Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Forense computacional em discos de estado sólido: desafios e perspectivas

João Vitor A. Ribeiro

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Ms. João José Costa Gondim

Brasília
2016

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Forense computacional em discos de estado sólido: desafios e perspectivas

João Vitor A. Ribeiro

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Ms. João José Costa Gondim (Orientador)
CIC/UnB

Prof. Dr. Robson de Oliveira Albuquerque
ENE/UnB

Prof. Dr. Camilo Chang Dórea
CIC/UnB

Prof. Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 4 de julho de 2016

Dedicatória

Dedico este trabalho á minha mãe, Marcia de Melo Assis.

Agradecimentos

Agradeço a Deus pelo dom da vida. A minha mãe pelo apoio incondicional durante essa jornada. Aos meus familiares e amigos pelo amparo. Ao meu orientador Prof. João Gondim pelos conhecimentos passados. Aos professores e funcionários da Universidade de Brasília.

Resumo

Este trabalho apresenta uma análise da usabilidade e da eficácia de técnicas usuais de recuperação de dados e de forense digital em um SSD - disco de estado sólido utilizando ferramentas usuais de recuperação de dados. É feito um estudo de caso em um SSD utilizando ferramentas usuais de recuperação de dados, *data carving* e informações pertinentes a tecnologia dos dispositivos e sistemas de arquivos. Também é realizada uma análise comparativa com os resultados da bibliografia. Os resultados obtidos sugerem que as tecnologias utilizadas em SSDs podem dificultar ou até inviabilizar a análise forense nos Sistemas Operacionais *Windows* e *Linux*.

Palavras-chave: SSD, Forense Digital, Exclusão de Arquivos, Sistemas de Arquivos, Sistemas Operacionais

Abstract

An analysis of the usability and the effectiveness of the standard practices in Data Recovery and Digital Forensics in a SSD - solid state disk using common data and file recovery tools is presented. The process of recovering data from a SSD using common tools of Data and File Recovery, *Data Carving* and with background knowledge in File Systems and in SSD technology, provided the basis for a comparative analysis with the results found in the bibliography. Results suggest that the technology used in SSDs can hinder or even prevent Forensic Analysis on *Linux* and *Windows* systems.

Keywords: SSD, Digital Forensics, File Deletion, File Systems, Operating Systems

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nos últimos tempos, tem-se verificado um crescimento no uso de computadores, *tablets*, *smartphones* assim como diversos outros sistemas computacionais. A tecnologia está cada vez mais presente nas interações sociais, assim como na esfera criminal, onde sistemas computacionais são utilizados para cometer ou mediar crimes. Nesse contexto, surge a atuação da Forense Digital.

Carrier [7], define uma investigação digital como um processo no qual desenvolvem-se e testam-se hipóteses para responder questões sobre eventos ocorridos em um contexto virtual. Carrier ilustra essa ideia com uma situação hipotética, na qual um servidor foi comprometido. É iniciada uma investigação para determinar como se deu esse fato assim como o seu responsável. São encontrados dados como entradas de eventos de *log*, ferramentas de ataque computacional assim como são identificadas diversas vulnerabilidades no próprio servidor. Com base nessas informações, é elaborada uma hipótese sobre qual(is) vulnerabilidade(s) o atacante utilizou para obter acesso e controle do servidor e o que foi feito após o ataque. As entradas de *log* são analisadas, configurações de rede e *firewall* são verificadas e a partir dessa averiguação a hipótese inicial levantada é confirmada ou refutada.

Uma investigação digital pode ter como produto um conjunto de evidências, as quais podem ou não configurar-se evidências em termos legais, a depender da admissibilidade das provas em um tribunal. Dessa forma, surge a definição de *forense computacional* ou *forense digital*, na qual Carrier define como o processo no qual se faz uso de meios científicos e tecnológicos para conduzir uma investigação digital para analisar *objetos digitais*.

No que se refere a esses objetos, alvos de uma investigação forense, como, por exemplo, aqueles objetos apreendidos ou encontrados em uma cena de um crime, encontram-se os computadores, sejam eles os de uso pessoal (*personal computer - PC*), servidores, *workstations*, entre outros. Esses computadores possuem componentes de armazenamento permanente chamados de memória secundária. Um tipo bastante comum de dispositivo de memória secundária é o disco rígido conhecido popularmente como HDD - *Hard Disk Drive*. Nos últimos anos, entretanto, por conta de fatores como menor preço, maiores velocidades de leitura e escrita, maior confiabilidade, entre outros, os discos de estado sólido - (SSD) *Solid State Disk* têm ganhado cada vez mais espaço e uso em um mercado de predominância dos discos rígidos [45] [41] [48].

Os discos de estado sólido no entanto apresentam diversos desafios à forense computacional em comparação com os discos rígidos tradicionais. Entre eles, mecanismos internos que podem dificultar ou inviabilizar a recuperação de dados excluídos nessas mídias, podendo levar à perda de informação que pode constituir potenciais evidências. São dois os principais fatores que podem levar a essa perda: o processo de coleta de lixo e o comando TRIM. Isso se deve em parte por conta do fato de os SSDs poderem fazer uso desses mecanismos de forma autônoma, ou seja, sem a necessidade da ação de um Sistema Operacional ou do usuário, sendo também possível que esses mecanismos sejam acionados pelos agentes citados, conforme explica Bell & Boddington em seu artigo intitulado *Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery?* em tradução livre: "Discos de Estado Sólido: O começo do fim para a prática atual em Recuperação Forense Digital" [2].

1.1 Objetivos

Este trabalho almeja realizar um apanhado do estado da arte da forense computacional em discos de estado sólido, assim como verificar em um SSD a influência de seus mecanismos internos para a forense digital dessas mídias, e por fim realizar uma análise comparativa com os resultados mais recentes da área.

O foco do trabalho está na análise da recuperabilidade de arquivos excluídos nos sistemas *Windows* e *Linux*, nos quais a recuperação dos arquivos pode ser afetada pelos mecanismos citados anteriormente. Dessa forma, objetiva-se a realização de experimentos de recuperabilidade de arquivos nos dois sistemas, a partir de imagens de disco do SSD, utilizando ferramentas comuns da área de Forense Computacional como *extundelete*, *foremost*, *scalpel*, entre outras.

Foram pensados cenários que envolvem a exclusão e a recuperação de arquivos, e com base nesses cenários foram propostos a realização de experimentos para verificar como se dá essa recuperação.

1.2 Trabalhos relacionados

Antonellis [1] em 2008 realiza um experimento de exclusão de imagens e de um arquivo de texto, sob o ambiente *Windows* com o Sistema de Arquivos **NTFS** obtendo apenas arquivos com 0s. O autor não menciona a presença ou a influência do comando TRIM, assim como também não menciona a marca e o modelo do SSD.

Bell e Boddington em 2010 [2] realizam quatro tipos de experimentos em um SSD modelo P64 de 64GB da Corsair:

- O primeiro é para verificar se os HDDs e os SSDs apresentam similaridades quanto a remanência de dados após a exclusão. Nesse experimento verificaram a influência das rotinas de *coleta de lixo*, que entrou em ação 3 minutos após a formatação rápida de um SSD inteiramente escrito com arquivos de texto. O disco foi apagado completamente após 3 minutos, aproximadamente. Utilizaram programas próprios para amostragem e verificação da remanência dos dados;
- No segundo experimento, empregam uma metodologia forense, utilizando um *write-blocker* e *disk imaging*. Nesse experimento conseguem recuperar 1090 de 316666 arquivos, dentre esses, nenhum intacto, apresentando uma taxa de perda de 90.8% do conteúdo dos arquivos;
- No terceiro, almejaram verificar a influência do processo de medição do primeiro experimento, porém obtiveram resultados similares a esse;
- Já no quarto experimento, desejavam verificar se um *write-blocker* consegue influenciar o comportamento interno de um SSD. Reportam resultados de 18.74% e 14.8% de perda dos dados em duas situações diferentes. Embora o uso do *write-blocker* não tenha impedido a exclusão definitiva dos dados após uma rápida formatação, conseguiu diminuir consideravelmente seus efeitos, dada a redução da taxa de perda do conteúdo dos arquivos.

Os testes de Bell e Boddington foram realizados no SO *Windows XP SP3*, que não possui suporte ao TRIM de forma nativa, requerendo programas de terceiros, usualmente do próprio fabricante, nas chamadas *toolboxes*.

Gebremaryam em 2011 [22] realiza um experimento no *Windows 7* com o comando TRIM habilitado, após uma formatação rápida em três modelos de SSDs: Crucial M4, Kingston SSDnow V 100 e Samsung S470. Obtendo os resultados da Figura 1.1:

SSD Type	**# of files stored**	**# of files recovered**	**Time consumed for recovery**
Crucial M4 (TRIM)	1522	0	1:15 hr
Samsung 470 series (TRIM)	1522	0	1:16 hr
Kingston SSDnow V 100	1522	1383	1:20 hr

Figura 1.1: Resultados obtidos por Gebremaryam para o comando TRIM [22]

Também realizam um experimento com vistas a determinar a influência do processo de coleta de lixo também após uma formatação rápida nos modelos mencionados assim como em um HDD, obtendo os resultados de acordo com a Figura 1.2:

Drive Type	**# of files stored**	**# of files recovered**
HDD	4,739	4,722
Crucial M4 SSD	17,627	17,625
Kingston SSDnow V 100	17,627	17,625
Samsung 470 series	17,627	17,625

Figura 1.2: Resultados obtidos por Gebremaryam para a coleta de lixo [22]

Os resultados de Gebremaryam mostram a influência ínfima da coleta de lixo na recuperação dos dados excluídos.

King e Vidas em 2011 [31] realizam uma ampla análise empírica, de diferentes cenários de utilização do disco (baixa, alta, e formatado), diferentes tamanhos de arquivo (pequeno e grande), sob a presença ou não do TRIM para SSDs de diferentes marcas: Imation, Corsair, OCZ, Kingston, Intel, entre outras, totalizando 16 dispositivos diferentes. Os resultados foram variáveis. No *Linux* (Ubuntu 9.04), que não suporta o TRIM nativamente, a porcentagem de blocos recuperados varia de 0 até 100%.

Já no *Windows 7*, relatam os resultados da Figura 1.3.

Bonetti et al. em 2013 [21] menciona que o TRIM funciona de forma diferente nos casos de formatação rápida (*quick format*) e nos casos de deleção de arquivos. Nos primeiros, o SO supostamente informa que todo o SSD pode ser "podado" pelo TRIM. Bonetti et al.

Table 2 – Percent blocks recovered on a TRIM-enabled disk with Windows 7

Test	Control	Intel1	Intel2	Intel3	OCZ1	Corsair1	Crucial1
Large File, Low Usage	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Large File, High Usage	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Large File, Format	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Small File, Low Usage	99.98%	0.00%	0.00%	0.00%	25.53%	25.53%	27.54%
Small File, High Usage	99.98%	0.00%	0.00%	0.00%	25.53%	27.54%	26.28%
Small File, Format	0.00%	0.00%	0.00%	0.00%	0.00%	24.46%	0.00%

This chart shows only those disks that support TRIM (and the control), which is 6 of the 16 tested.

Figura 1.3: Resultados obtidos por King e Vidas [31]

realiza experimentos em três modelos de SSD: Samsung S470, Crucial M4 e Corsair F60 e nos sistemas *Windows* e *Linux*:

- **TRIM**: Utilizando metodologia similar a de Bell e Boddington, verificaram a influência do comando TRIM na formatação rápida e na exclusão de arquivos no *Windows*. No SSD Samsung S470, o disco foi apagado em 10 segundos na formatação rápida e os setores, na exclusão de arquivos, em 5. No Crucial M4, esse foi apagado antes mesmo da notificação pelo SO, e na exclusão de arquivos, levou 10 segundos. Já o Corsair F60 apresentou comportamento diferenciado. Benetti et al. relatam que após uma formatação rápida, apenas uma porcentagem do disco é apagado e que essa porcentagem é proporcional ao espaço total utilizado conforme mostra a Figura 1.4. No caso da exclusão de arquivos esse SSD também apresentou resultados variáveis, onde arquivos são efetivamente excluídos em 3 segundos ou nem o são, podendo ser recuperados. Já no *Linux* (Ubuntu 12.04), relatam que após a formatação rápida todos os SSDs foram apagados em aproximadamente 15 segundos, explicado pelo fato de em todos eles o *Linux* utilizar o mesmo *driver* AHCI - *Advanced Host Controller Interface* para gerenciar os dispositivos. Na exclusão de arquivos no SSD da Samsung nenhum arquivo foi apagado. No modelo da Crucial, os blocos são apagados somente no momento da desmontagem. Por fim, no SSD da Corsair todos os arquivos foram excluídos.

- **Coleta de lixo**: Utilizando Sistemas Operacionais e *drivers* que não suportam o TRIM, Bonetti et al. verificaram que após a exclusão de arquivos e horas de tempo ocioso, os SSDs da Samsung e Corsair não realizaram coleta de lixo, apesar de na sua especificação constar que o fazem.

- **Recuperabilidade de arquivos**: Experimentando com os diferentes modelos de SSD e com TRIM habilitado, no *Windows* (**NTFS**), obtiveram 0% de recuperabilidade nos SSDS da Samsung e da Crucial. No modelo da Corsair obtiveram 70.79% de recuperabilidade. Já no *Linux* (**ext4**), a recuperabilidade foi nula em todos os discos.

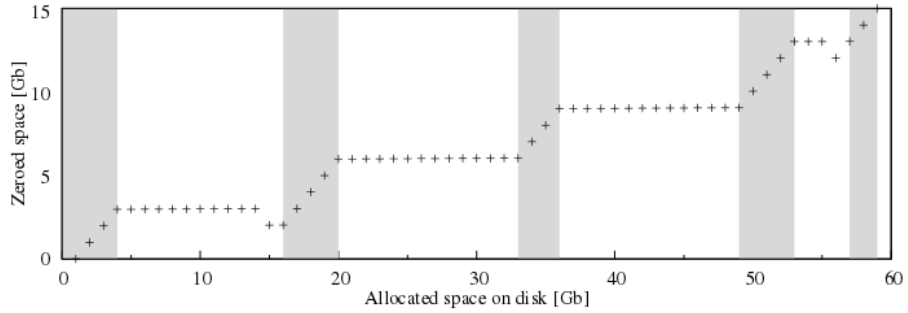


Figura 1.4: Resultados obtidos por Bonetti et al. que relacionam a quantidade de blocos excluídos com a porcentagem de uso do disco Corsair F60 [21]

- **Outros**: além dos fatores acima, Bonetti et al. realizam experimentos referentes a **padrões de apagamento**, **compressão** e **nivelamento de desgaste**.

Dos experimentos realizados por Bonetti et al., verifica-se uma particularidade da interação do SSD da Corsair com o Sistema de Arquivos **NTFS**, apresentando resultados significativamente diferenciados.

Nisbet et al. [38] analisam a retenção de dados nos sistemas *Windows 7* (**NTFS**), *Linux Ubuntu 11.10* (**ext4**) e *Mac OS X 10.7* (**HFS+**) em diferente situações (carga de trabalho ociosa e em atividade, uso baixo e alto de disco e com comando TRIM habilitado ou não). Além disso, testam a eficácia do TRIM manual como uma medida de anti-forense.

