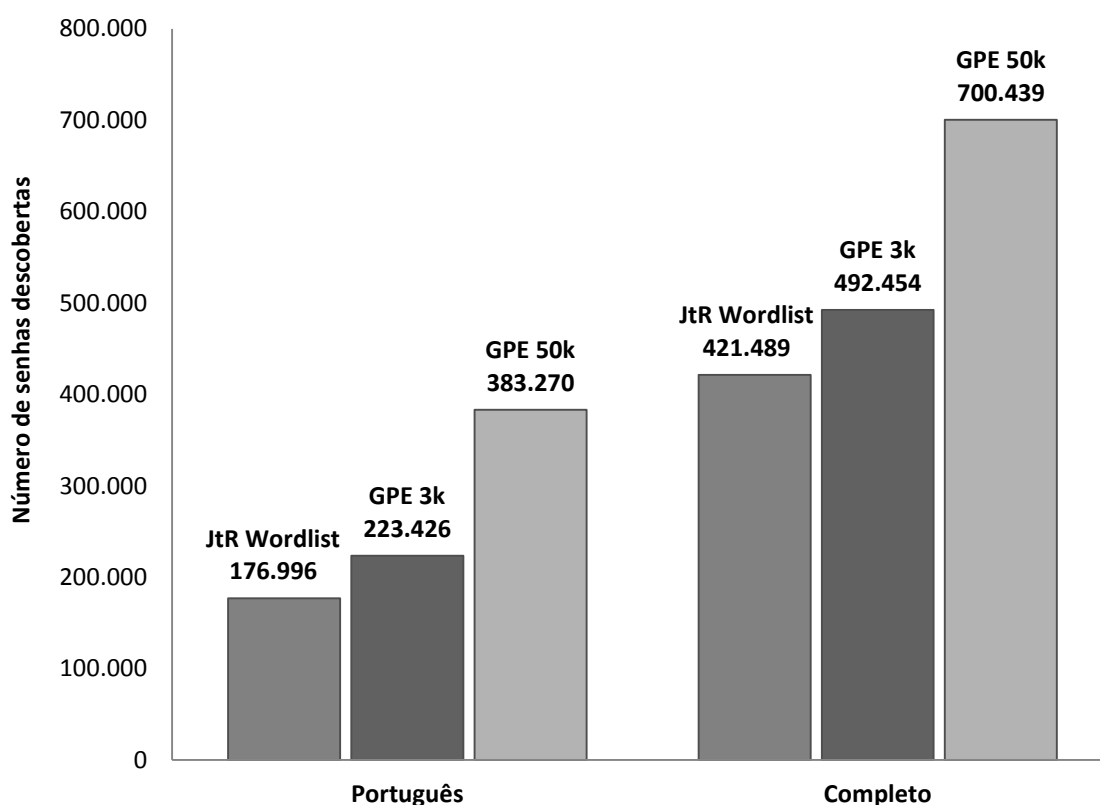


Figura 6.5: Número de senhas descobertas com os métodos GPE e JtR-wordlist

Verifica-se que as regras de alteração do JtR não apresentam um bom resultado em comparação com as GPEs. As maiores diferenças foram em relação ao dicionário auxiliar “Português”, cujos resultados foram próximos aos obtidos por (Weir, 2010).

Verifica-se ainda que o melhor resultado (*GPE 50k – Completo*) foi capaz de encontrar 27% do total de senhas do conjunto de controle, mesmo com a limitação do número de tentativas. O resultado do teste anterior (com 500 milhões de tentativas realizadas) havia encontrado apenas 3% a mais de senhas. Isso comprova que o início das GPEs contém – como previsto – as sugestões de senhas mais prováveis de terem sido utilizadas.

A Tabela 6.4 apresenta as dez primeiras senhas sugeridas pela *GPE 50k* e pelo *JtR-wordlist* utilizando o dicionário “Completo”. Verifica-se que as primeiras senhas sugeridas pelo JtR apresentam um baixo número de ocorrências, ou seja, esse método necessita de mais tempo (tentativas) para atingir bons resultados.

A Figura 6.6 apresenta o número de tentativas necessárias para encontrar um determinado número de senhas do conjunto de controle (utilizando o dicionário “Completo”). Verifica-se que o JtR também cresce muito rapidamente e se estabiliza, porém, com respostas sempre inferiores às GPEs.

Tabela 6.4: Primeiras senhas sugeridas e encontradas pela GPE e JtR-wordlist

	GPE 50k – Completo			JtR-wordlist – Completo		
	Senha sugerida	Senhas encontradas		Senha sugerida	Senhas encontradas	
		Ocorrências	Posição		Ocorrências	Posição
01	123456	32703	1	aaaaaa	69	985
02	123456789	1040	28	aaaaaaaa	3	12096
03	102030	2978	2	aaazzz	1	22508
04	010203	1749	7	aabbccdd	3	12097
05	12345678	1066	25	aaajff	1	22509
06	101010	1374	14	aallliccee	1	22510
07	654321	2104	3	aardvark	1	22511
08	159753	798	52	aassaa	1	22512
09	1234567	596	83	abacatada	4	10091
10	123321	617	80	abacate	120	542

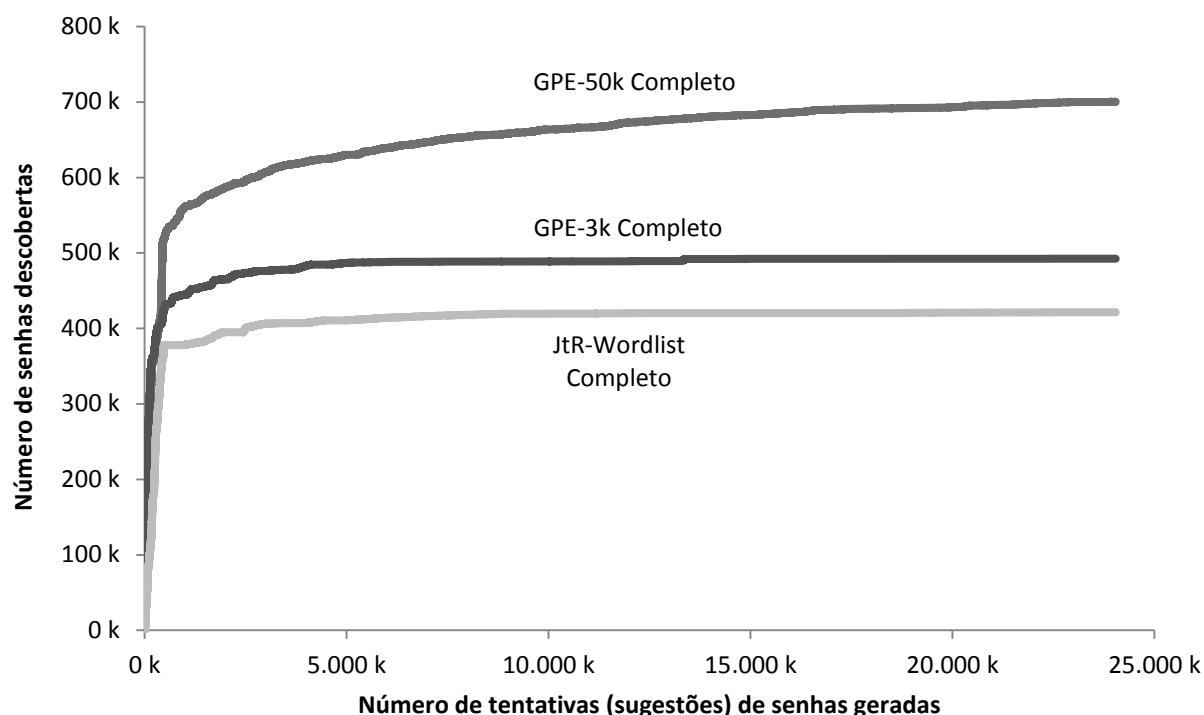


Figura 6.6: Número de senhas sugeridas *versus* senhas descobertas – Regras de Alteração

### 6.2.3 - Probabilísticos (GPE *versus* JtR-incremental)

A análise realizada agora visa comparar dois métodos probabilísticos: Gramáticas Especializadas e Cadeias de Markov (que são implementadas por padrão no modo *incremental* do JtR). As comparações aqui realizadas referem-se à GPE com o melhor resultado: *GPE 50k – Completo*.