Os resultados obtidos por Nisbet et al. constam na Figura 1.5. Quanto ao uso do TRIM como uma medida anti-forense, os autores concluem que o TRIM por si só não é uma medida adequada para a sanitização de um SSD, relatando que em alguns casos, 185 a 214 MB remanesceram em um disco após a instrução do comando. [38]

Júnior e Ruy [19] em 2015 afirmam que foi possível recuperar arquivos excluídos em um SSD da marca ADATA, modelo Premier SP800 de 32GB que possui suporte ao TRIM. Os autores não entram em detalhes sobre até que ponto se dá essa recuperabilidade.

1.3 Estrutura do Trabalho

O trabalho foi estruturado da seguinte forma:

O Capítulo 2 aborda superficialmente os métodos e as estruturas de dados envolvidos nos Sistemas de Arquivos, assim como fornece uma explanação em alto nível dos elementos de sua organização em dispositivos de armazenamento.

Table 1: Percent of payload bytes retained after initial payload deletion

Table - Percent of Bytes retained after initial payload deletion during each test.												
	smallone.small			largeone.large			random.small			random.large		
Test Scenario	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+
Idle, Low Usage, TRIM	100.00 %	100.00 %	100.00 %	18.71 %	100.00 %	49.52 %	100.00 %	100.00 %	100.00 %	22.52 %	100.00 %	16.27 %
Idle, Low Usage, NOTRIM	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
Idle, High Usage, TRIM	100.00 %	100.00 %	100.00 %	11.83 %	100.00 %	20.05 %	100.00 %	100.00 %	100.00 %	6.73 %	100.00 %	6.16 %
Idle, High Usage, NOTRIM	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
Activity, Low Usage, TRIM	100.00 %	100.00 %	100.00 %	10.96 %	100.00 %	25.48 %	100.00 %	100.00 %	100.00 %	10.28 %	100.00 %	41.64 %
Activity, Low Usage, NOTRIM	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
Activity, High Usage, TRIM	100.00 %	100.00 %	100.00 %	18.90 %	100.00 %	42.99 %	100.00 %	100.00 %	100.00 %	10.58 %	100.00 %	24.26 %
Activity, High Usage, NOTRIM	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %

Table 2: Percent of payload bytes retained 1 hour after payload deletion

Table - Percent of Bytes retained after one hour from first extraction during each test.												
	smallone.small			largeone.large			random.small			random.large		
Test Scenario	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+
Idle, Low Usage, TRIM	0.00 %	100.00 %	0.00 %	0.16 %	100.00 %	0.15 %	0.39 %	100.00 %	0.39 %	0.55 %	100.00 %	0.58 %
Idle, Low Usage, NOTRIM	24.45 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
Idle, High Usage, TRIM	0.00 %	100.00 %	0.00 %	0.08 %	100.00 %	0.13 %	0.39 %	100.00 %	0.39 %	0.47 %	100.00 %	0.50 %
Idle, High Usage, NOTRIM	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
Activity, Low Usage, TRIM	0.00 %	100.00 %	0.00 %	0.14 %	100.00 %	0.16 %	0.39 %	100.00 %	0.39 %	0.58 %	100.00 %	0.50 %
Activity, Low Usage, NOTRIM	100.00 %	100.00 %	36.00 %	100.00 %	100.00 %	96.01 %	100.00 %	100.00 %	35.25 %	100.00 %	100.00 %	100.00 %
Activity, High Usage, TRIM	0.00 %	100.00 %	0.20 %	0.19 %	100.00 %	0.15 %	0.39 %	100.00 %	0.39 %	0.52 %	100.00 %	0.58 %
Activity, High Usage, NOTRIM	100.00 %	59.57 %	4.38 %	100.00 %	98.98 %	95.60 %	100.00 %	0.39 %	35.26 %	100.00 %	0.39 %	100.00 %

Table 3: Percent of payload bytes retained 5 hours after deletion

Table - Percent of Bytes retained after last extraction during each test.												
	smallone.small			largeone.large			random.small			random.large		
Test Scenario	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+	NTFS	Ext4	HFS+
Idle, Low Usage, TRIM	0.00 %	100.00 %	0.00 %	0.16 %	100.00 %	0.15 %	0.39 %	100.00 %	0.39 %	0.55 %	100.00 %	0.58 %
Idle, Low Usage, NOTRIM	24.45 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
Idle, High Usage, TRIM	0.00 %	100.00 %	0.00 %	0.08 %	100.00 %	0.13 %	0.39 %	100.00 %	0.39 %	0.47 %	100.00 %	0.50 %
Idle, High Usage, NOTRIM	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
Activity, Low Usage, TRIM	0.00 %	100.00 %	0.00 %	0.14 %	100.00 %	0.16 %	0.39 %	100.00 %	0.39 %	0.58 %	100.00 %	0.50 %
Activity, Low Usage, NOTRIM	0.37 %	0.36 %	1.28 %	100.00 %	99.40 %	87.34 %	1.01 %	100.00 %	0.38 %	100.00 %	100.00 %	100.00 %
Activity, High Usage, TRIM	0.38 %	100.00 %	0.22 %	0.19 %	100.00 %	0.16 %	0.39 %	100.00 %	0.39 %	0.52 %	100.00 %	0.58 %
Activity, High Usage, NOTRIM	4.30 %	59.57 %	0.23 %	100.00 %	97.56 %	87.25 %	0.58 %	0.40 %	6.37 %	100.00 %	100.00 %	100.00 %

Figura 1.5: Resultados obtidos por Nisbet et al [38]

O Capítulo 3 traz explicações sobre características fundamentais dos Discos de Estado Sólido, tratando dos mecanismos inerentes a esse tipo de mídia e das principais diferenças entre os HDDs e os SSDs.

O Capítulo 4 aborda os princípios, ferramentas, procedimentos e problemáticas inerentes às áreas de Recuperação Forense e Forense Computacional. Também aborda os desafios propostos pelos SSDs.

O Capítulo 5 trata dos experimentos realizados em um SSD, discutindo procedimentos, técnicas e resultados.

O Capítulo 6 por fim traz a conclusão do estudo realizado, assim como faz um levantamento dos pontos a serem considerados em trabalhos futuros.

Capítulo 2

Sistemas de Arquivos

Em um sistema computacional, dados podem ser armazenados em diferentes dispositivos físicos, como discos magnéticos (HDDs), discos de estado sólido (SSDs), disquetes, discos ópticos, fitas magnéticas entre outros. Para facilitar a manipulação desses dados em um computador foi concebido o conceito de **arquivo**. Um arquivo é uma coleção de dados relacionados entre si e é também uma unidade lógica de armazenamento, a qual é mapeada para um dispositivo físico.

Para que seja possível o armazenamento permanente assim como a recuperação de dados, os computadores necessitam de métodos para estruturar e organizar arquivos e fornecer uma visão lógica e uniforme do sistema de armazenamento. O conceito de **Sistema de Arquivos** também é utilizado para denominar o componente ou módulo do Sistema Operacional responsável pela criação, exclusão e gerência dos arquivos em modo geral.

Um **Sistema de Arquivos** define as regras gerais para as operações sobre os arquivos, gerência de espaço, restrições sobre os nomes dos arquivos, organização dos diretórios, metadados, controle de acesso, integridade, consistência, entre outros. A importância do Sistema de Arquivos como módulo de um Sistema Operacional reside na gerência dos dados do usuário e dos arquivos do próprio sistema.

2.1 Mídias de armazenamento

Uma mídia de armazenamento pode ser definida como um meio digital no qual dados são armazenados e operacionados. Cada tipo de mídia de armazenamento possui especificidades quanto aos aspectos citados anteriormente, como gerência de espaço e consistência dos dados, e por sua vez demandará o uso de um Sistema de Arquivos com certas carac-

terísticas. Nessa seção será feita uma introdução em alto nível dos discos magnéticos, dos discos de estado sólido e das mídias *Flash*. No Capítulo 3 serão detalhados os mecanismos inerentes aos discos de estado sólido e aos sistemas computacionais sobrejacentes a esse tipo de mídia de armazenamento.

2.1.1 Discos magnéticos

Um disco rígido é constituído essencialmente de uma pilha de pratos revestidos de uma superfície magnética e uma cabeça retrátil para leitura e escrita. A Figura 2.1 mostra a visão interna de um disco rígido.

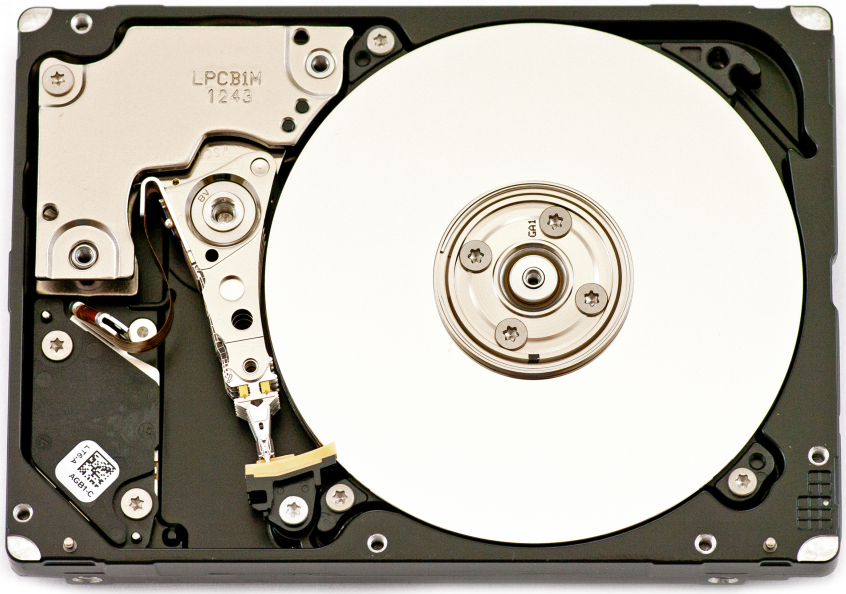


Figura 2.1: Estrutura interna de um disco SAS [39]

Conforme Patterson & Hennessy [43], cada disco possui duas superfícies magnéticas que podem ser escritas ou lidas. A pilha de discos é rotacionada a uma frequência que pode variar de 5400 a 15000 RPM (rotações por minuto). Cada disco possui diâmetro entre 1 e 3.5 polegadas, onde cada superfície de disco é dividida em círculos concêntricos chamados de **trilhas**.

Endereçamento CHS

Como explica Carrier [7], para cada trilha no disco rígido é atribuído um endereço de fora para dentro, começando com o endereço **0**. Como esse endereçamento é comum a todos os discos, utiliza-se o termo **cilindro** para se referir a todas as trilhas de um determinado endereço. Por exemplo, o cilindro de número **0** se refere a todas as trilhas, superiores

e inferiores, de endereço **0** de todos os discos. Uma **trilha** é dividida em **setores**. O setor é a menor unidade de armazenamento em um disco rígido, possuindo comumente 512 *bytes* ou 4096 *bytes*. Ao setor é atribuído um endereço na trilha, começando pelo endereço **1**.

Dentro do disco, a cabeça retrátil é utilizada para as operações de leitura e escrita. Para acessar um setor específico, utiliza-se o endereço do cilindro **C** (*cylinder*), o endereço da cabeça **H** (*head*) e por último o endereço do setor **S** (*sector*). Essa forma de endereçamento é conhecida como **CHS** - *Cylinder-head-sector*. A Figura 2.2 ilustra esse modelo.

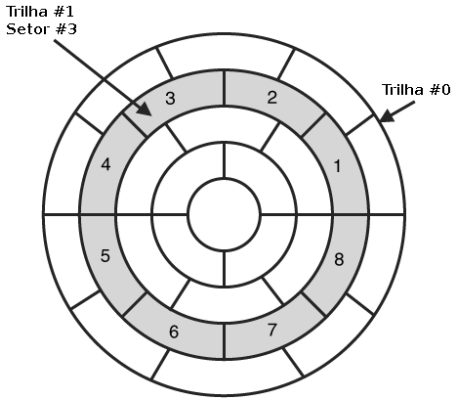


Figura 2.2: Estrutura CHS - *Cylinder-Head-Sector*; Adaptada de [29]

Em operações de escrita em um determinado bloco, o endereço desse é mapeado para certa região do disco e essa região é magnetizada de acordo com os dados a serem escritos, sobrescrevendo os dados previamente contidos nessa região.

Endereçamento LBA

O modo de endereçamento CHS apresenta algumas limitações. A especificação original ATA - *Advanced Technology Attachment* - definia 16 *bits* para endereçar o cilindro, 4 *bits* para a cabeça e 6 *bits* para o setor, porém as BIOSs - *Basic Input Ouput System* - antigas utilizavam 10, 8 e 6 *bits*, respectivamente, limitando o espaço endereçável a 504 MB. [7]

Para contornar essa limitação, foi criado um sistema de tradução nas BIOSs no qual endereços solicitados são mapeados para endereços da especificação. Esse método em si é limitado, dado que cada BIOS define uma quantidade possível de *bits* para endereçar cada componente (cabeça, cilindro, setor) apresentando uma geometria de disco diferente da existente fisicamente no disco. Historicamente, foram utilizados espaços endereçáveis

máximos de 2.1 GB, 3.2GB, 4.2GB, entre outros, dependentes da implementação de tal sistema de tradução por cada BIOS. [3]

O sistema CHS teve de ser abandonado, e foi concebido o modo de endereçamento LBA - *Logical Block Address*. Como o nome sugere, trata-se um endereçamento lógico de blocos, no qual não se utiliza uma tupla (cilindro, cabeça, setor), para endereçar blocos, mas apenas um número, tratando-se de um sistema linear. Com esse sistema, O Sistema Operacional e os *softwares* têm uma visão uniforme do armazenamento, dispensando a necessidade de se "saber" a organização física de um disco.

Para manter compatibilidade com certas mídias de armazenamento e certos sistemas de arquivos que utilizam o método CHS, é feita uma tradução dos endereços no último para o modo LBA, segundo a seguinte fórmula:

LBA = (((CILINDRO * cabeças_por_cilindro + CABEÇA) * setores_por_trilha) + SETOR - 1) [7]

2.1.2 Memórias Flash

Existem dois tipos principais de memórias *Flash*, são elas NOR e NAND [34]. A memória NOR suporta acesso aleatório em *bytes*, e é usada principalmente para armazenar código. Já a memória NAND é voltada para armazenar dados, possuindo maior densidade e sendo acessada na forma de setores. [4]

O tipo de memória *flash* NOR é ideal para aplicações de baixa densidade e que necessitam de alta velocidade de leitura, sendo em sua maioria *read-only* (somente leitura). [51] A memória NOR é amplamente utilizada no contexto de sistemas embarcados, no que se refere a *boot* (inicialização), Sistemas Operacionais e código *execute-in-place* (método de execução de código onde os programas são executados diretamente do dispositivo de armazenamento secundário).