O número de senhas recuperadas com o método JtR-incremental foi de 867.484 e a GPE descobriu 779.307 senhas. Ou seja, o modo probabilístico do JtR recuperou 11% mais senhas que a GPE treinada com 50 mil senhas brasileiras utilizando o dicionário auxiliar “Completo”. Este é único resultado em que as GPEs são superadas por outros métodos.

Entretanto, a análise das senhas encontradas mostra que, em geral, a GPE recupera senhas mais complexas (maiores e com mais símbolos) e de forma mais rápida que o método JtR-incremental.

O tamanho médio das senhas recuperadas pela GPE é de 7,26 caracteres e do JtR-incremental é de 6,44 caracteres. A Figura 6.7 apresenta o número de senhas encontradas em relação ao seu tamanho. Nota-se que o JtR-incremental só recuperou senhas de 6, 7 e 8

caracteres, enquanto a GPE recuperou senhas de até 14 caracteres. Além disso, o JtR só superou a GPE em senhas de seis caracteres.

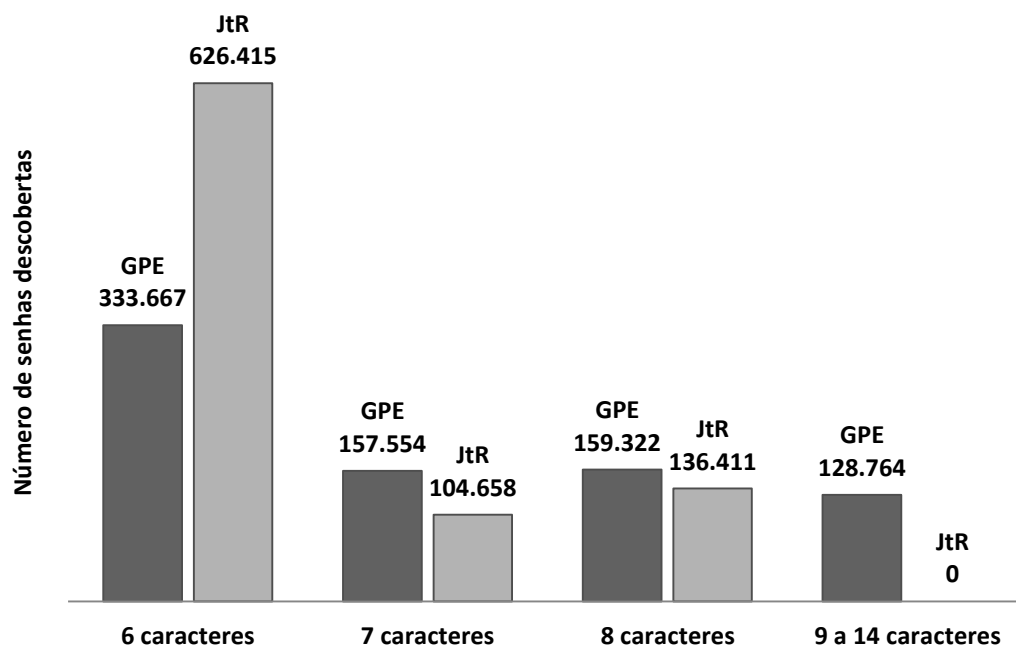


Figura 6.7: Número de senhas descobertas *versus* tamanho (utilizando GPE e JtR)

Outro aspecto relevante é que a GPE recuperou um maior número de senhas compostas por símbolos: 729 contra 51 senhas recuperadas pelo JtR. A Figura 6.8 apresenta a proporção das senhas recuperadas em cada método de acordo com sua composição (letras, dígitos e símbolos). Observa-se que o JtR só atingiu 7% do total de senhas recuperadas contendo símbolos. Além disso, a GPE só foi superada pelo JtR em senhas relativamente simples, compostas somente por dígitos (em especial, sequências compatíveis com datas).

A Tabela 6.5 apresenta as dez primeiras senhas sugeridas pela *GPE 50k – Completo* e pelo *JtR-incremental*. Como ocorrido com JtR configurado para o modo *wordlist*, as primeiras senhas sugeridas pelo JtR apresentam um baixo número de ocorrências.

Já a Figura 6.9, apresenta o número de senhas necessárias para encontrar um determinado número de senhas do conjunto de controle. O gráfico mostra que a GPE é superior ao JtR até pouco mais de 300 milhões de sugestões geradas. O pico observado no JtR refere-se à geração de senhas numéricas compatíveis com datas. Observa-se ainda que o gráfico da GPE é mais suave que os demais, característica desse método probabilístico. Isso ocorre

porque são geradas senhas cada vez menos prováveis ocasionando, como esperado, a estabilização do gráfico.

Analisando a Tabela 6.5 em conjunto com a Figura 6.9, verifica-se que a GPE recupera senhas muito mais rapidamente que o JtR.

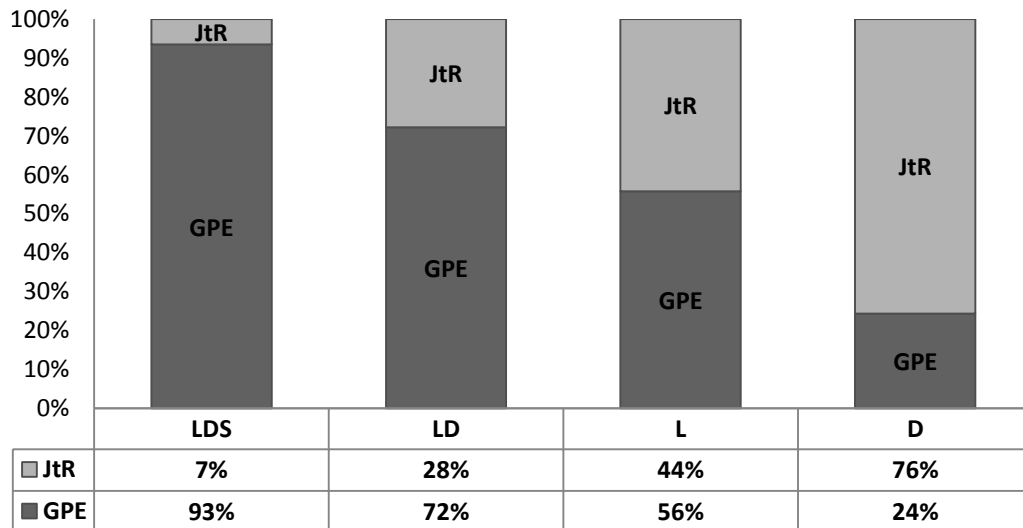


Figura 6.8: Proporção das senhas com sua composição (letras, dígitos e símbolos)