Já as memórias NAND, em contrapartida, são voltadas para armazenagem de alta densidade, onde ocorre um *tradeoff* entre a capacidade de acesso aleatório e o tamanho da célula (menor que uma célula NOR, em geral) levando a um *chip* menor e por conseguinte menor custo. [51]

As memórias NAND podem ser divididas em duas categorias, *Single-Level-Cell* (SLC) e *Multi-Level-Cell* (MLC). Uma memória NAND SLC armazena apenas um *bit* enquanto

que uma MLC pode armazenar mais de um. Este último tipo de memória apresenta, em geral, menor tempo de vida e maior latência de acesso que a primeira. [4]

As memórias Flash suportam três tipos de operações básicas, são elas: leitura, escrita e apagamento. Para que ocorra uma operação de escrita é necessária uma operação de apagamento prévio. O processo de escrita e apagamento é conhecio como *erase-write cycle* - ciclo de escrita/apagamento, porém cada vez que ele é executado tem-se o desgaste da memória, o que confere a esse tipo de armazenamento tempo de vida limitado a um número determinado de ciclos. Uma memória NAND MLC típica suporte até 10000 ciclos, enquanto que uma SLC suporta até 100000 ciclos. [4]

2.1.3 Discos de estado sólido

A maioria dos discos de estado sólido utiliza a memória NAND, em uma forma que pode ser chamada de *Managed NAND* (NAND gerenciada). Este termo é utilizado apenas para diferenciar os dispositivos que utilizam a memória NAND como forma primária de armazenamento em uma forma "crua" (*raw* NAND) dos dispositivos que possuem uma interface mediadora das operações de acesso, por meio uma controladora, que implementa mecanismos para, por exemplo, gerenciar o acesso a blocos ou minimzar o desgaste inerente a esse tipo de memória. Mecanismos esses que podem interferir diretamente na prática forense digital. Os SSDs e suas características serão abordados em maiores detalhes no Capítulo 3.

2.2 Abstrações

Uma abstração no contexto da Ciência da Computação e da Informática pode ser entendida como uma técnica na qual um usuário ou outro sistema interage com um sistema sem precisar entender os detalhes de como este funciona. Um exemplo bastante pertinente é o da programação. Um programador escreve um programa em uma linguagem de programação, para que o o computador execute uma tarefa ou resolva um problema. Essa linguagem no entanto, não pode ser entendida pelo computador na forma que está, necessitando ser traduzida por meio de um compilador, (ou interpretada, ou submetida a um processo híbrido), de modo que terá como produto final uma linguagem mais próxima do *hardware*, chamada de linguagem de máquina.

Essa linguagem é diretamente entendida pelo *hardware*, ou em algum momento passa a ser, a depender da arquitetura. Essa quando decodificada, especifica operações nos níveis de circuitos eletrônicos, portas lógicas, cargas elétricas etc. O limite para o nível

mais básico de abstração depende apenas do aprofundamento desejado. No exemplo dado e excluindo-se casos específicos, o programador não precisa conhecer em detalhes as estruturas internas ou o funcionamento do tradutor que utilizou (até um certo ponto), do *hardware*, dos componentes eletrônicos deste, sequer dos princípios físicos ou de eletromagnetismo que os regem.

Também é possível pensar no processo de abstração com exemplos práticos como documentos, fotos, textos, sons em formato digital. Um arquivo de texto sendo mostrado em um monitor de vídeo apresenta diversos níveis de abstração. O conteúdo visual desse arquivo é apresentado para o usuário em uma linguagem natural, contendo, por exemplo, caracteres alfanuméricos. Estes por sua vez, são codificados em algum padrão, como por exemplo, **ASCII** - *American Standard Code for Information Interchange* (Código Padrão Americano para o Intercâmbio de Informação). Esse código é mapeado para 0s e 1s, ou seja, código binário. Para que seja possível abrir esse arquivo com um programa, ele contém informações que o identificam, como cabeçalho, números mágicos, metadados etc. O conjunto de bits que formam o arquivo é armazenado em um dispositivo de armazenamento secundário, que converte os dados em forma binária em níveis de magnetismo ou níveis de cargas elétricas, a depender do substrato da mídia responsável pelo armazenamento. Além desses processos, existem inúmeros outros envolvidos na simples tarefa de visualização desse arquivo, como carregamento na memória primária, representação gráfica no *hardware* de vídeo, interfaceamento com o monitor de vídeo etc.

2.3 Volumes e partições

Um volume, dentro de um contexto computacional pode ser entendido como uma região de armazenamento com um único Sistema de Arquivos. Um volume é uma abstração criada pelo Sistema Operacional para o usuário. Um Sistema de Volume possui duas finalidades básicas. A primeira é a de agrupar diferentes dispositivos físicos sob uma mesma organização lógica, fornecendo um dispositivo único.

Na Figura 2.3 tem-se que a partir de dois discos são formados 3 volumes: C, D e E. A outra finalidade é a de criar partições dentro de um mesmo volume. Também na Figura 2.3 tem-se as partições de 1 a 5. A Figura 2.4 ilustra a atribuição de diferentes Sistemas Operacionais, com diferentes Sistemas de Arquivos, para cada partição.

Embora conceitos similares, há uma diferença fundamental. Os volumes existem no nível de Sistema Operacional, enquanto que as partições existem no nível físico, depen-

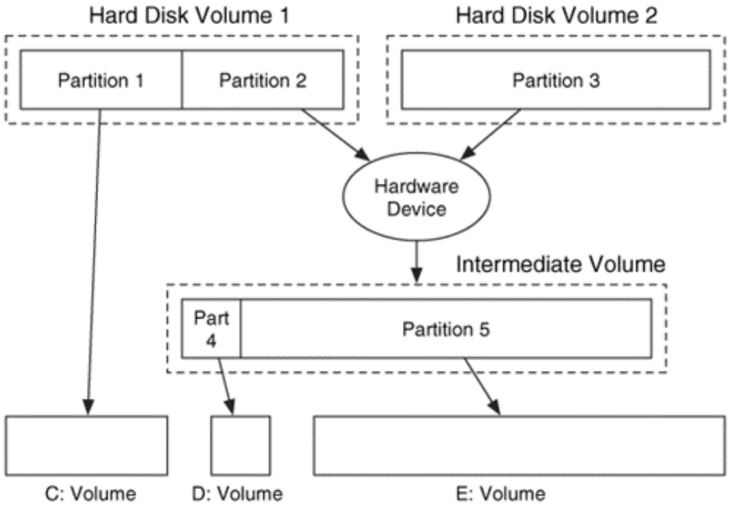


Figura 2.3: Agrupamento de 2 discos para formar diferentes volumes [7]

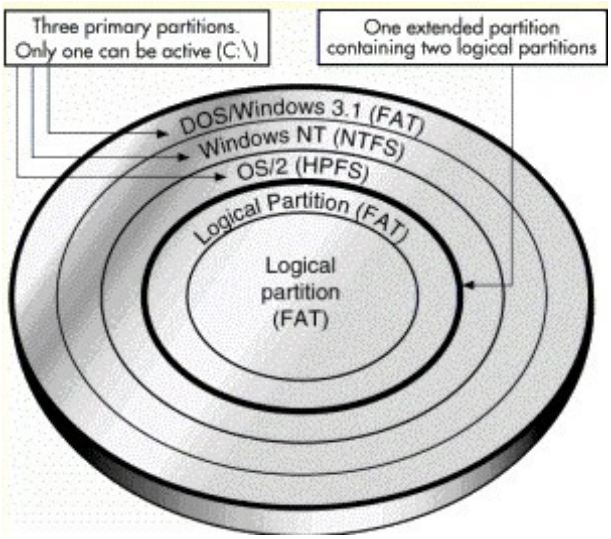


Figura 2.4: Disco particionado em 5 partições [47]

dendente do tipo de mídia, porém o sistema de particionamento usado possui uma certa dependência do SO. Vale ressaltar que também é possível a existência de um volume em um único arquivo, como por exemplo as imagens de disco de CDs e DVDs, chamadas de imagens ISO, nome derivado do sistema de arquivos comum a esse tipo de mídia, o *ISO 9660*. Também existem os discos rígidos virtuais (*Virtual Hard Disks*.)

Em termos técnicos, de acordo com Carrier [7], um volume é uma coleção de setores que um SO ou aplicação que pode utilizar para armazenar dados. Os setores não precisam ser contíguos, é necessário apenas que aparentem ser para viabilizar o armazenamento. Cada SO utilizará um Sistema de Volumes próprio, que irá definir um modo de particionamento específico. Um sistema básico de particionamento precisa apenas definir o primeiro e o

último setor de uma partição e o Sistema de Arquivos utilizado. A Figura 2.5 ilustra essa ideia.

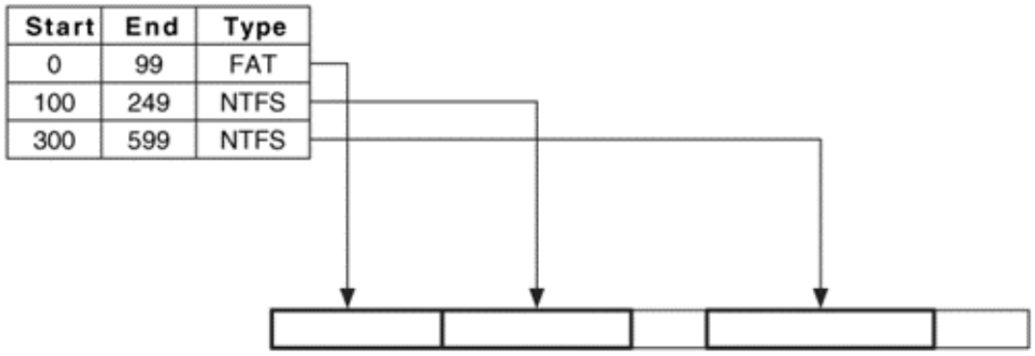
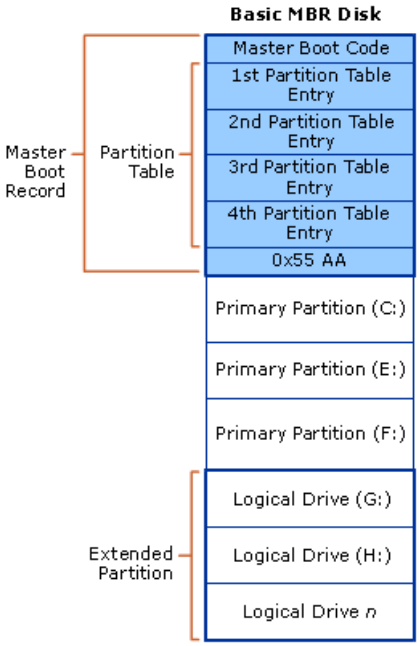
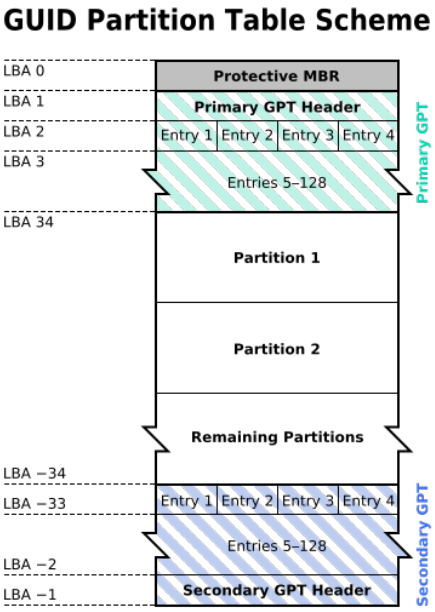


Figura 2.5: Esquema básico de particionamento [7]

Os padrões para particionamento mais utilizados atualmente são o **MBR** - *Master Boot Record* e o **GPT** - *GUID Partition Table*. A Figura 2.6 ilustra cada um dos padrões.



(a) Layout de partições MBR [12]



(b) Layout de partições GPT [30]

Figura 2.6: Layouts MBR e GPT

2.4 Organização

Em termos gerais, tem-se um dispositivo de mídia. Sobre ele são criadas abstrações, tais como volumes e partições. Em uma partição (ou em um volume quando houver uma correspondência 1 para 1) há um Sistema de Arquivos. Cada Sistema de Arquivos define um tipo de organização dos arquivos, ou seja, como serão armazenados na mídia, seus atributos, metadados, entre outras estruturas.

Existem diversos Sistemas de Arquivos para os mais diversos dispositivos de armazenamento (discos rígidos, fita magnética, discos ópticos, memória *Flash* etc.), para diversas aplicações (Banco de Dados, transacionais, rede etc.). Cada um possuindo uma organização própria e certas peculiaridades. Exemplos: **NTFS, ext, ext2, ext3, ext4, FAT12, FAT16, FAT32, HFS, HFS+, ISO 9660**.

Entre os elementos fundamentais de Um Sistema de Arquivos da família **ext**, por exemplo, a partir da segunda versão **ext2**, estão:

- Um **superbloco**: descreve o estado geral do Sistema de Arquivos: sua geometria, tamanho do bloco, número de blocos, lista de blocos livres, número de *inodes*, número mágico, entre outras informações;

- ***Array* de *inodes***: abreviatura para *index nodes* (nós de índice). Cada *inode* descreve um arquivo, contendo informações gerais, como seu dono, tipo de arquivo, permissões de acesso, tempo de acesso etc. Além disso, contém ponteiros para os blocos de dados;

- **Blocos de dados**: armazenam o conteúdo do arquivo.

A Figura 2.7 ilustra as abstrações vistas até agora, para um Sistema de Arquivos da família Unix.

Todo arcabouço computacional e as abstrações vistas até agora confluem para fornecer o substrato necessário para implementar a abstração *arquivo* como uma coleção de dados relacionados entre si e também como uma unidade lógica de armazenamento de alto nível.

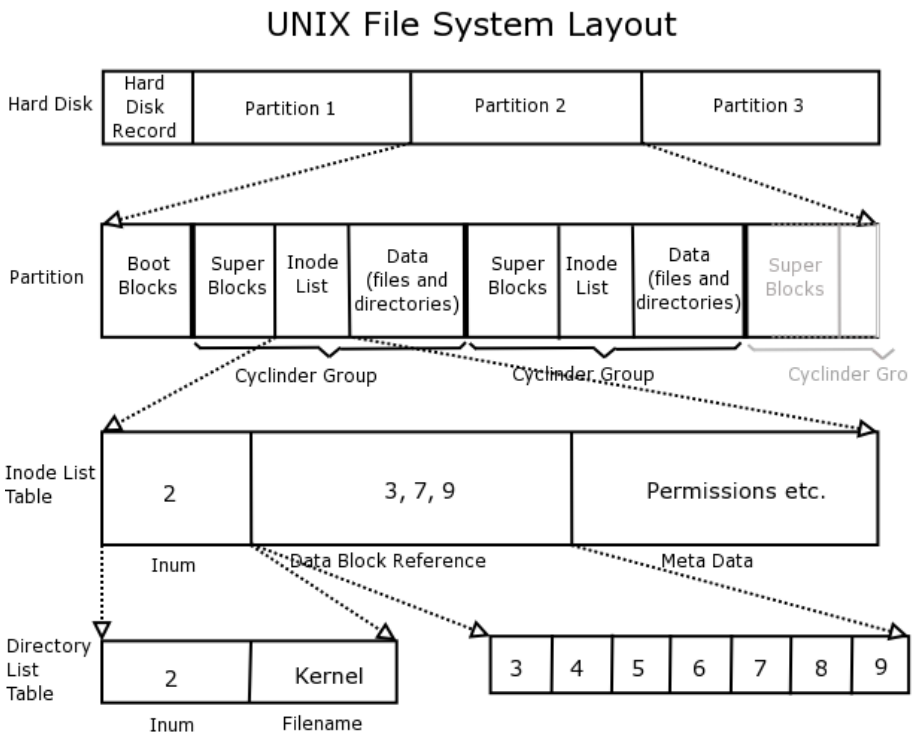


Figura 2.7: *Layout* de um Sistema de Arquivos UNIX [53]

2.5 Exclusão de arquivos

A operação de exclusão de um arquivo é em geral onerosa, se implementada de forma literal, pois envolve a escrita de 0s sobre todo os blocos (a nível de Sistema de Arquivos) ou de setores (a nível físico). Para tal, tem-se o que pode ser chamado de **deleção nominal**, na qual o espaço correspondente ao conteúdo de um arquivo é marcado como livre pelo Sistema de Arquivos, podendo ser sobrescrito pela criação de novos arquivos ou pelo deslocamento de arquivos existentes. Nesse tipo de deleção, pode ser possível a recuperação total ou parcial dos arquivos e de seus conteúdos, a depender da sobrescrita ou não de certos blocos ou setores. Mesmo na sobrescrita total, não há garantias da irrecuperabilidade dos dados. Esse preceito é entendido como *Data remanence* (remanência de dados).