Tabela 6.5: Primeiras senhas sugeridas e encontradas pela GPE e JtR-incremental

	GPE 50k – Completo			JtR-incremental		
	Senha sugerida	Senhas encontradas		Senha sugerida	Senhas encontradas	
		Ocorrências	Posição		Ocorrências	Posição
01	123456	32703	1	marine	25	3680
02	123456789	1040	28	maring	1	77707
03	102030	2978	2	maris1	1	77708
04	010203	1749	7	marise	32	2183
05	12345678	1066	25	marana	11	10050
06	101010	1374	14	maran1	1	77709
07	654321	2104	3	mandys	1	77710
08	159753	798	52	mannie	1	77711
09	1234567	596	83	mickey	61	695
10	123321	617	80	michel	114	335

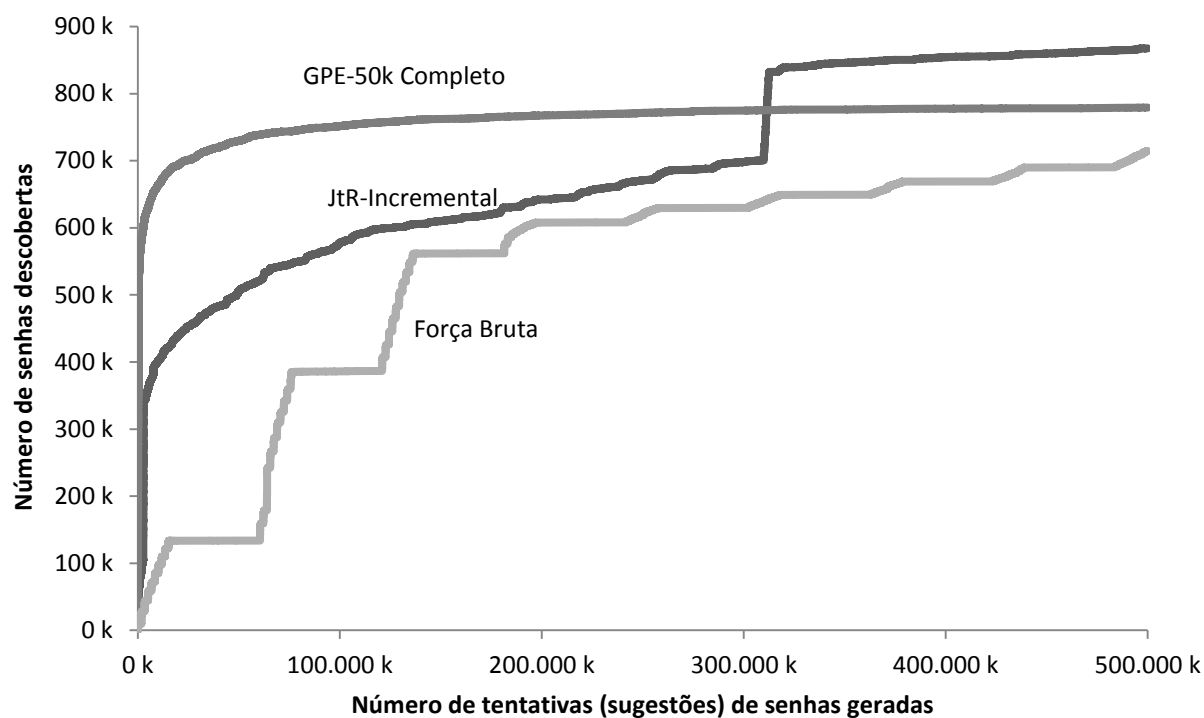


Figura 6.9: Número de senhas sugeridas *versus* senhas descobertas – diversos

Portanto, apesar do número absoluto de senhas utilizando o *JtR-incremental* ter sido superior à *GPE 50k – Completo*, essas senhas foram mais simples (curtas e com poucos símbolos) e obtidas mais lentamente. Se limitássemos a análise a senhas com sete ou mais caracteres, os resultados da GPE superariam o JtR.

#### 6.2.4 - Força Bruta (GPE *versus* Força Bruta)

O método de Força Bruta foi capaz de recuperar 714.356 senhas, enquanto que a melhor GPE recuperou 779.307 senhas. Ou seja, a *GPE 50k – Completo* recuperou 9% mais senhas que a método de Força Bruta.

Apesar desse valor não ser tão significativo, as senhas recuperadas pela GPE são muito mais complexas e foram obtidas muito mais rapidamente, como pode ser observado na Figura 6.9. Os picos do método de força bruta vistos no gráfico em questão referem-se, como no *JtR-incremental*, a senhas numéricas compatíveis com datas ou números de telefone.

Como se limitou a 500 milhões de tentativas, só foram obtidas por força bruta senhas de seis caracteres (utilizando um dicionário de 36 elementos – letras minúsculas e números – existem mais de 2 bilhões de senhas possíveis com seis caracteres).

Verifica-se que ataques por força bruta são eficientes para senhas muito curtas, onde é possível gerar e testar rapidamente todas as combinações possíveis. Porém, quando se aumenta o tamanho e o conjunto possível de caracteres, esse método se torna completamente ineficiente.

Entretanto, como observado em alguns resultados aqui apresentados, uma tática interessante pode ser a utilização de um ataque de força bruta adaptado. Esse ataque seria realizado antes de outros métodos e limitado a senhas de seis e oito dígitos compatíveis com datas e números de telefone.

### **6.3 - EXPERIMENTOS ADICIONAIS**

Além dos testes anteriormente descritos, foram realizados outros dois experimentos com parâmetros de operação bastante distintos dos demais:

- a) *Realimentação*: criação de uma nova gramática especializada, treinada somente com as senhas já descobertas nos testes anteriores;
- b) *Contínuo*: comparação de uma GPE e o JtR sem limitar o número de senhas (para uma comparação adequada, duas máquinas virtuais idênticas foram utilizadas durante um mesmo período de tempo).

#### **6.3.1 - Realimentação**

Este teste adicional utiliza todas as senhas obtidas nos testes anteriores para o treinamento de uma nova gramática especializada. Dessa forma, foram criadas novas GPEs utilizando somente as senhas já descobertas do grupo de controle. Essa nova gramática foi treinada utilizando todas as 2.091.653 senhas do grupo de controle até então obtidas (equivalente a 80% do total).



Mais uma vez, cada dicionário foi limitado a 500 milhões de entradas. Foram utilizados os mesmos procedimentos descritos no Capítulo 5 e, além dos dicionários auxiliares “Português” e “Completo”, foi criado mais um dicionário auxiliar, composto apenas por letras do conjunto de senhas descobertas.

Esses dicionários foram identificados como: *GPE-R – Português*, *GPE-R – Completo* e *GPE-R – Senhas*, apresentando as seguintes regras de formação:

a) *GPE-R – Português*:

- Treinamento: mais de 2.000.000 de senhas anteriormente descobertas;
- Dicionário Auxiliar: mais de 80.000 palavras comuns em português;

b) *GPE-R – Completo*:

- Treinamento: mais de 2.000.000 de senhas anteriormente descobertas;
- Dicionário Auxiliar: mais de 500.000 palavras de diversas fontes;

c) *GPE-R – Senhas*:

- Treinamento: mais de 2.000.000 de senhas anteriormente descobertas;
- Dicionário Auxiliar: mais de 280.000 palavras e fragmentos alfabéticos retirados do conjunto de senhas já descobertas;

O número de comparações positivas entre os dicionários realimentados e o grupo de controle é apresentado na Figura 6.10. Esses valores representam o número total de senhas descobertas no teste em questão.

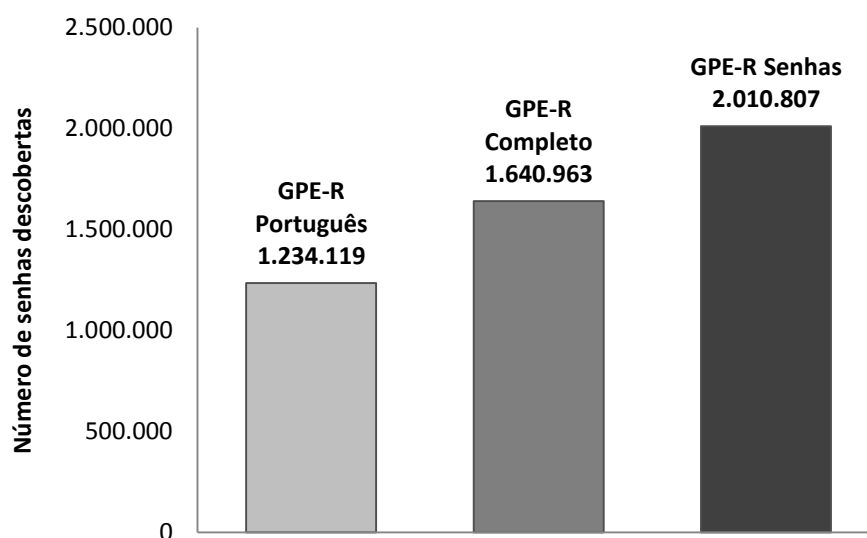


Figura 6.10: Número de senhas descobertas com GPEs realimentadas

Em comparação aos melhores resultados sem realimentação, os ganhos são de 183% e 110%, com os dicionários “Português” e “Completo”, respectivamente.

Já utilizando o novo dicionário auxiliar “Senhas”, obteve-se o melhor resultado até então, equivalente a 77% de todas as senhas do conjunto de controle. Entretanto, já eram conhecidas 2.091.653 senhas dos testes anteriores, que equivalem a 80% do total.

Este teste adicional confirma que as GPEs serão mais eficazes quando treinadas com senhas similares àquelas sendo atacadas.

### 6.3.2 - Contínuo

No último teste realizado, foram comparados os dois melhores métodos até então: *GPE-R – Senhas* e *JtR-incremental*. Mas dessa vez, os métodos foram limitados ao tempo em que ficaram processando (e não a 500 milhões de tentativas). Ambos os ataques foram realizados por duas máquinas virtuais idênticas que realizaram o ataque ininterruptamente durante 24 dias e 15 horas.

O número de comparações positivas entre esses dois métodos e o grupo de controle é apresentado na Figura 6.11.

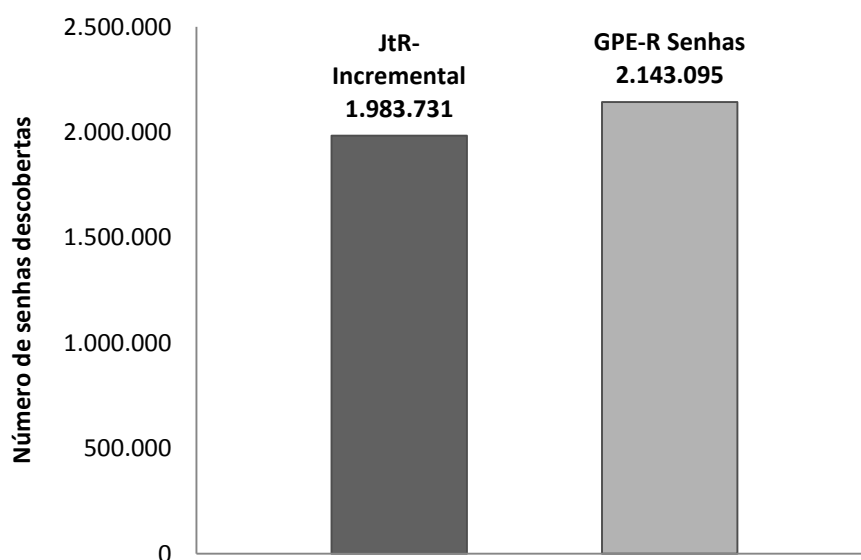


Figura 6.11: Senhas descobertas depois de mais de 24 dias de processamento

Após mais de 24 dias de processamento, verifica-se que o número de senhas descobertas pela GPE é 8% maior que pelo JtR. O JtR limitado a 500 milhões de tentativas havia descoberto 867.484 senhas e, neste experimento, o ganho obtido foi de 128%. Este resultado mostra que o número de senhas descobertas pelo JtR tende a crescer mais lentamente com o tempo, como observado na Figura 6.9.

Já a GPE limitada a 500 milhões de senhas havia descoberto 2.010.807 e, neste experimento, o ganho foi de apenas 7%, ou seja, a grande maioria das senhas já havia sido encontrada nas primeiras tentativas.

Este teste adicional confirma que as GPEs são muito rápidas e que não precisam de muitas tentativas para recuperar um grande número de senhas. Assim, as primeiras senhas sugeridas pelas GPEs são, de fato, muito prováveis de terem sido utilizadas. Com isso, se uma senha não for descoberta logo no início da GPE, a chance de ela ser encontrada posteriormente se reduz muito rapidamente.

O número total de senhas descobertas, incluindo todos os testes regulares e os dois testes adicionais, foi de 2.226.868, equivalente a 85% do total do grupo de controle.

## 7 - CONCLUSÃO

O presente trabalho foi motivado pela falta de métodos eficazes para tratar arquivos que tenham sido criptografados de forma segura. Mecanismos robustos de criptografia, quando bem empregados, podem impossibilitar o acesso da Justiça a dados apreendidos e, em última instância, impedir a materialização de provas e a eventual responsabilização dos envolvidos.

Esta obra foi realizada no contexto forense em que, durante a análise de computadores desligados (*post-mortem* e *off-line*), peritos se deparam com arquivos ou discos rígidos criptografados. Nesses casos, quando algoritmos criptográficos robustos são utilizados, só resta ao perito tentar descobrir qual senha foi utilizada pelo usuário.

Assim, ao invés de tentar explorar eventuais vulnerabilidades do sistema ou varrer todo o espaço de chaves do algoritmo criptográfico, a solução é atacar o elemento mais fraco envolvido: a senha criada pelo usuário.

A ampla literatura científica sobre esse assunto converge para um ponto: pessoas não utilizam senhas seguras, que, em geral, são difíceis de serem memorizadas. Senhas seguras devem ser longas, imprevisíveis e nunca reutilizadas e isso é exatamente o oposto do observado na prática. Dessa forma, enquanto usuários precisarem memorizar suas senhas, elas serão vulneráveis a ataques inteligentes.

Foram analisados seis métodos de recuperação de senhas à disposição do perito: recuperação direta, força bruta, dicionários, regras de alteração, algoritmos probabilísticos e métodos pré-computados. Esses métodos podem ser acelerados por meio de quatro técnicas diferentes: unidades de processamento gráfico, arranjo de portas programáveis, computação em grade e computação na nuvem. Além disso, quatro ferramentas muito utilizadas pela perícia implementam parte desses métodos e técnicas: DNA, EDPR, PKF e JtR.