A deleção nominal então se diferencia da **deleção efetiva**. Na última, teoricamente os arquivos não podem ser recuperados após a sua exclusão, pelo menos não utilizando os métodos e conhecimentos do estado da arte de Recuperação Forense.

Neste capítulo foi abordada a estrutura geral dos Sistemas de Arquivos, bem como abstrações relacionadas, como volumes e partições. Também foram abordados conceitos-chave como organização dos Sistemas de Arquivos, exclusão de arquivos e mídias de

armazenamento. No próximo capítulo um tipo de mídia de armazenamento será abordado com maior profundidade: o disco de estado sólido.

Capítulo 3

Discos de Estado Sólido

De acordo com Nisbet et al. [38], as memórias de estado sólido estão disponíveis em diversos formatos há mais de 25 anos. Com a crescente redução do preço, assim como do tamanho, os SSDs se tornam cada vez mais uma alternativa viável à utilização de discos rígidos, tanto nos computadores pessoais, *notebooks*, quanto nos computadores utilizados como servidores.

Nos últimos anos, os SSDs têm se tornado uma tecnologia emergente. São baseados em circuitos integrados, fornecendo diversas vantagens como menor consumo de energia, menor tamanho, resistência a choques e a característica mais impotante de alto desempenho no acesso a dados aleatórios. [4].

A maioria dos SSDs atuais utiliza a a arquitetura NAND planar, ou seja em duas dimensões. Porém já se encontram no mercado dispositivos com a tecnologia V-NAND (*Vertical* NAND) na qual em vez da diminuição do tamanho das células, tem-se o empilhamento de camadas de células na vertical, proporcionando maior densidade e maior capacidade de armazenamento. [14]

3.1 Arquitetura de um SSD

SSDs são normalmente construídos utilizando arranjos de memória *Flash*, que são ligados por um barramento serial a uma **controladora** [4]. A controladora é um dos elementos mais importantes de um SSD, pois é responsável pela execução de rotinas e controle de mecanismos internos. A controladora recebe e processa requisições do Sistema de Arquivos por meio de uma interface de conexão como, por exemplo, a interface **SATA** - *Serial AT Attachment*.

Para facilitar o entendimento da arquitetura de um SSD é possível pensar em três abstrações (camadas). São elas: HIL - *Host Interface Layer*, FTL - *Flash Translation Layer* e FIL - *Flash Interface Layer*. A Figura 3.1 ilustra essa divisão.

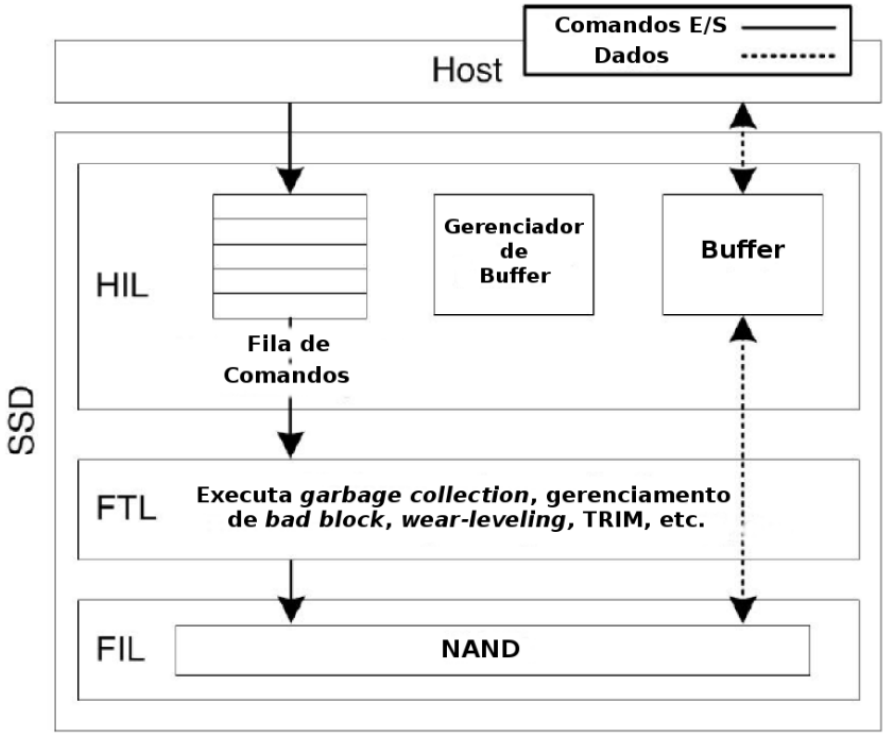


Figura 3.1: Arquitetura SSD [18]

3.1.1 HIL - Host Interface Layer

Nessa camada, comandos de E/S (Entrada/Saída) fornecidos pelo Sistema de Arquivos e/ou SO são enfileirados e atendidos na ordem em que chegam. [35]. Os SSDs costumam utilizar interfaces próprias dos HDDs, como a já citada **SATA**. Também há a presença de outras como: **SAS** - *Serial Attached SCSI*, **ATA/IDE** - *Advanced Technology Attachment/Integrated Drive Electronics*, **PCIe** - *Peripheral Component Interconnect Express* e a popular **USB** - *Universal Serial Bus*.

3.1.2 FTL - Flash Translation Layer

Camada de especial importância pois é responsável pela gerência de certos mecanismos internos utilizados para otimizar o desempenho e por emular o funcionamento de um HDD em termos lógicos, possibilitando a compatibilidade e a interoperabilidade com os sistemas computacionais.

Como sumariza Chen et al. [4], essa camada desempenha três papeis fundamentais: **Mapeamento lógico de blocos, Coleta de lixo e Nívelamento de desgaste**, explicados a seguir:

Mapeamento lógico de blocos

Os discos de estado sólido operam pelo armazenamento de dados em blocos de tipicamente 512 kB, subdivididos tipicamente em páginas de 4 kb, resultando em 128 páginas por bloco nessa configuração [40]. Tais blocos são constituídos de arranjos de transístores *NAND* semelhantes aos chips utilizados em processadores, no entanto conforme explica Bell & Boddington [2], tais arranjos possuem entre outras diferenças, uma fundamental: a presença de uma porta lógica extra que permite a persistência de uma carga elétrica. Esse mecanismo é utilizado para representar os dados de forma permanente.

Devido a essa organização física, é necessário mapear os endereços dos blocos lógicos (LBA) na visão do Sistema de Arquivos/SO para endereços de blocos físicos. Esse mapeamento é diferente do mapeamento LBA dos HDDs porque muda constatemente por conta do nivelamento de desgaste e da coleta de lixo, que serão explicados posteriormente. É possível um mapeamento em nível de páginas ou um mapeamento em nível de blocos [4], sendo que ambos são realizados na camada FTL.

Nivelamento de desgaste

Geralmente as operações de escrita apresentam certa localidade, ou seja, frequentemente as mesmas regiões de armazenamento são escritas. Esse princípio é conhecido como princípio da localidade espacial. Naturalmente, as escritas em um SSD também estão sujeitas a esse preceito, onde um conjunto de blocos (lógicos) é constantemente referenciado para escrita pelo Sistema de Arquivos. Como as memórias *Flash* possuem um número limitado de ciclos *erase-write*, foi concebido o mecanismo de nivelamento de desgaste (*wear leveling*) que uniformiza essas operações pelos blocos físicos do disco. Logo, escritas subsequentes em um bloco lógico na realidade correspondem a escritas em blocos diferentes (a controladora copia o conteúdo de um bloco físico para outro de modo a evitar que o primeiro seja sobrescrito), assumindo que o SSD encontra-se em um estado não desgastado.

Coleta de lixo

Dados são escritos na granularidade de páginas, porém, geralmente, somente blocos inteiros podem ser apagados [2]. À medida que o SSD vai sendo utilizado, "sobram"

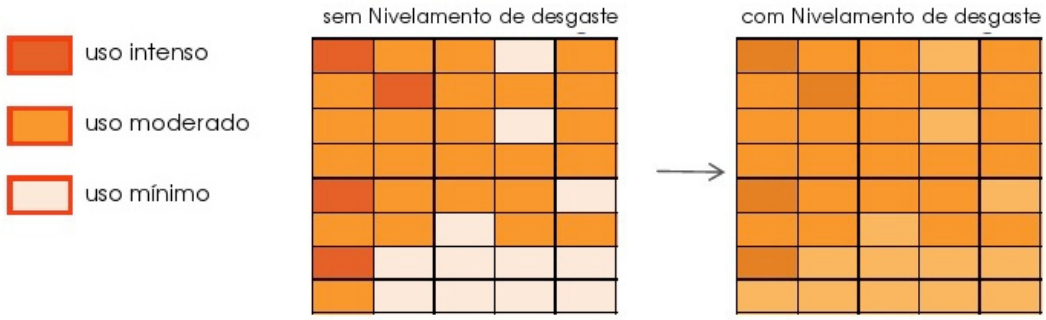


Figura 3.2: Nivelamento de desgaste; Adaptada de [36]

menos blocos sem páginas livres, por conta do nivelamento de desgaste que uniformiza as operações de escrita no SSD. Dessa forma, quando deseja-se reutilizar uma página que contém dados desatualizados (por exemplo, numa atualização de arquivo de texto), é necessário copiar as páginas válidas desse bloco para outro bloco na mídia, para que o primeiro seja apagado. Esse processo é custoso e é um dos grandes limitadores de desempenho no longo prazo.

Para amenizar esse problema, foram concebidas rotinas para a controladora do SSD que identificam blocos contendo páginas não utilizadas ou contendo dados desatualizados, antes de uma operação de escrita. Após a identificação, a controladora realiza o procedimento explicado acima assim que possível. Essa rotina recebe o nome de "coleta de lixo", porém na verdade a rotina faz a "coleta" tanto de dados dados válidos quanto inválidos. No entanto há um problema central, a controladora do SSD "não sabe" o que são dados inválidos (ou lixo), pois essa informação é dependente do Sistema de Arquivos e do Sistema Operacional e cada combinação desses define um conjunto de regras para a deleção de dados e utilização de espaço livre. O SO não comunica à controladora os blocos passíveis de apagamento. Bell & Boddington [2] conjecturam que após a cópia de uma página pelo mecanismo do *wear leveling*, esta está sujeita a coleta de lixo em um momento oportuno. Nesse cenário, seria possível o apagamento automático sem a comunicação do SO.

Existem, portanto, dois tipos principais de coleta de lixo: a *background garbage collection* e a *filesystem-aware garbage collection*. A primeira trata-se do caso mais geral, onde foi enviada uma requisição de escrita a um bloco em uso. Já a segunda se refere a coleta de lixo onde a controladora "ciente" do Sistema de Arquivos em uso, verifica o histórico de arquivos apagados e informações globais sobre o SA. Bell & Boddington [2] exemplificam esse tipo de coleta de lixo com a interação entre o Sistema de Arquivos NTFS e SSDs da marca Samsung.

Para lidar com o cenário mais geral, foi criado o comando TRIM que será explicado na Seção 3.3.

Gerência de Blocos Ruins e ECC

A Gerência de Blocos Ruins (em inglês *Bad Block Management*) trata de rotinas de manutenção no nível físico. Por meio de *bits* de paridade e Código de Correção de Erros (*Error Correcting Code - ECC*) a controladora verifica erros de operação e se possível os corrige. Caso contrário, o bloco onde ocorreu o erro é marcado como "bloco ruim" e um bloco reserva o substitui. [13]

3.1.3 FIL - Flash Interface Layer

Camada de nível mais baixo, é constituída em essência pelas memórias NAND. Dentro dela, são realizadas as operações de leitura, escrita e apagamento. Sendo que na leitura a menor unidade é a página. A escrita também é geralmente realizada em termos de páginas, e por fim o apagamento é feito em nível de blocos. [4]

3.2 Provisionamento

Nos SSDs, o provisionamento é um mecanismo que objetiva aumentar o tempo de vida do dispositivo, atuando junto com o nivelamento de desgaste e com a coleta de lixo. Algumas áreas (blocos) são reservados para a controladora, sendo estas chamadas de área de sobre-provisionamento ou em inglês, *over-provisioning area* (OP). Esses blocos podem ser utilizados quando uma operação de escrita é enviada, requisitando bloco(s) para escrita. Esta é realizada na OP, enquanto que um bloco *dirty* (inválido para escrita, ou apagado) entra na área de provisionamento. Os blocos *dirty* são apagados (coleta de lixo) quando o SSD encontra-se em estado de baixa atividade. [23]

Por conta do provisionamento, os SSDs possuem uma quantidade de memória física maior do que o especificado (ou reconhecido pelo Sistema Operacional), porém a memória efetiva continua a mesma.

3.3 Comando TRIM

O comando TRIM (em português aparar/podar) é um comando no nível de Sistema Operacional que sinaliza á controladora que existem blocos não utilizados (que foram excluídos) para que esta apague previamente os blocos de modo a antecipar posteriores

escritas (como já explicado, requerem um apagamento prévio). A Figura 3.3 ilustra o funcionamento do TRIM.