A análise comparativa desses recursos, indica que não há uma solução única. Os métodos, dispositivos e ferramentas apresentados precisam ser utilizados de forma complementar e o sucesso na recuperação de uma senha depende de uma série de fatores. Para algoritmos criptográficos robustos, o principal fator é a presença de bons dicionários personalizados. Esses dicionários são criados a partir de informações biográficas do suspeito e de dados

(em claro) coletados da mídia apreendida. Com isso, esses dicionários personalizados se tornam únicos, servindo especificamente para o caso em análise.

Este trabalho introduz a técnica de criação de Dicionários Probabilísticos Personalizados (DPPs). Para isso, foi utilizado o promissor método de Gramáticas Probabilísticas Especializadas (GPEs), proposto por (Glodek, 2008) e aperfeiçoado por (Weir *et al.*, 2009).

Esse método utiliza um conjunto de senhas em claro para produzir uma gramática, que é composta pela estrutura de formação dessas senhas, incluindo ocorrências de letras, números e símbolos. Por meio de um dicionário auxiliar, são sugeridas senhas em ordem decrescente de probabilidade. O método é, portanto, dividido em duas fases: treinamento da gramática e geração das senhas sugeridas.

Para o treinamento da gramática (primeira fase) – que só precisa ser realizado uma vez – foram coletadas 50 mil senhas de usuários brasileiros. Uma breve análise dessas senhas mostra algumas estatísticas interessantes:

- Senhas repetidas: 8%;
- Senha mais frequente: *123456* (equivalente a 1% do total);
- Tamanho médio: 9,5 (desconsideradas senhas com menos de 6 caracteres);
- Senhas entre seis e oito caracteres: 52%;
- Senhas compostas somente por números: 35%;
- Senhas compostas somente por letras: 29%;
- Senhas compostas por letras e números: 32%;
- Senhas contendo símbolos: 4%;
- Senhas que contêm letras maiúsculas: 1%;
- Senhas contendo uma data válida ou possível número de telefone: 20%.

A criação das senhas sugeridas (segunda fase) é realizada por meio de dicionários auxiliares que, no caso de DPPs, são obtidos com informações biográficas e de dados em claro coletados da mídia do suspeito.

Como DPPs são criados para casos específicos e, em última instância, para a decifragem de um único arquivo, é difícil compará-los com outros métodos. Em função disso, os testes

realizados utilizaram Dicionários Probabilísticos Especializados (DPEs) que são DPPs sem a personalização biográfica.

Os DPEs foram então comparados com outras três técnicas bastante utilizadas: Regras de Alteração, Cadeias de Markov e Força Bruta. Para cada técnica foi gerado um mesmo número de sugestões de senhas (500 milhões na maioria dos casos), os *hashes* dessas sugestões foram calculados e comparados com um conjunto de controle formado por mais de 2,6 milhões de *hashes*. Os *hashes* desse conjunto de controle são de senhas de usuários brasileiros com seis ou mais caracteres (as senhas em claro não são conhecidas e elas não estão relacionadas ao conjunto utilizado para treinamento da gramática).

Somando todos os resultados obtidos, foram descobertas 2.226.868 senhas, equivalente a 85% do total do conjunto de controle. Os testes e resultados mais relevantes são:

- 1) *Comparação de Regras de Alteração (GPE versus JtR-wordlist)*: o número de senhas recuperadas pelas GPEs foi superior ao JtR em todos os testes. Os ganhos variam de 17% a 117%, dependendo do dicionário e treinamento da gramática. As primeiras sugestões geradas pelo JtR resultaram em um número baixo de senhas descobertas, indicando que esse método é mais lento.
- 2) *Comparação de Métodos Probabilísticos (GPE versus JtR-incremental)*: o JtR recuperou 11% mais senhas que a GPE. Esse foi o único teste em que o total de senhas recuperadas pela GPE foi inferior. Apesar de ter sido superada em termos absolutos, o JtR só ultrapassa a GPE nas senhas simples, formadas apenas por números (em especial sequências compatíveis com datas e números telefônicos). O JtR só recuperou senhas com 6, 7 e 8 caracteres e a GPE senhas com até 14 caracteres foram recuperadas. Além disso, a GPE descobriu um número maior de senhas contendo símbolos. Finalmente, a velocidade de recuperação de senhas da GPE é muito superior ao JtR.

Em resumo, comprovou-se que as GPEs apresentam um resultado muito superior às demais técnicas até então utilizadas pela perícia. Em geral, as GPEs recuperaram senhas mais complexas e mais rapidamente que os demais métodos.

Dessa forma, conclui-se que um ótimo método para recuperar senhas de um suspeito investigado é a geração de um DPP com a GPE brasileira aqui desenvolvida e dados biográficos específicos do caso sob análise.

## **7.1 - CONTRIBUIÇÕES**

Este trabalho realizou um extenso estudo sobre a utilização de senhas, abordando uma série de aspectos presentes em dezenas de publicações.

Foi realizada uma análise comparativa dos métodos de recuperação de senhas existentes, incluindo ferramentas de *hardware* e *software* desenvolvidos para esse fim.

Foram analisadas 50 mil senhas brasileiras, gerando dados estatísticos importantes e inéditos no Brasil.

Comprovou-se que o método probabilístico de gramáticas especializadas funciona para o idioma português e com senhas brasileiras.

Foi criada uma gramática probabilística brasileira, que poderá ser utilizada em casos nacionais (em função de seu tamanho, não foi possível disponibilizá-la de forma impressa neste trabalho).

Foi sugerido um novo procedimento para a recuperação de senhas durante o exame pericial de mídias apreendidas: os Dicionários Probabilísticos Personalizados.

## **7.2 - TRABALHOS FUTUROS**

Uma oportunidade para pesquisas futuras é tentar reduzir as limitações presentes no método de GPEs, apresentadas no Capítulo 5 (capitalização, números/símbolos e similaridade).

Outra linha de ação é criação de perfis de ataque (ou mesmo novos módulos de operação) da ferramenta DNA que simulem o comportamento de uma gramática probabilística fornecida.

Na linha de probabilidade e estatística, pode-se desenvolver e testar cadeias de Markov para o idioma português, comparando os resultados com as implementações do JtR e do DNA.

Na área de desenvolvimento de sistemas, pode-se criar uma ferramenta que gere diretamente DPPs a partir do fornecimento dos dados personalizados (dados biográficos e palavras-chave do caso). O aplicativo poderia fornecer um número de senhas escolhido pelo usuário e possuir diversas gramáticas previamente treinadas e configuradas.

Do total de senhas do grupo de controle, 15% não foram descobertas com nenhum dos métodos empregados. Essas senhas são, provavelmente, longas e devem conter muitos símbolos. Desenvolver um método para descobri-las seria de grande utilidade para a perícia e para a comunidade científica.

Finalmente, pode-se ainda testar o método de DPPs em casos reais.



## REFERÊNCIAS BIBLIOGRÁFICAS

AccessData Corp. (2010). AccessData PRTK 6.5 & DNA 3.5 User Guide, Lindon, Utah, EUA: AccessData.

Adams A. & Sasse M. (1999). Users are not the enemy. In *Communications of the ACM*, v. 42, n. 12, p. 40-46, New York, NY, EUA: ACM.