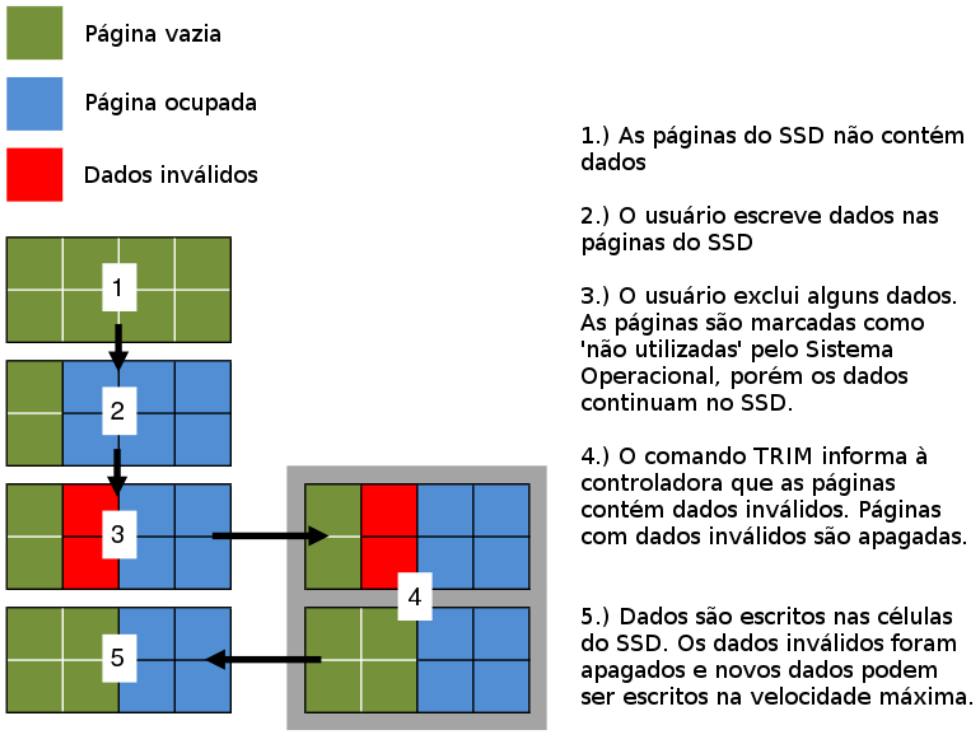


Figura 3.3: Funcionamento do comando TRIM; Adaptada de [26]

3.3.1 Leitura determinística após o TRIM

De acordo com Gubanov e Afonin , cada SSD implementa de forma diferente o comando TRIM. Alguns implementam o que é chamado **DRAT** - *Deterministic Read After Trim* (Leitura Determinística Após o Trim) ou **DZAT** - *Deterministic Zeroes After Trim* - ("Zeros" Determinísticos Após o Trim) [23]. No primeiro, após o envio do comando TRIM pelo SO à controladora seguido de um comando de leitura desses blocos, o SSD retorna blocos contendo valores fixos. Já no segundo, esses valores são 0s. Outra implementação possível do comando retorna o conteúdo dos blocos até que as rotinas internas de coleta de lixo entrem em ação.

A especificação 8 da interface ATA [33] traz as possíveis implementações do TRIM, como ilustra a Tabela 3.1.

Cada fabricante define como a controladora implementa o comando TRIM, pois não há um padrão. Dessa forma, no caso de um certo par SO e Sistema de Arquivos que

Implementação TRIM	Descrição
-	O comando TRIM não é suportado Os blocos lógicos permanecem inalterados
Aleatório	Comportamento indeterminado após a leitura Cada leitura do blocos blógicos pode retornar diferentes valores
DRAT	Leitura determinística após o TRIM Cada leitura dos blocos lógicos retorna os mesmos valores
DZAT	Leitura determinística após o TRIM Cada leitura dos blocos retorna 0s

Tabela 3.1: Diferentes implementações do comando TRIM

suporte o comando TRIM, o processo de exclusão e posterior leitura dos blocos lógicos "aparados" poderá apresentar diferentes comportamentos quando comparado a outro par SO e Sistema de Arquivos.

Em alguns sistemas operacionais, o TRIM é enviado automaticamente. Em outros, é dependente de configuração, podendo ser necessária uma demanda explícita.

3.3.2 Suporte ao TRIM nos Sistemas Operacionais

Entre os sistemas operacionais que oferecem suporte ao comando TRIM encontram-se: Windows 7 e sucessores [10], Mac OS X (a partir da versão 10.6.8) [46] e o Linux [37]. Além desses, o sistema Android para dispositivos móveis passou a incorporar o TRIM a partir da versão 4.3. [32]

A suportabilidade não é plena, pois depende também do Sistema de Arquivos, e além disso, poderá depender de habilitação ou configuração prévia, como já foi dito.

Linux

No geral, em se tratando de sistemas da família *Linux* que suportam o TRIM, (ou seja, com os Sistemas de Arquivos adequados), as principais formas de execução do TRIM são: manual, pelo comando **fstrim**; automática, também pelo comando **fstrim** só que na forma agendada; (*scheduled trim*), e, por último, a opção de montar um volume com a *flag -discard*.

Na distribuição *Debian*, por exemplo, o TRIM é desabilitado por padrão. [42] Na distribuição Ubuntu, nas versões 14.04 e posteriores, o TRIM automático é habilitado por

padrão na forma agendada, porém somente para certos SSDs de algumas marcas. [52]. A opção de montar um volume com a *flag discard* é desencorajada, principalmente por questões de desempenho.

Os desenvolvedores de algumas distribuições *Linux* mantém uma lista negra de dispositivos/*drivers*/*firmwares* no que se refere ao TRIM. No código fonte de uma biblioteca de *drivers* do *kernel* do *Linux* é possível encontrar uma *struct* (estrutura de dados) que enumera certos dispositivos que alegadamente não tratam adequadamente comandos TRIM ou comandos TRIM enfileirados, como ilustra a Figura 3.4:

```
/* devices that don't properly handle queued TRIM commands */
{ "Micron_M500_*",              NULL,   ATA_HORKAGE_NO_NCQ_TRIM |
                                        ATA_HORKAGE_ZERO_AFTER_TRIM, },
{ "Crucial_CT*M500*",           NULL,   ATA_HORKAGE_NO_NCQ_TRIM |
                                        ATA_HORKAGE_ZERO_AFTER_TRIM, },
{ "Micron_M5[15]0_*",           "MU01", ATA_HORKAGE_NO_NCQ_TRIM |
                                        ATA_HORKAGE_ZERO_AFTER_TRIM, },
{ "Crucial_CT*M550*",           "MU01", ATA_HORKAGE_NO_NCQ_TRIM |
                                        ATA_HORKAGE_ZERO_AFTER_TRIM, },
{ "Crucial_CT*MX100*",          "MU01", ATA_HORKAGE_NO_NCQ_TRIM |
                                        ATA_HORKAGE_ZERO_AFTER_TRIM, },
{ "Samsung SSD 8*",             NULL,   ATA_HORKAGE_NO_NCQ_TRIM |
                                        ATA_HORKAGE_ZERO_AFTER_TRIM, },
{ "FCCT*M500*",                 NULL,   ATA_HORKAGE_NO_NCQ_TRIM |
                                        ATA_HORKAGE_ZERO_AFTER_TRIM, },

/* devices that don't properly handle TRIM commands */
{ "SuperSSpeed S238*",          NULL,   ATA_HORKAGE_NOTRIM, },
```

Figura 3.4: *blacklist* de SSDs na biblioteca ATA do *Linux* [50]

Além de problemas com a implementação do TRIM, outros motivos desencorajam a adoção automática do comando, seja na forma agendada se realizada excessivamente, ou na forma da montagem de volume com a *flag discard*. É possível citar, entre outros:

- *Bugs* nos *firmwares* dos SSDs, que podem levar a corrupção dos dados [42];
- *Bugs* no provisionamento; [23]
- Desempenho: implementações ineficientes das rotinas de coleta de lixo e nivelamento de desgaste;
- Criptografia: blocos apagados pelo TRIM se diferenciam dos blocos cifrados, logo tal informação pode ser utilizada para atacar a criptografia do SSD.

Windows

Os sistemas da família *Windows* possuem suporte ao TRIM, a partir da versão 7, incluindo *Windows 8* e *Windows 8.1*, no que se refere a SOs para *desktops* e *notebooks*. Já na area dos servidores, a partir da versão *Windows Server 2008 R2*, ou seja, também nas versões *Windows Server 2012* e *Windows Server 2012 R2* tem-se o reconhecimento dos SSDs e o suporte ao TRIM. [11]

OS X

Até a versão 10.10.4 do OS X, alguns computadores da Apple tinham suporte ao TRIM, porém apenas para os SSDs da própria marca. Com essa versão, foi adicionado um utilitário via linha de comando, o *trimforce*, para habilitar o TRIM em SSDs de outras marcas. [24]

Capítulo 4

Recuperação Forense e Forense Digital

4.1 Investigação Digital e Forense Digital

Carrier [7] define uma investigação digital como sendo o processo onde tem-se o levantamento e o teste de hipóteses para responder questões sobre eventos digitais. O objeto de investigação é algum dispositivo digital envolvido ou relacionado a um incidente ou a um crime.

Já a Forense Digital refere-se a investigações conduzidas utilizando metódos científicos com o propósito de compor provas perante um tribunal ou corte. A principal diferença entre a Investigação Digital e a Forense Digital reside no atendimento a certos requisitos legais.

4.2 Procedimentos

Carrier [7] divide o processo de investigação da cena do crime em três etapas, são elas: preservação da cena do crime, busca de evidências e a reconstrução de eventos. Além disso, conceitua dois tipos de análises: *dead analysis* e *live analysis*:

Live analysis: É uma análise em que são utilizados recursos próprios do ambiente computacional da cena do crime para encontrar evidências. Nesse tipo de análise as evidências estão mais sujeitas a adulterações e interferências do próprio processo.

Dead analysis: Nesse tipo de análise os objetos digitais são analisados em um ambiente controlado voltado para a prática investigativa. É preferível à *dead analysis* na maioria dos casos, porém nem sempre é possível.

Nesse trabalho o enfoque será nas investigações referentes a Sistemas de Arquivos e *Live analysis*, mas casos práticos podem envolver outras variáveis como por exemplo *logs* de eventos e rede.

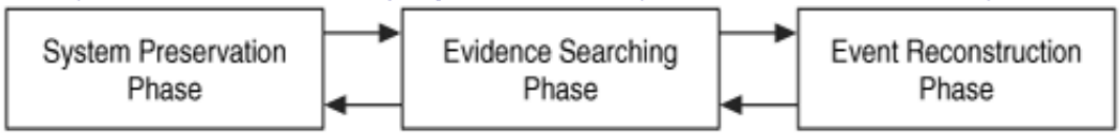


Figura 4.1: Processo de investigação da cena do crime [7]

4.2.1 Preservação da cena do crime

Tipicamente fazem parte da cena do crime em um evento ou incidente digital:

- Os dispositivos físicos, como computadores pessoais, *notebooks*, celulares, *tablets*, pen drives etc. Correspondem ao *hardware*.
- Os dados, programas em execução em um dado momento, estado da memória e do SO etc. Correspondem ao *software*.

Na fase de preservação da cena do crime, é necessário preservar o estado do sistema computacional a ser objeto de análise. Os passos a serem tomados dependerão do tipo de sistema e das circunstâncias em que ele se encontra. Pode ser necessário manter o dispositivo ligado para obter informações de execução de programas armazenadas em memória volátil (*live analysis*), ou pode ser necessário desligar todo o dispositivo imediatamente (*dead analysis*). Casos específicos podem demandar uma forma híbrida de análise.

Técnicas de preservação

É necessário durante um processo investigativo empregar técnicas que minimizem ou impossibilitem a alteração do estado do sistema computacional encontrado e consequentemente das evidências em potencial. Em se tratando de dispositivos de armazenamento, é comum o uso de *write-blockers*, dispositivos que como o nome sugere, bloqueiam operações de escrita enquanto permitem operações de leitura, ou seja, permitem a aquisição de dados sem alterar o estado do dispositivo. Também existem *write-blockers* na forma de programas (*software write-blockers*).

Em muitos casos é necessário manter uma cópia das informações digitais contidas no dispositivo de armazenamento, para evitar alteração e também para realizar análises posteriores. Um método muito empregado é o de *disk imaging* (imagem de disco). Nesse metódo, tipicamente é criada uma imagem (arquivo) contendo uma cópia fiel setor-a-setor do conteúdo da mídia. Sobre essas imagens são realizadas as etapas posteriores, com o intuito de preservar o estado original do dispositivo.

Durante o processo de *disk imaging*, é necessário garantir a integridade dos dados obtidos a partir do disco. Para tal, é utilizada uma função de *hash* criptográfico. Uma função *hash* recebe uma sequência de dados de tamanho variável fornecendo como saída uma sequência de dados de tamanho fixo, chamada de *digest* ou resumo. O *digest* pode ser entendido como uma "impressão digital" daquela sequência de dados.

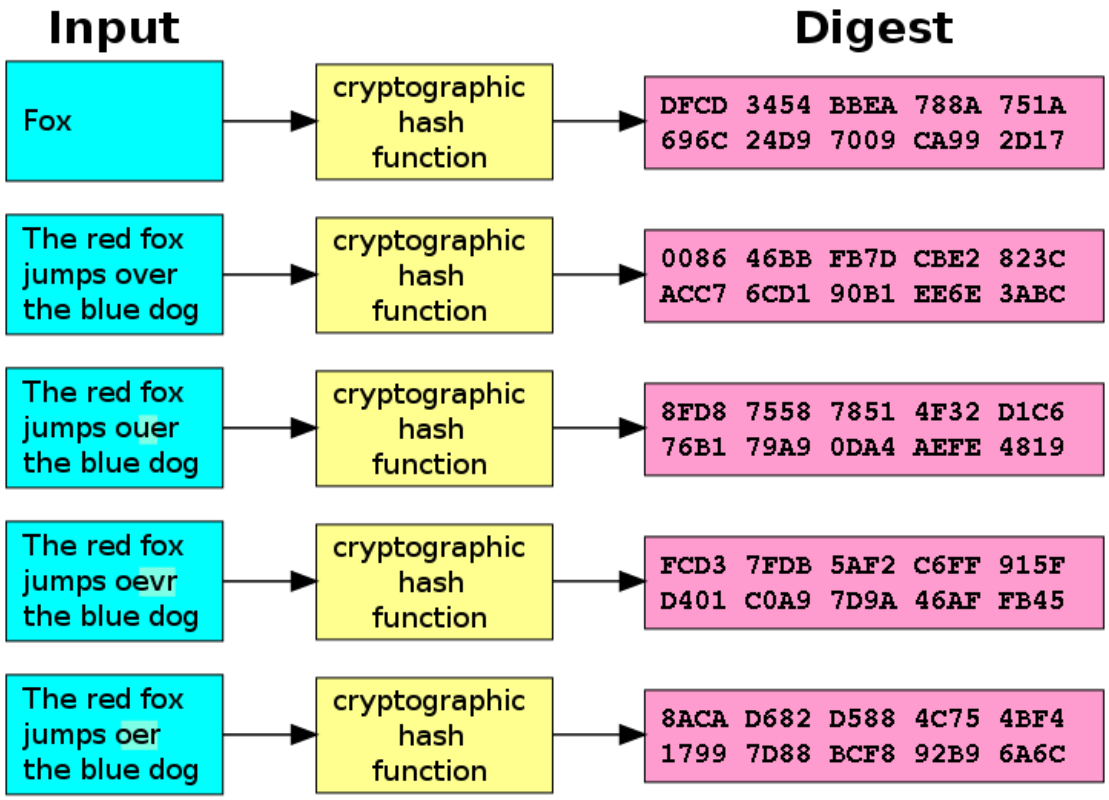


Figura 4.2: Função de *hash* criptográfico [49]

A robustez da prova digital está diretamente relacionada com o emprego de uma função de hash adequada. Para ilustrar a importância do processo de *hashing*, é possível imaginar um cenário em que um disco rígido é obtido durante uma operação de busca e apreensão. É criada uma imagem do disco, associada a um valor de *hash*. O dispositivo é destinado a um local especial até que seja requisitado novamente. Sobre a imagem criada são encontradas

evidências de considerável relevância para o caso. Alguma parte como um advogado, por exemplo, pode requisitar a verificação do *hash* da imagem com o do disco, original. Se há diferença, os dados foram alterados. Caso os valores sejam idênticos, não há garantia de integridade, porém a violação desta é extremamente improvável, a depender da robustez do algoritmo e da função de *hash* utilizados.

A confiança no uso desse sistema reside no atendimento a certos requisitos. É imprescindível que a função de *hash* possua as seguintes propriedades:

- Que seja impraticável, dado um valor de *hash*, obter a mensagem original (ou a sequência de dados original), ou seja, que a função não seja inversível;
- Que seja impraticável alterar uma sequência de dados sem que essa alteração modifique o valor do *hash*;
- Que seja impraticável construir duas sequências de dados diferentes com valores de *hash* iguais (colisão). Logo, a função *hash* ideal é injetora.

Além dessas propriedades, para fins práticos, é desejável que o tempo do cálculo do *hash* de uma sequência de dados seja mínimo.

São utilizados tipicamente os algoritmos da família **SHA** - *Secure Hash Algorithm* como o **SHA-1, SHA-256**, assim como o algoritmo **MD5** - *Message Digest Algorithm 5*, embora esse último possua comprovadas vulnerabilidades e seja possível ataques de colisão em tempo 2^{18}. [54]

4.2.2 Busca de evidências

Após a etapa de preservação dos dados, se dará início a procura pelas evidências que comprovem ou refutem hipóteses levantadas sobre o incidente. Cada incidente irá definir um conjunto de procedimentos iniciais que guiarão a procura por dados digitais. Por exemplo, no caso de invasão de um comptuador, poderão ser verificados os *logs* de autenticação, a presença de programas maliciosos como *keyloggers*, *rootkits*, *worms* etc. Outro exemplo é o de suspeita de distribuição de material pornográfico envolvendo menores, em que são analisados o Sistema de Arquivos a procura de imagens.

No último exemplo citado, em muitos outros, é bastante comum em uma investigação digital a procura por nomes específicos de arquivos, e por formatos específicos de imagens: `.jpeg, .jpg, .png, .bmp, .gif` etc. Também é comum buscar arquivos pelo seu

conteúdo, por certas informações como nomes, datas, palavras-chave, ou por metadados como tempo de acesso, modificação, exclusão etc.

4.2.3 Reconstrução de eventos

A última etapa consiste, quando couber, fazer a ligação entre as evidências encontradas, as circunstâncias e demais informações pertinentes para determinar que eventos ocorreram no sistema em análise e de que forma se deram. Essa etapa geralmente é procedida de um relatório ou laudo.

4.3 Exclusão de arquivos ou dados

Na Forense Digital, a fase de busca de evidências envolve a análise de dados, que subdivide-se em diferentes análises em cada nível de abstração. No que se refere a dispositivos de armazenamento, o foco é geralmente na análise do Sistema de Arquivos.

No caso mais comum, é possível obter os dados (arquivos) de forma plena, empregando buscas ou varreduras no disco, podendo ser necessário utilizar técnicas mais sofisticadas para encontrar arquivos ocultos ou que se encontrem sob a ação de mecanismos de segurança. Em certas situações, embora seja possível a obtenção dos arquivos, eles se encontram com seu conteúdo e/ou metadados cifrados. De toda forma, nos cenários mencionados, os arquivos encontram-se plenos e integram a estrutura lógica do Sistema de Arquivos.

Em muitos casos, no entanto, seja pela falta de evidências ou por insuficiênca dessas, é necessário realizar o que pode ser chamada de Recuperação Forense. Nesse processo, complementar a Forense Digital (ou parte dela), também se procura por arquivos excluídos, geralmente de forma "permanente" . O termo "permanente" é utilizado para se referir a exclusões emitidas pelo SO, que podem ou não configurar uma exclusão efetiva, as chamadas **exclusões nominais**, conforme já explicado. Geralmente excetuam-se como objetos de análise da Recuperação Forense os arquivos enviados para "Lixeira" ou equivalente, pois esses foram apenas movidos de lugar, ou aparentam terem sido.

Grande parte dos Sistemas de Arquivos, quando avaliados no escopo de sua abstração, implementam a deleção nominal. Logo, caso não haja sobrescrita dos blocos/setores, é possível recuperar parcialmente arquivos excluídos, e nos melhores casos, arquivos completos. Esse fato é característico da maioria dos dispositivos de armazenamento magnético, porém, em se tratando de discos de estado sólido, o cenário não é tão otimista.

4.4 Recuperação forense

É possível definir duas abordagens diferentes para a recuperação de arquivos ou dados excluídos em uma mídia de armazenamento, de forma extremamente simplista. A primeira é uma análise estrutural de volumes, de Sistemas de Arquivos (metadados, tipos/extensões de arquivos, nomes dos arquivos, *journaling*), ou do meio físico. A segunda trata-se da análise do conteúdo dos arquivos (*data carving*).

A principal diferença entre as duas abordagens é que na primeira é utilizado o arcabouço computacional responsável pelas abstrações envolvendo arquivos como unidades lógicas. Por exemplo: um arquivo de imagem no formato `.jpeg` é excluído num sistema *Linux* com o **ext4**, sendo enviado para a lixeira. Nesta, é excluído novamente. Para recuperá-lo, no Sistema de Arquivos específico, é possível analisar o *journal* (diário que contém as mudanças ainda não efetivadas no Sistema de Arquivos) para reverter a operação de exclusão. Se o procedimento for realizado em um sistema *Windows* com o **NTFS** por exemplo, é possível reconstruir a entrada na MFT - *Faster File Table* se os blocos não foram sobrescritos.

Já na segunda abordagem, em vez de se utilizar estruturas ou metadados, é feita uma busca pela assinatura (ou *magic numbers*) do formato `.jpeg`, ou seja pelos valores hexadecimais `FF D8 FF DB`. Se encontrados, tenta-se obter o arquivo original a partir do cabeçalho. Essa técnica é conhecida como *file carving* ou *data carving*, que não se limita a análise de assinaturas e cabeçalhos. Tal técnica no entanto é altamente suscetível a falsos positivos.

Na prática, por conta de características dos dispositivos de armazenamento, uso contínuo (alta fragmentação), nuances dos Sistemas de Arquivos, entre outros fatores, pode ser necessário analisar tanto os blocos não alocados quanto os alocados, pois pode haver fragmentação interna, blocos parcialmente excluídos etc. Dessa forma, em certas cituações pode ser necessáro utilizar as duas abordagens de forma complementar.

4.4.1 Ferramentas

Existem diversas ferramentas (*software*) para a forense computacional em geral, assim como para prática forense voltada para recuperação de arquivos excluídos. Desde simples utilitários via linha de comando como o *scalpel*, para *Linux*, de *software* livre, ferramentas proprietárias como o *EnCase*, para *Windows*, bibliotecas de programas como o *The Sleuth*

Kit, para *Windows* e *Unix-like,* a Sistemas Operacionais com módulos dedicados à forense digital, como o *Kali Linux*.

Embora existam ferramentas mais genéricas, algumas se limitam a um Sistema de Arquivos específico, como por exemplo o *extundelete*, utilizado nos sistemas *Linux* para os Sistemas de Arquivos da família **ext**. O *magicrescue*, por exemplo, voltado para *file carving*, funciona na base de arquivos *recipes* (receitas) que descrevem como recuperar um tipo específico de arquivo. Cada ferramenta explora características estruturais e arquiteturais de um sistema computacional ou de uma camada de abstração para realizar a recuperação forense, quando possível. A Tabela 4.1 contém algumas das principais ferramentas de recuperação forense.

Ferramenta	Sistema Operacional	Descrição	Autor(es)
extundelete	*Linux*	Utilitário de recuperação de arquivos excluídos em uma partição *ext3* ou *ext4*. Utiliza informações armazenadas no *journal* das partições para realizar a recuperação. [8]	N E Case
foremost	*Linux*	Utilitário via linha de comando para recuperar arquivos com base em seus cabeçalhos, rodapés e estruturas internas. [27]	Jesse Kornblum, Kris Kendall e Nick Mikus
magirescue	*Linux*	*File carver* que faz a varredura em um dispositivo de blocos e por meio de um programa externo realiza a extração dos arquivos. [28]	Jonas Jensen
scalpel	*Linux Windows OS X*	Utilitário que realiza *file* carving e indexação. [25]	G. G. Richard III
The Sleuth Kit ®	*Linux Windows*	Biblioteca e coleção de ferramentas de linha de comando que permitem a investigação de imagens de disco. [6]	Brian Carrier
autopsy	*Linux Windows*	Plataforma de forense digital e interface gráfica para o *The Sleuth Kit* ® [5]	Brian Carrier
FTK Imager	*Windows*	Ferramenta de pré-visualização e *imaging* utilizada para a aquisição de dados. [16]	ACCESS DATA

Tabela 4.1: Ferramentas de recuperação forense

4.5 O desafio proposto pelos SSDs

O crescente uso dos SSDs, por conta de seus mecanismos e características explicados brevemente no Capítulo 3, pode representar um desafio para a prática atual de Recuperação Forense ou Forense Digital. Em síntese, os possíveis obstáculos ou impedimentos são:

- A execução das rotinas de coleta de lixo e nívelamento de desgaste realizadas pela controladora de acordo de forma autônoma;
- O provisionamento de blocos realizado pela controladora;
- O fato das rotinas acima estarem encapsuladas na *Flash Translation Layer*, não sendo visíveis ou manipuláveis de forma direta;
- O comando TRIM, tanto nas formas automáticas (montagem de volume com *flag discard* ou agendado) quanto na forma manual.

É possível afirmar que, de modo geral, a prática da anti-forense em dispositivos de armazenamento (memória secundária) foi altamente favorecida em detrimento da prática forense com a introdução dos SSDs, tanto passivamente quanto ativamente. Passivamente com a execução das rotinas do SSD sem interferência ou com interferência mínima. Blocos apagados possuem efemeridade quanto ao seu conteúdo, a depender da conveniência e da oportunidade dos algoritmos implementados na controladora. Ativamente com a execução manual do TRIM.

Diante desse contexto, é possível imaginar alguns casos práticos. Um criminoso precavido pode programar a sua máquina de modo a fazer backups automáticos utilizando criptografia e *scripts* para execução do TRIM periodicamente, limpando possíveis rastros deixados por arquivos de log, por exemplo. Um indivíduo que tendo conhecimento da chegada da polícia, pode acionar um "*kill switch*" excluindo dados sensíveis e executando o TRIM. Em ambos os cenários, assume-se o pior caso (que os dados não podem ser recuperados como nos HDDs, pelo menos não com as práticas atuais.)

4.5.1 Metodologia *black-box versus* metodologia *white-box*

Uma metodologia de análise *black-box* (caixa preta) é uma na qual aspectos internos de um dispositivo ou *software* são ignorados, atentando-se para funcionalidades ou entradas e saídas. Por exemplo, o teste de uma operação de multiplicação em uma calculadora eletrônica é um teste de caixa preta.

Já uma metodologia de análise de *white-box* (caixa branca) pode ser definida como aquela em que um objeto ou *software* é averiguado segundo seu funcionamento interno e suas estruturas internas.

Segundo Bonetti et. al [21], a metodologia *white-box* de análise forense voltada aos SSDs é possível porém é muito dificultosa, demorada e cara. Isso se deve ao fato de que

para realizar uma análise forense desse tipo necessita-se de *hardware* específico para o acesso da interface *Flash* diretamente, pois, com as técnicas atuais, não é possível apenas via *software* por conta da arquitetura de um SSD.

4.5.2 Hardware específico para Forense em SSD

Gubanov e Afonin [23] mencionam em seu artigo a falta de *hardware* específico para a prática forense nos SSDs. Embora os mecanismos de *imaging* sejam similares ou idênticos aos utilizados nos HDDs, os dispositivos *write-blockers* são incapazes de impedir a execução das rotinas do SSD, e até mesmo de impedir a execução de um comando TRIM prévio. Isso se deve em parte por conta da maioria dos *write-blockers* funcionarem na camada HIL - *Host Interface Layer*. Além disso, pela falta de padrões e pela obscuridade das implementações do *hardware* interno, demanda-se muitas vezes soluções específicas.

4.5.3 SSD *Bait-and-switch*

Gubanov e Afonin [23] descrevem em um fenômeno que já ocorreu no mercado de SSDs, o *bait-switching*, no qual os fabricantes, após lançarem um novo disco no mercado e obterem boas avaliações (levando a uma grande demanda), alteram as especificações de *hardware* e/ou de *software* (*firmware*) do SSD sem alterar o seu modelo. Uma alteração possível é a da controladora. Os autores citam dois casos: as marcas Kingston e PNY.

Embora não sendo necessariamente um desafio para a prática forense, pode representar um, pois tais alterações podem ter grandes impactos em uma análise forense guiada por pressupostos errôneos. No entanto, os autores citam um caso em que essa alteração foi benéfica (para a Forense Digital): um SSD da marca PNY teve sua controladora alterada para uma em que por conta de uma implementação ineficiente da coleta de lixo, os dados ficaram mais tempo disponíveis para recuperação.

Capítulo 5

Experimentação

5.1 Experimentos propostos

A partir do levantamento do estado da arte da recuperação forense envolvendo discos de estado sólido realizado no Capítulo 1, foi proposta a realização de cinco experimentos, com base em cinco cenários, com o intuito de comprovar ou contradizer os resultados atuais e por fim realizar uma análise comparativa. Os experimentos foram divididos em dois grupos: sistema *Linux* com Sistema de Arquivos **ext4** e sistema *Windows* com o Sistema de Arquivos **NTFS**.

Uma diferença entre os experimentos propostos e os trabalhos relacionados é que os testes serão guiados por arquivos. Dessa forma, consequentemente, é testada a eficácia da coleta de lixo, do TRIM, e o simples uso de um SSD como prática de anti-forense.

5.1.1 *Linux* (ext4)

Objetivos

Nos experimentos no sistema *Linux* com o Sistema de Arquivos **ext4**, deseja-se verificar a influência da coleta de lixo e do comando TRIM na recuperação de arquivos excluídos, utilizando-se para a recuperação desses algumas das ferramentas apresentadas no Capítulo 4.

Cenários

Foram levantados dois cenários no Sistema Operacional referido:

1. Exclusão de arquivos, sem a atuação do comando TRIM em qualquer forma e posterior tentativa de recuperação, com o objetivo de verificar a atuação da coleta de lixo;

2. Exclusão de arquivos, seguida do envio do comando TRIM manual após um tempo t e posterior tentativa de recuperação com o objetivo de verificar o efeito do comando TRIM e atuação da coleta de lixo.

Descrição

Foram formulados casos de teste para os referidos cenários. Primeiro foi criada e montada uma imagem de partição formatada no SSD. Após isso, foi feita a cópia de arquivos de teste selecionados para a partição seguida da exclusão desses arquivos. Após um certo período de tempo, foi enviado o comando TRIM no caso do Cenário 2. A imagem foi desmontada e foram aplicadas sob esta as ferramentas de recuperação mencionadas no Capítulo 4. Os procedimentos mencionados serão detalhados nas seções dedicadas a cada experimento.

5.1.2 Windows (NTFS)

Objetivos

Nos experimentos no sistema *Windows* com o Sistema de Arquivos **NTFS**, deseja-se verificar a influência da coleta de lixo e do comando TRIM na recuperação de arquivos excluídos, utilizando-se para a recuperação desses as ferramentas apresentadas no Capítulo 4.