Almasi G. & Gottlieb A. (1994). Highly Parallel Computing. Segunda edição: The Benjamin Cummings Publishing Company Inc.

Bengtsson J. (2007). Parallel password cracker: A feasibility study of using Linux clustering technique in computer forensics. In *Proceeding of the Second International Workshop on Digital Forensics and Incident Analysis (WDFIA 2007)*: IEEE.

Blakstad J., Nergard R., Jaatun M. & Gligoroski D. (2009). All in a day's work: Password cracking for the rest of us. In *Proceedings of the Norwegian Information Security Conference (NISK 2009)*, Noruega.

Böhm M., Leimeister S., Riedl C. & Krcmar H. (2010). Cloud Computing and Computing Evolution. In *Cloud Computing Technologies Business Models Opportunities and Challenges*: CRC Press.

Bonneau J. & Preibusch S. (2010). The password thicket: technical and market failures in human authentication on the web. In *The Ninth Workshop on the Economics of Information Security (WEIS 2010)*, EUA.

Bonneau J., Just M., & Matthews G. (2010). What's in a Name? Evaluating statistical attacks on personal knowledge questions. In *Proceedings of the 14<sup>th</sup> International Conference on Financial Cryptography and Data Security (FC 2010)*, Tenerife, Espanha.

Brostoff S. & Sasse M. (2000). Are Passfaces more Usable than passwords? A field trial investigation. In *Proceedings of People and Computers XIV - Usability or Else! (HCI 2000)*, p. 405-424: Springer.

Brown A., Bracken E., Zoccoli S. & Douglas K. (2004). Generating and remembering passwords. In *Applied Cognitive Psychology*, v. 18, p. 641-651: Wiley InterScience.

Burr W., Dodson D., & Polk W. (2006). NIST Special Publication 800-63 Version 1.0. 2 - Electronic Authentication Guideline, *Computer Security Division, Information Technology Laboratory*, National Institute of Standards and Technology, Gaithersburg, MD, EUA: NIST.

Candia A. (2008). Um sistema distribuído de criptoanálise computacional para uso forense. Dissertação de Mestrado, Departamento de Engenharia de Produção, Universidade Federal de Santa Maria.

Casey E. & Stellatos G. (2008). The impact of full disk encryption on digital forensics. In *ACM SIGOPS Operating Systems Review*, v. 42, n. 3, p. 93-98, EUA: ACM.

- Casey E. (2002). Practical approaches to recovering encrypted digital evidence. In *International Journal of Digital Evidence*, v. 1, n.3, EUA.
- Chiasson S., Forget A., Biddle R. & Oorschot P.C. van (2008). Influencing users towards better passwords: persuasive cued click-points. In *Proceedings of the 22<sup>nd</sup> British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction (BCS-HCI 2008)*, British Computer Society Swinton, Reino Unido: ACM.
- Conklin A., Dietrich G. & Walz D. (2004). Password-based authentication: a system perspective. In *Proceeding of the 37<sup>th</sup> Annual Hawaii International Conference on System Sciences*, pp. 170-179, EUA: IEEE.
- Dandass Y. (2008). Using FPGAs to Parallelize Dictionary Attacks for Password Cracking. In *Proceedings of the 41<sup>st</sup> Annual Hawaii International Conference on System Sciences (HICSS 2008)*: IEEE.
- De Luca A., Denzel M. & Hussmann H. (2009). Look into my eyes! Can you guess my Password? In *Proceedings of the 5<sup>th</sup> Symposium on Usable Privacy and Security (SOUPS 2009)*, Pittsburgh, PA, EUA: ACM.
- Dell'Amico M., Michiardi P. & Roudier Y. (2010). Password strength: an empirical analysis. In *Proceedings of 2010 IEEE International Conference on Computer Communications (INFOCOM 2010)*, EUA: IEEE.
- ElcomSoft Co. Ltd. (2011). ElcomSoft Distributed Password Recovery – Online Documentation [<http://www.elcomsoft.com/help/edpr/index.html>].
- Etter B. (2001). The forensic challenges of e-crime. In *7<sup>th</sup> Indo-Pacific Congress on Legal Medicine and Forensic Sciences*, Australasian Centre for Policing Research, Austrália: NCJRS.
- Florêncio D. & Herley C. (2007). A large-scale study of web password habits. In *Proceedings of the 16<sup>th</sup> International Conference on World Wide Web (WWW 2007)*, p. 657-665, Banff, Alberta, Canadá: ACM.
- Florêncio D., Herley C. & Coskun B. (2009). Do strong web passwords accomplish anything?. In *Proceedings of the 2<sup>nd</sup> USENIX Workshop on Hot Topics in Security (HOTSEC 2007)*, USENIX Association Berkeley, EUA: ACM.
- Flynn M. (1972). Some Computer Organizations and Their Effectiveness. In *IEEE Transactions on Computers*, v. c-21, n. 9, p. 948-960: IEEE.
- Forget A. (2008). Helping users create and remember more secure text passwords. In *Proceedings of the 22<sup>nd</sup> British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction (BCS-HCI 2008)*, v. 2, Swinton, Reino Unido: British Computer Society.
- Forget A., Chiasson S. & Biddle R. (2007). Helping users create better passwords: Is this the right approach?. In *Proceedings of the 3<sup>rd</sup> Symposium on Usable Privacy and Security (SOUPS 2007)*, EUA: ACM.

- Forget A., Chiasson S., Oorschot P.C. van & Biddle R. (2008). Improving text passwords through persuasion. In *Proceedings of the 4<sup>th</sup> Symposium on Usable Privacy and Security (SOUPS 2008)*, EUA: ACM.
- Fragkos G. & Tryfonas T. (2007). A Cognitive Model for the Forensic Recovery of End-User Passwords. In *Proceedings of the Second International Workshop on Digital Forensics and Incident Analysis (WDFIA 2007)*, p. 48-54: IEEE.
- Fragkos G., Xyno K. & Blyth A. (2005). The use of computers idle-time and parallel processing over a network to perform password threat assessment. In *Proceedings of the 4<sup>th</sup> European Conference on Information Warfare and Security (ECIW 2005)*, p. 99-105: Academic Conferences.
- Frykholm N. & Juels A. (2001). Error-Tolerant Password Recovery In *Proceedings of the 8<sup>th</sup> ACM Conference on Computer and Communications Security (CCS 2001)*, EUA: ACM.
- Gaw S. & Felten E. (2006). Password management strategies for online accounts. In *Proceedings of the 2<sup>nd</sup> Symposium on Usable Privacy and Security (SOUPS 2006)*, EUA: ACM.
- Gehring E. (2002). Choosing passwords: security and human factors. In *IEEE 2002 International Symposium on Technology and Society (ISTAS 2002)*, p. 369-373, EUA: IEEE.
- Glodek W. (2008). Using a Specialized Grammar to Generate Probable Passwords. Tese de Mestrado. Departamento de Ciência da Computação, Universidade do Estado da Flórida, EUA.
- Grawemeyer B. & Johnson H. (2009). How secure is your password? Towards modelling human password creation. In *Proceedings of the The First Trust Economics Workshop*, University College London, London, Reino Unido.
- Güneysu T. (2009). Cryptography and Cryptanalysis on Reconfigurable Devices. Tese de Doutorado, Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Alemanha.
- Malone D. & Maher K. (2011). Investigating the Relationship Between Password Distribution and Zipf's Law. Hamilton Institute, NUI Maynooth. National University of Ireland, Irlanda.
- Güneysu T., Kasper T., Novotný M. & Paar C. (2008). Cryptanalysis with COPACOBANA. In *IEEE Transactions on Computers*, vol. 57, n. 11, p. 1498-1513: IEEE.
- Hanawal M & Sundaresan R. (2010). Randomised attacks on passwords, *Program On Advanced Research in Mathematical Engineering*, Department of Electrical Communication Engineering, Indian Institute of Science, Índia.
- Hart D. (2008). Attitudes and practices of students towards password security. In *Journal of Computing Sciences in Colleges*, v. 23, p. 169-174, EUA: CCSC.