Cenários

Foram levantados três cenários no Sistema Operacional referido:

3. Exclusão de arquivos, sem a atuação do comando TRIM em qualquer forma e posterior tentativa de recuperação, com o objetivo de verificar a atuação da coleta de lixo;

4. Exclusão de arquivos, com o comando TRIM automático habilitado e posterior tentativa de recuperação, com o objetivo de verificar o efeito do comando TRIM quando enviado de forma autônoma pelo SO, além de verificar a atuação da coleta de lixo;

5. Exclusão de arquivos, seguida do envio do comando TRIM manual após um tempo t e posterior tentativa de recuperação com o objetivo de verificar o efeito do comando

TRIM e atuação da coleta de lixo. O TRIM automático está desabilitado durante esse experimento.

Descrição

Foram formulados casos de teste para os referidos cenários. Primeiro foi criada uma partição formatada no SSD. Após isso, foi feita a cópia de arquivos de teste selecionados para a partição seguida da exclusão desses arquivos. Após um certo período de tempo, foi enviado o comando TRIM no caso do Cenário 5. Foi criada uma imagem da partição e sob esta foram aplicadas sob esta as ferramentas de recuperação mencionadas no Capítulo 4. Os procedimentos mencionados serão detalhados nas seções dedicadas a cada experimento.

5.2 Ambiente

O SSD alvo de testes é o *850 EVO 2.5'* de 256GB da marca Samsung:

(a) Visão frontal (b) Visão traseira

Figura 5.1: Samsung 850 EVO 2.5' 250GB [44]

Verifica-se que o SSD 850 EVO de acordo com o fabricante, possui suporte ao TRIM e possui um algoritmo de coleta de lixo automática, como ilustra a Figura 5.2 retirada do *site* da Samsung.

O SSD em questão encontrava-se em estado seminovo, onde foi sujeito ao uso de um usuário comum, utilizando certas aplicações como navegador, *softwares* de edição de texto, terminal, visualizadores de mídia (aúdio, texto) etc. configurando-se, em maioria, uso de baixo impacto para o sistema de arquivos.

Special Feature	TRIM Support	TRIM Supported
	S.M.A.R.T Support	S.M.A.R.T Supported
	GC (Garbage Collection)	Auto Garbage Collection Algorithm
	Encryption Support	AES 256 bit Encryption (Class 0), TCG/Opal, IEEE1667 (Encrypted drive)
	WWN Support	World Wide Name supported
	Device Sleep Mode Support	Yes

Figura 5.2: Recursos especiais do SSD Samsung 850 EVO [14]

Os experimentos foram conduzidos em um computador *desktop* com os seguintes componentes de *hardware*:

- CPU Intel(R) Core(TM) i5-4670K CPU @ 3.40GHz
- Placa mãe Asus MAXIMUS VI HERO
- Placa de vídeo nVIDIA GeForce GTX 760
- Memória: 12 GB RAM DDR3 (4+8)
- 1 HDD WDC WD5000AAKX-00U6AA0 3.5' 500GB 7200RPM
- 1 SSD Samsung 850 EVO 2.5' 256 GB
- 1 Leitor/gravador de CD/DVD + periféricos

Nos Experimentos 1 e 2 , o Sistema Operacional é da família *Linux*, distribuição *debian* versão *3.16.0-4-amd64* de 64 *bits*. O sistema de arquivos é o **ext4**. Nos Experimentos 3,4 e 5, o Sistema Operacional foi o *Windows 7 Ultimate*, de 64 *bits*. com Sistema de Arquivos **NTFS**. Ambos os Sistemas Operacionais foram instalados no SSD, em partições diferentes. Os volumes referentes aos testes são montados (seus sistemas de arquivos são anexados ao SO) de forma dinâmica. Nas seções 5.3 a 5.7 serão abordadas as especificidades de cada experimento.

Quanto ao estado da máquina durante a realização dos testes, o computador realizava tarefas de baixa demanda de *hardware*, como navegador, editor de texto, visualizador de imagens etc. Por limitações técnicas, não foi possível isolar totalmente a atuação do SSD. Tentou-se minimizar o uso do disco fora dos casos de teste, porém essa é uma tarefa dificultosa dado que o próprio SO constatemente executa operações de leitura e escrita para a sua execução, mesmo em estado *idle* (parado).

Quanto ao método de análise forense, trata-se de *live analysis*, ou seja, são utilizados recursos do próprio ambiente computacional para se encontrar evidências.

Dessa forma, pode-se pensar nos testes como realistas (ou pessimistas) do ponto de vista de situações concretas onde não se dispõe de ambiente hermeticamente controlado para a prática forense.

5.3 Cenário 1

5.3.1 Descrição

Exclusão de arquivos, sem a atuação do comando TRIM em qualquer forma e posterior tentativa de recuperação, com o objetivo de verificar a atuação da coleta de lixo.

5.3.2 Procedimentos

Para a realização dos experimentos correspondentes aos Cenários 1 e 2, no *Linux*, foram seguidos os procedimentos a seguir.

Criação das imagens de partição

O método experimental abaixo é baseado no trabalho de Cunha sobre técnicas de remoção segura de dados em sistemas de arquivos *ext3*. [15]

Foi criado um arquivo contendo apenas zeros com o utilitário **dd** do Linux conforme a Figura 5.3.

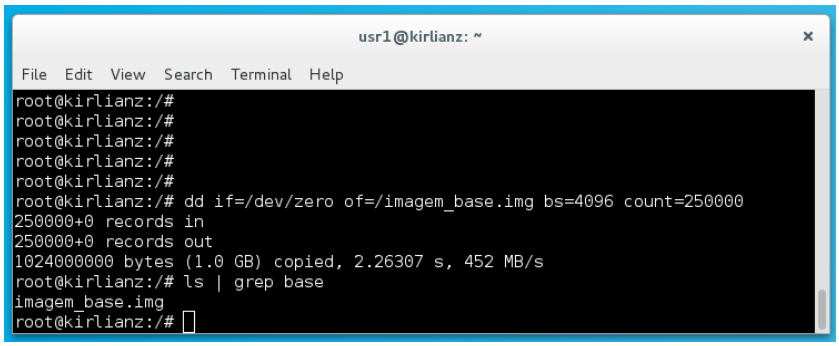


Figura 5.3: Criação de um arquivo de aproximadamente 1GB com o utilitário **dd**

Foi então criada uma partição com o utilitário **fdisk** conforme a Figura 5.4.

A partição foi montada no dispositivo virtual **loop0**, com o comando **losetup** e formatada com o sistema de arquivos *ext*4, por meio do utilitário **mkfs** e então foi desmontada conforme a Figura 5.5.

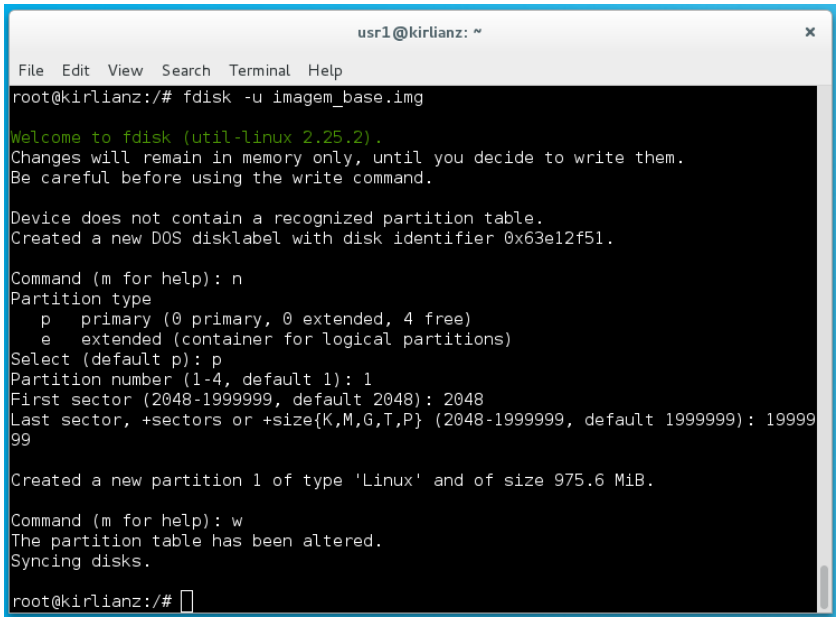


Figura 5.4: Particionamento da imagem de disco com o utilitário **fdisk**

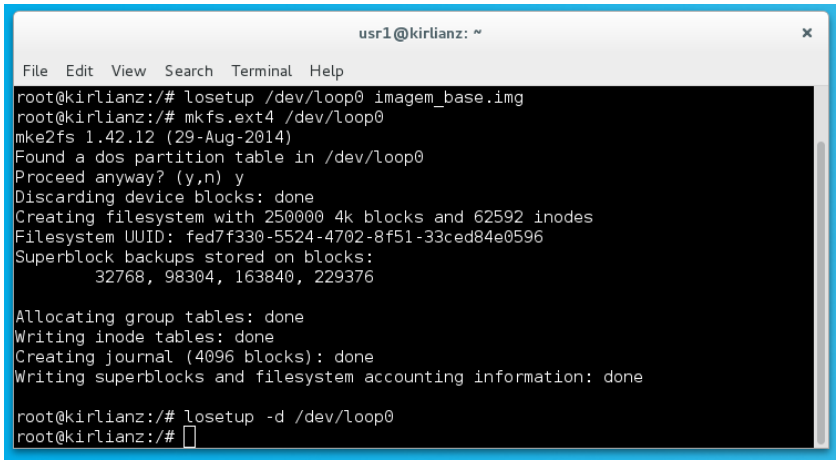


Figura 5.5: Formatação da partição com o sistema de arquivos *ext*4

Arquivos

Foram selecionados 8 arquivos de diferentes formatos conforme a Tabela 5.1.

O arquivo `(Aya Higuchi) Chopin - Waltz in A minor, B. 150.mp3` é um arquivo de música no formato MPEG-1 Layer 3 com taxa de amostragem de 480000Hz e *bitrate* de 128 kbps. Créditos: MUSOPEN: `https://musopen.org/music/2617/frederic-chopin/waltz-in-am-b-150/`

O arquivo `campo01.mp4` está no formato MPEG-4 video com resolução 1920 x 1080, *codec* H.264. Créditos: Videezy.com: `http://www.videezy.com/`

Nro	Nome	Tamanho	Tipo
1	(Aya Higuchi) Chopin - Waltz in A minor, B. 150.mp3	2.1 MB	Arquivo de música (.mp3)
2	campo01.mp4	35.6 MB	Vídeo (.mp4)
3	folhas01.png	28.8 kB	Imagem (.png)
4	milho01.jpg	933.5 kB	Imagem (.jpg)
5	terra01.gif	927.0 kB	Imagem (.gif)
6	texto01.txt	61 B	Texto (.txt)
7	texto02.rtf	120.0 kB	Texto (.rtf)
8	relatorio01.pdf	2.5 MB	Documento (.pdf)

Tabela 5.1: Arquivos de teste

O arquivo `texto01.txt` contém a palavra ”TEXTO” repetida 10 vezes e o arquivo `texto02.rtf` contém uma sequência de número aleatórios entre 1 e 10000, de acordo com as Figuras 5.6 e 5.7.

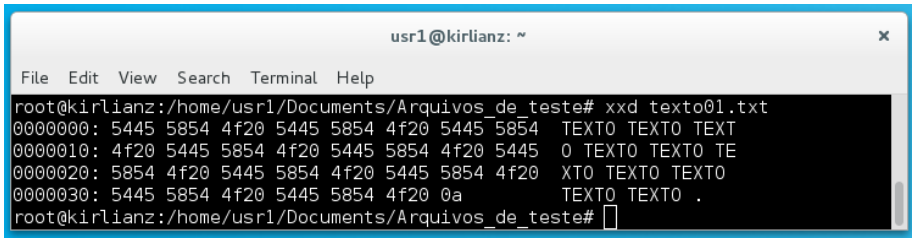


Figura 5.6: Conteúdo do arquivo de texto texto01.txt

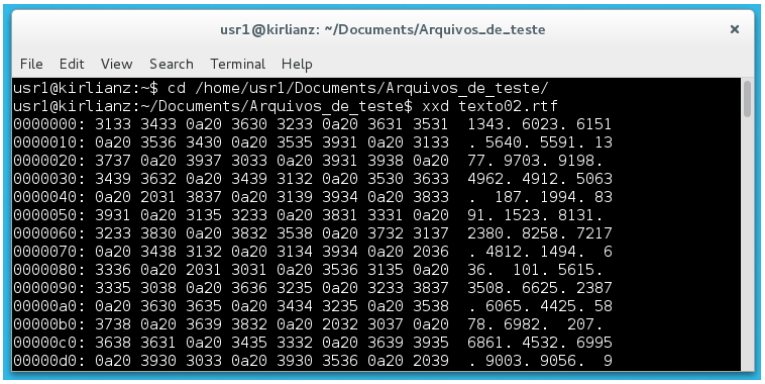


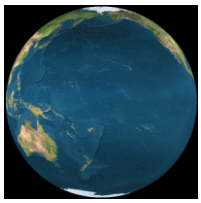
Figura 5.7: Parte do conteúdo do arquivo de texto texto02.rtf

O arquivo no formato .pdf é um relatório: *2015 State of Small Business Report* da Wasp Barcode Technologies. Fonte: `https://www.waspbarcode.com/static/waspbarcode/images/pdf/small-biz-report-0115-web.pdf`

Já os arquivos de imagens utilizadas constam na Figura 5.8.

(a) folhas01.png [9]

(b) milho01.jpg [20]



(c) terra01.gif [17]

Figura 5.8: Imagens utilizadas

Imagem de teste

A imagem de partição criada anteriormente foi então montada por meio do seguinte comando:

```
mount ssd01.img /mnt/SSD01
```

e os arquivos mencionados na Seção 5.3.2 copiados por meio do comando **cp**:

```
cp -a /home/usr1/Documents/Arquivos_de_teste/. /mnt/SSD01/
```

A imagem foi então desmontada com o comando **umount**:

```
umount /mnt/SSD01
```

Essa imagem servirá de base para os experimentos dos Cenários 1 e 2.

Neste experimento deseja-se avaliar a eficácia de técnicas de recuperação de dados comumente utilizadas em discos rígidos quando aplicadas em discos de estado sólido. O objetivo do experimento é verificar a influência de rotinas de *Garbage Collection* assim como outras rotinas internas do dispositivo que possam influenciar a recuperação de dados. O comando TRIM está **desabilitado** por padrão no Sistema Operacional utizado (*Linux*).

A imagem de partição `imagem_base.img` construída anteriormente foi copiada por meio do comando **cp**:

```
cp imagem_base ssd01.img
```

onde a imagem a ser trabalhada é a imagem `ssd01.img` que será substituída a cada teste, o que equivale a "zerar" o volume. A imagem foi montada por meio do comando **mount**:

```
mount ssd01.img /mnt/SSD01/
```

A partição montada com os arquivos é ilustrada pela Figura 5.9.