- Hayashi E. & Hong J. (2011). A diary study of password usage in daily life. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI 2011)*, Vancouver, BC, Canadá: ACM.
- Helkala K. & Snekkenes E. (2009). Password generation and search space reduction. In *Journal of Computers*, v. 4, p. 663-669: Academy Publisher.
- Hellman M. (1980). A cryptanalytic time-memory trade-off, In *IEEE Transactions on Information Theory*, v. 26, n. 4, p. 401-406: IEEE.
- Herbordt M., Vancourt T., Gu Y., Sukhwani B., Conti A., Model J. & Disabello D. (2007). Achieving High Performance with FPGA-Based Computing. In *Computer*, v. 40, n. 3, p. 50-57: IEEE.
- Hu G., Ma J. & Huang B. (2009). Password Recovery for RAR Files Using CUDA. In *Proceeding of the Eighth IEEE International Conference on Dependable Autonomic and Secure Computing (DASC 2009)*, p. 486-490: IEEE.
- Inglesant P. & Sasse M. (2010). The true cost of unusable password policies: password use in the wild. In *Proceedings of the 28<sup>th</sup> International Conference on Human Factors in Computing Systems (CHI 2010)*, EUA: ACM.
- Ives B., Walsh K. & Schneider H. (2004). The domino effect of password reuse. In *Communications of the ACM - Human-computer Etiquette*, v. 47, n. 4, p. 75-78: New York, NY, EUA: ACM.
- Jacob B., Brown M., Fukui K. & Trivedi N. (2005). Introduction to Grid Computing. IBM International Technical Support Organization: IBM.
- Jean-Jacques Q., Francois-Xavier S., Rouvroy G., Jean-Pierre D. & Jean-Didier L. (2002). A Cryptanalytic Time-Memory Tradeoff: First FPGA Implementation. In *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, Lecture Notes in Computer Science, p. 780-789: Springer Berlin/Heidelberg.
- Jermyn I., Mayer A., Monroe F., Reiter M. & Rubin A. (1999). The Design and Analysis of Graphical Passwords. In *8<sup>th</sup> USENIX Security Symposium*, Washington, DC, EUA.
- Kedem G. & Ishihara Y. (1999). Brute force attack on UNIX passwords with SIMD computer. In *Proceedings of the 8<sup>th</sup> Conference on USENIX Security Symposium (SSYM 1999)*, v.8: ACM.
- Kleinhans H., Butts J. & Sheno S. (2009). Password Cracking Using Sony Playstations. *IFIP Advances in Information and Communication Technology*, v. 306, p. 215-227: Springer Berlin/Heidelberg.
- Komanduri S. & Hutchings D. (2008). Order and entropy in picture passwords. In *Proceedings of Graphics Interface Conference (GI 2008)*, Canadian Information Processing Society, p. 115–122: ACM.
- Komanduri S., Shay R., Kelley P., Mazurek M., Bauer L., Christin N., Cranor L. & Egelman S. (2011). Of Passwords and People: Measuring the Effect of Password-

Composition Policies. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI 2011)*, Vancouver, BC, Canadá: ACM.

Krauter K., Buyya R. & Maheswaran M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. In *Software: Practice and Experience*, v. 32, n. 2, p. 135-164: John Wiley & Sons, Ltd.

Kumar M., Garfinkel T., Boneh D. & Winograd T. (2007). Reducing shoulder-surfing by using gaze-based password entry. In *Proceedings of the 3<sup>rd</sup> Symposium on Usable Privacy and Security (SOUPS 2007)*, EUA: ACM.

Kuo C., Romanosky S. & Cranor L. (2006). Human selection of mnemonic phrase-based passwords. In *Proceedings of the 2<sup>nd</sup> Symposium on Usable Privacy and Security (SOUPS 2006)*, EUA: ACM.

Kuppens L. (2010). Soluções para acelerar o processo de quebra de senhas: Estudo comparativo entre GPU, FPGA, Grid e Rainbow Tables. Monografia de Especialização, Criptografia e Segurança em Redes, Universidade Federal Fluminense, Brasil.

Lach J. (2010). Using mobile devices for user authentication. In *Communications in Computer and Information Science*, vol. 79, p. 263-268: Springer.

Lin D., Dunphy P., Olivier P. & Yan J. (2007). Graphical passwords & qualitative spatial relations. In *Proceedings of the 3<sup>rd</sup> Symposium on Usable Privacy and Security (SOUPS 2007)*, EUA: ACM.

Lowman S. (2010). The Effect of File and Disk Encryption on Computer Forensics, 2010.

Marechal S. (2007). Advances in password cracking. In *Journal in Computer Virology*, v. 4, n. 1, p. 73-81, França: Springer-Verlag.

Mentens N., Batina L., Preneel B. & Verbauwhede I. (2006). Time-Memory Trade-Off Attack on FPGA Platforms: UNIX Password Cracking. In *Proceedings of the International Workshop on Applied Reconfigurable Computing (ARC 2006)*, LNCS, v. 3985, p. 323-334: Springer.

Morris R. & Thompson K. (1979). Password security: A case history. In *Communications of the ACM*, v. 22, n. 11, p. 594-597, New York, NY, EUA: ACM.

Narayanan A. & Shmatikov V. (2005). Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12<sup>th</sup> ACM Conference on Computer and Communications Security (CCS 2005)*, EUA: ACM.

Oechslin P. (2003). Making a faster cryptanalytic time-memory trade-off. In the *23<sup>rd</sup> Annual International Cryptology Conference (CRYPTO 2003)*, Advances in Cryptography, v. 2729, p. 617-630: Springer Berlin/Heidelberg.

O’Gorman L., Bagga A. & Bentley J. (2005). Query-directed passwords. In *Computers & Security*, vol. 24, 2005, p. 546-560, EUA: Elsevier.

Openwall Project (2011). John the Ripper Password Cracker – Online Documentation [<http://www.openwall.com/john/doc/>].

- Owens J., Houston M., Luebke D., Green S., Stone J. & Phillips J. (2008). GPU Computing. In *Proceedings of the IEEE*, v. 96, n. 5, p. 879-899: IEEE.
- Passware Inc. (2011). Passware Kit Forensic – Online Documentation, [<http://www.lostpassword.com/kit-forensic.htm>].
- Phong P., Dung P., Tan D., Duc N. & Thuy N. (2010). Password recovery for encrypted ZIP archives using GPUs. In *Proceedings of the 2010 Symposium on Information and Communication Technology (SoICT 2010)*: ACM.
- Pinkas B. & Sander T. (2002). Securing passwords against dictionary attacks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, EUA: ACM.
- Rabiner L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, v. 77, n. 2, p. 257-286: IEEE.
- Recordon D. & Reed D. (2006). OpenID 2.0: a platform for user-centric identity management. In *Proceedings of the Second ACM Workshop on Digital Identity Management (DIM 2006)*, p. 11-16, EUA: ACM.
- Riley S. (2006). Password security: what users know and what they actually do. In *Usability News*, v. 8, n.1, Software Usability Research Laboratory (SURL), Wichita State University, EUA.
- Roth T. (2011). Breaking encryption in the cloud: GPU accelerated supercomputing for everyone. In *Black Hat Technical Security Conference: Europe 2011*.
- Ryoo S., Rodrigues C., Bagsorkhi S., Stone S., Kirk D. & Hwu W-mei. (2008). Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *Proceedings of the 13<sup>th</sup> ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2008)*, New York, NY, EUA: ACM.
- Schechter S., Brush A. & Egelman S. (2009). It's No Secret: Measuring the security and reliability of authentication via 'secret' questions. In *Proceedings of the 30<sup>th</sup> IEEE Symposium on Security and Privacy (SP 2009)*, pp. 375-390, EUA: ACM.
- Schechter S., Herley C. & Mitzenmacher M. (2010). Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. In *Proceedings of the 5<sup>th</sup> USENIX Conference on Hot Topics in Security (HotSec 2010)*, EUA: ACM.
- Schneier B. (2000). *Secrets and Lies: Digital Security in a Networked World*. New York, NY, EUA: John Wiley & Sons.
- Shannon C. (1948). The mathematical theory of communication. 1963. In *The Bell System Technical Journal*, v. 27, n. 4, EUA.
- Shannon C. (1951). Prediction and entropy of printed English. In *Bell System Technical Journal*, vol. 30, p. 50-64, EUA.

- Shay R., Bhargav-Spantzel A. & Bertino E. (2007). Password Policy Simulation and Analysis. In *Proceedings of the 3<sup>rd</sup> ACM Workshop on Digital Identity Management (DIM 2007)*, Alexandria, Virginia, EUA: ACM.
- Shay R., Komanduri S., Kelley P., Leon P., Mazurek M., Bauer L., Christin N. & Cranor L. (2010). Encountering stronger password requirements: User attitudes and behaviors. In *Proceedings of the Sixth Symposium on Usable Privacy and Security (SOUPS 2010)*, EUA: ACM.
- Someren N. van (2007). RIPA part III - The intricacies of decryption. In *Digital Investigation*, v. 4, p. 113-115, EUA: Elsevier.
- St. Clair L., Johansen L., Enck W., Pirretti M., Traynor P., McDaniel P. & Jaeger T. (2006). Password exhaustion: predicting the end of password usefulness, *Information Systems Security*, Lecture Notes in Computer Science, v. 4332, p. 37-55: Springer Berlin/Heidelberg.
- Stajano F. (2011). Pico: No more passwords! In *Proceedings of Nineteenth International Workshop on Security Protocols*, Cambridge, Inglaterra, Reino Unido.
- Stubblebine S. & Oorschot P.C. van (2004). Addressing online dictionary attacks with login histories and humans-in-the-loop. In *Financial Cryptography 2004*, p. 39-53: Springer-Verlag LNCS.
- Theoharoulis K., Manifavas C. & Papaefstathiou I. (2009). High-End Reconfigurable Systems for fast Windows' Password Cracking. In *17th IEEE Symposium on Field Programmable Custom Computing Machines (FCCM 2009)*: IEEE.
- Thing V & Ying H.-M. (2009). A novel time-memory trade-off method for password recovery. In *Digital Investigation*, v. 6, p. 114-120, EUA: Elsevier.
- Thing V. (2010). Virtual Expansion of Rainbow Tables. In *Proceedings of Sixth International Conference on Digital Forensics (IFIP 2010)*, Hong Kong: Springer.
- Thorpe J. & Oorschot P.C. van (2004). Graphical dictionaries and the memorable space of graphical passwords. In *Proceedings of the 13<sup>th</sup> Conference on USENIX Security Symposium (SSYM 2004)*, San Diego, CA, EUA: ACM.
- Topkara U., Atallah M. & Topkara M. (2007). Passwords decay, words endure: Secure and re-Usable multiple password mnemonics. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC 2007)*, EUA: ACM.
- Vaquero L., Rodero-Merino L., Caceres J. & Lindner M. (2009). A Break in the Clouds: Towards a Cloud Definition. In *ACM SIGCOMM Computer Communication Review*, v. 39, n. 1, p. 50-55: ACM.
- Vu K.-P., Proctor R., Bhargav-Spantzel A., Tai B.-L., Cook J. & Schultz E. (2007). Improving password security and memorability to protect personal and organizational information. In *International Journal of Human-Computer Studies*, v. 65, n. 08, p. 744-757: Elsevier.

- Weedbrook C. & Gu M. (2005). Quantum Passwords, Department of Physics, arXiv:quant-ph/0506255v3, University of Queensland, Austrália.
- Weir M. (2010). Using Probabilistic Techniques to Aid in Password Cracking Attacks. Tese de Doutorado, Departamento de Ciência da Computação, Universidade do Estado da Flórida, EUA.
- Weir M., Aggarwal S., Collins M., & Stern H. (2010). Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17<sup>th</sup> ACM Conference on Computer and Communications Security (CCS 2010)*, pp. 162-175, EUA: ACM.
- Weir M., Aggarwal S., Medeiros B. de, & Glodek B. (2009). Password cracking using probabilistic context-free grammars. In *Proceedings of the 30<sup>th</sup> IEEE Symposium on Security and Privacy (SP 2009)*, pp. 391-405, EUA: IEEE.
- Weiss R. & De Luca A. (2008). PassShapes: utilizing stroke based authentication to increase password memorability. In *Proceedings of the 5<sup>th</sup> Nordic Conference on Human-computer Interaction: Building Bridges (NordiCHI 2008)*, EUA: ACM.
- Wollinger T., Guajardo J. & Paar C. (2004). Security on FPGAs: State-of-the-art implementations and attacks. In *ACM Transactions on Embedded Computing Systems (TECS 2004)*, v. 3, n. 3, EUA:ACM.
- Yan J., Blackwell A., Anderson R. & Grant A. (2000). The memorability and security of passwords - Some empirical results. In *Technical Reports by the University of Cambridge Computer Laboratory*, UCAM-CL-TR-500, Cambridge, Reino Unido: UCAM.
- Yan J., Blackwell A., Anderson R. & Grant A. (2004). Password memorability and security: empirical results. In *IEEE Security Privacy Magazine*, v. 2, p. 25-31: IEEE.
- Yee K.-P. & Sitaker K. (2006). Passpet: Convenient Password Management and Phishing Protection. In *Proceedings of the 2<sup>nd</sup> Symposium on Usable Privacy and Security (SOUPS 2006)*, EUA: ACM.
- Zhang J., Luo X., Akkaladevi S. & Ziegelmayer J. (2009). Improving multiple-password recall: an empirical study. *European Journal of Information Systems*, DOI 10.1057: EJIS.
- Zhang Y., Monroe F. & Reiter M. (2010). The security of modern password expiration: an algorithmic framework and empirical analysis. In *Proceedings of the 17<sup>th</sup> ACM Conference on Computer and Communications Security (CCS 2010)*, p. 176-186, EUA: ACM.
- Zonenberg A. (2009). Distributed hash cracker: A cross-platform GPU-accelerated password recovery system. Rensselaer Polytechnic Institute, Troy, NY, EUA.