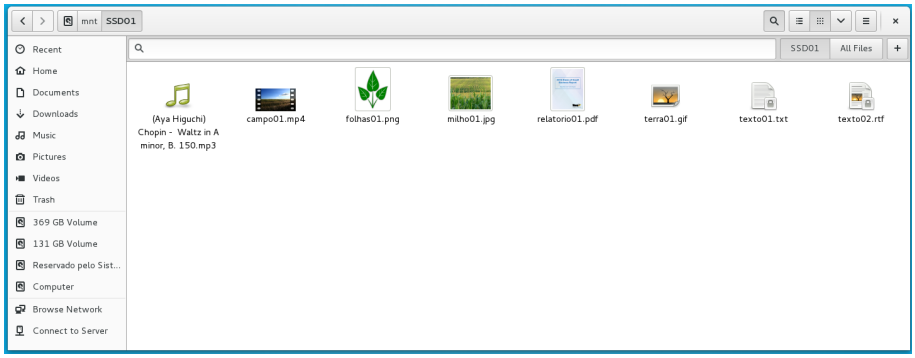


Figura 5.9: Partição de teste montada

Os arquivos da partição foram então removidos por meio do comando **rm**:

```
rm /mnt/SSD01/*
```

Que exclui todos os arquivos do diretório.

Após uma quantidade t de segundos, a partição foi desmontada e sobre a imagem obtida foram aplicadas as ferramentas de recuperação de dados descritas anteriormente, assim como princípios de *data carving*. A partição foi desmontada por meio do comando **umount**:

```
umount /mnt/SSD01/*
```

Realização do experimento

Foi criado um *script* para automatizar o processo de restauração, contendo os comandos mencionados anteriormente, referentes a montagem, deleção dos arquivos e desmontagem da imagem, cuja execução é exemplificada na Figura 5.10.

Figura 5.10: Exemplo de um teste do experimento 1

As seguintes ferramentas de recuperação de dados e *file carving* foram selecionadas para serem aplicadas sobre a imagem desmontada (em modo somente leitura):

- *extundelete* versão 0.2.4 com a biblioteca **libext2fs** versão 1.42.12 por N E Case [8];
- *foremost* foremost versão 1.5.7 por Jesse Kornblum, Kris Kendall, e Nick Mikus [27];
- *magicrescue* versão 1.1.9 por Jonas Jensen [28];
- *scalpel* versão 1.60 por Golden G. Richard III [25];
- *autopsy* versão 2.24 por Brian Carrier [5].

5.3.3 Resultados

Inicialmente planejava-se fazer uso de todas as ferramentas para a recuperação dos arquivos, porém na maioria dos casos a ferramenta *extundelete* foi suficiente. Os resultados do experimento podem ser verificados na Tabela 5.2.

A coluna "Arquivos" contém os números dos arquivos de acordo com a Tabela 5.1.

5.3.4 Análise dos resultados

Foi possível recuperar todos os arquivos excluídos, até um tempo $t = 4500s$ (1 hora e 15 minutos) após a exclusão, no *Linux*. É um resultado positivo para a Forense Digital, considerando que grande parte dos sistemas *Linux* ainda não adota o uso do comando TRIM em suas formas automáticas conforme a seção 3.3.2.

		Arquivos							
t(s)	n	1	2	3	4	5	6	7	8
0	10	100%	100%	100%	100%	100%	100%	100%	100%
1	10	100%	100%	100%	100%	100%	100%	100%	100%
2	10	100%	100%	100%	100%	100%	100%	100%	100%
3	10	100%	100%	100%	100%	100%	100%	100%	100%
4	10	100%	100%	100%	100%	100%	100%	100%	100%
5	10	100%	100%	100%	100%	100%	100%	100%	100%
6	10	100%	100%	100%	100%	100%	100%	100%	100%
7	10	100%	100%	100%	100%	100%	100%	100%	100%
8	10	100%	100%	100%	100%	100%	100%	100%	100%
9	10	100%	100%	100%	100%	100%	100%	100%	100%
10	5	100%	100%	100%	100%	100%	100%	100%	100%
20	5	100%	100%	100%	100%	100%	100%	100%	100%
30	5	100%	100%	100%	100%	100%	100%	100%	100%
40	5	100%	100%	100%	100%	100%	100%	100%	100%
50	5	100%	100%	100%	100%	100%	100%	100%	100%
100	5	100%	100%	100%	100%	100%	100%	100%	100%
200	5	100%	100%	100%	100%	100%	100%	100%	100%
300	5	100%	100%	100%	100%	100%	100%	100%	100%
400	5	100%	100%	100%	100%	100%	100%	100%	100%
500	5	100%	100%	100%	100%	100%	100%	100%	100%
600	5	100%	100%	100%	100%	100%	100%	100%	100%
1200	1	100%	100%	100%	100%	100%	100%	100%	100%
1500	1	100%	100%	100%	100%	100%	100%	100%	100%
3000	1	100%	100%	100%	100%	100%	100%	100%	100%
4500	1	100%	100%	100%	100%	100%	100%	100%	100%

Tabela 5.2: Resultados obtidos nos experimentos do Cenário 1
$t(s)$: tempo em segundos; n: número de vezes que o experimento foi realizado

Em se tratando de anti-forense, usuários dos sistemas *Linux* que são indiferentes a fatores não facilmente observáveis como funcionamento da mídia de armazenamento de sua máquina, por exemplo, provavelmente serão indiferentes também em relação ao TRIM. Entretanto, à medida que os SOs forem evoluindo e o uso dos SSDs seja mais difundido, acredita-se que os os SOs da família *Linux* venham a diminuir o distanciamento com os SSDs, habilitando o comando de forma automática.

5.4 Cenário 2

5.4.1 Descrição

Neste experimento deseja-se verificar o impacto do comando TRIM na recuperação de dados (arquivos) quando aplicado a uma partição contendo arquivos excluídos por meio de deleção simples.

O comando TRIM No Sistema Operacional em questão (Linux) é desabilitado por padrão quando uma partição é montada sem a opção *discard*. Logo nesse experimento será aplicado o comando **TRIM** manual e seus efeitos serão estudados. Dessa forma o experimento pode ser sumarizado pela exclusão de arquivos, seguida do envio do comando TRIM manual após um tempo t e posterior tentativa de recuperação com o objetivo de verificar o efeito do comando TRIM e atuação da coleta de lixo.

5.4.2 Procedimentos

Realização do experimento

O experimento consiste no uso da mesma imagem `imagem_base` do experimento anterior contendo os arquivos da Tabela 5.1. A imagem foi montada, os arquivos removidos e após uma quantidade t de tempo foi enviado o comando TRIM para a partição e esta foi desmontada. Os procedimentos anteriores ao envio do comando TRIM nesse experimento foram os mesmos do Cenário 1.

A sequência de comandos executados foi:

```
cp imagem_teste ssd01.img
mount ssd01.img /mnt/SSD01/
rm /mnt/SSD01/*
```

Após t segundos:

```
/sbin/fstrim /mnt/SSD01/ -v
umount /mnt/SSD01/
```

5.4.3 Resultados

Os resultados obtidos podem ser verificados na Tabela 5.3.

		Ferramentas				
t(s)	n	*extundelete*	*foremost*	*magicrescue*	*scalpel*	*Autopsy*
0	10	100%	-	-	-	-
1	10	100%	-	-	-	-
2	10	100%	-	-	-	-
3	10	100%	-	-	-	-
4	10	100%	-	-	-	-
5	10	100%	-	-	-	-
6	10	*	0%	0%	0%	*
7	10	*	0%	0%	0%	*
8	10	*	0%	0%	0%	*
20	10	*	0%	0%	0%	*
60	10	*	0%	0%	0%	*
150	10	*	0%	0%	0%	*

Tabela 5.3: Resultados obtidos nos experimentos do Cenário 2
$t(s)$: tempo em segundos; n: número de vezes que o experimento foi realizado
* apenas os nomes e tamanho dos arquivos foram recuperados

Nos casos onde $t = 1, 2, 3, 4, 5$, foi possível recuperar todos os arquivos com a ferramenta *extundelete* que utiliza o *Journal* e metados para realizar a recuperação dos arquivos. Logo foi desnecessário utilizar as demais ferramentas.

A partir de 6 segundos após a exclusão dos arquivos e envio do comando TRIM por meio do comando `/sbin/fstrim /mnt/SSD01/ -v`, apenas conseguiu-se obter os arquivos com seus respectivos nomes com a ferramenta *extundelete*, porém seus conteúdos estavam "zerados" - não foi possível abrir, visualizar ou reproduzir os arquivos com suas ferramentas usuais. A feramenta *Autopsy* apenas identificou os arquivos deletados porém os identificou como possuindo tamanho 0. Já as ferramentas *foremost*, *magicrescue* e *scalpel* não apresentaram resultados para os seus tipos de arquivos reconhecidos.

A Figura 5.11 apresenta o resultado de uma recuperação descrita no parágrafo anterior.

O resultado obtido pela ferramenta *extundelete* foi o mais próximo de alguma recuperação dos arquivos excluídos. A Figura 5.12 mostra o conteúdo do arquivo de texto `texto01.txt` após a recuperação.

Os arquivos obtidos também tiveram seus cabeçalhos escritos com 0's.

Diante dos resultados obtidos para $t \geq 6$, foi refeito um caso de teste onde os arquivos obtidos estavam zerados e comparou-se a imagem da partição desmontada com a imagem

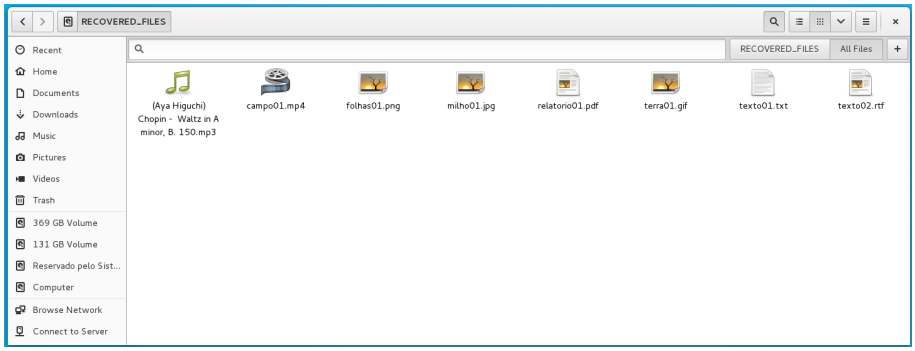


Figura 5.11: Resultado da tentativa de recuperação de dados com a ferramenta *extundelete* do experimento 2 onde $t \geq 6$

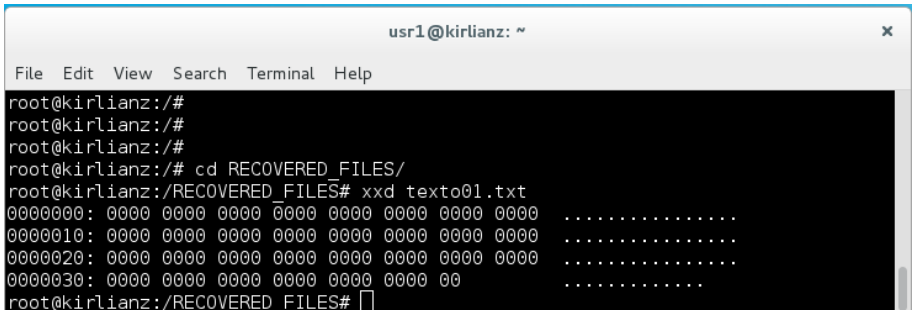


Figura 5.12: Conteúdo do arquivo texto01.txt

original `imagem_base.img` sendo utilizado para tal um editor Hexadecimal com a função *diff*. Verificou-se que os endereços na imagem correspondentes aos blocos de dados estavam escritos com $0's$. As Figuras 5.13 e 5.14 contêm a diferença descrita, para o caso do arquivo de texto `texto01.txt`:

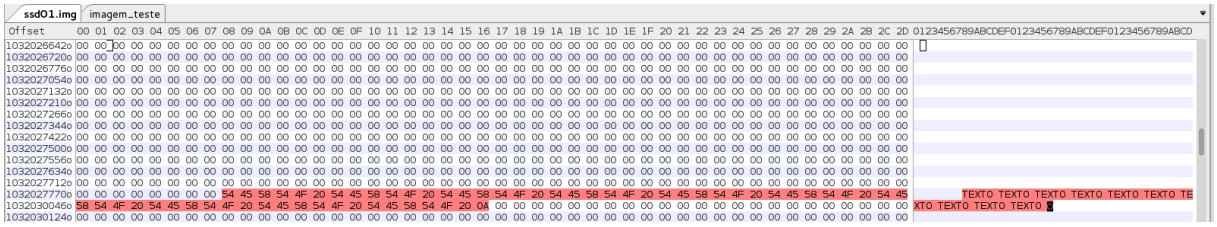


Figura 5.13: Conteúdo do arquivo texto01.txt na imagem de partição de teste

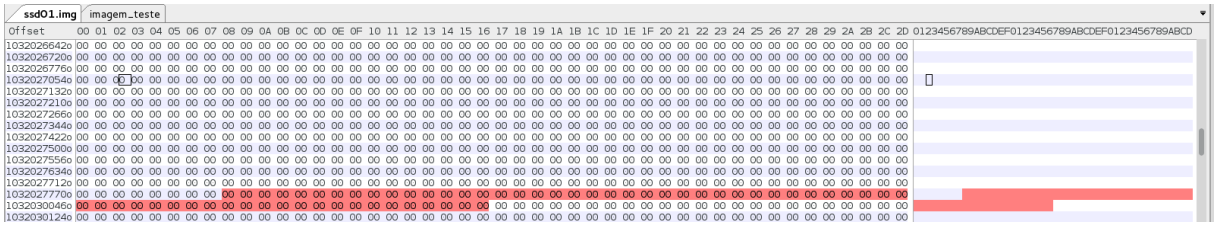


Figura 5.14: Conteúdo do arquivo texto01.txt na imagem de partição após a deleção e o comando TRIM

5.4.4 Análise dos resultados

Os efeitos do TRIM na recuperação de arquivos excluídos se mostraram agressivos. Resultado notadamente negativo para a prática de Forense Digital via *software* e segundo técnicas *black-box*. A janela de oportunidades para a recuperação de arquivos excluídos é menor que 6 segundos. Usuários avançados dos sistemas *Linux* encontram no TRIM manual uma forma eficaz e eficiente de anti-forense.

5.5 Cenário 3

5.5.1 Descrição

Este cenário consiste na exclusão de arquivos, sem a atuação do comando TRIM em qualquer forma e posterior tentativa de recuperação, com o objetivo de verificar a atuação da coleta de lixo. O Sistema Operacional é o *Windows 7*.

5.5.2 Procedimentos

Para a realização dos experimentos correspondentes aos Cenários 3, 4 e 5 foram seguidos os procedimentos descritos a seguir.

Criação de um volume

Foi criado um volume utilizando a ferramenta "Gerenciamento de Disco do Windows" como ilustram as Figuras 5.15 a 5.17.

O volume de testes possui tamanho de 1024 MB, e Sistema de Arquivos **NTFS** com tamanho do bloco de 4 kB, formatado completamente (escrita com 0s). Após criado, o volume aparece como utilizável pelo *Windows*, como ilustra a Figura 5.18.

O volume C é o que contém o SO, está instalado no SSD, assim como o volume Z. Já os volumes D e E estão situados em um HDD.

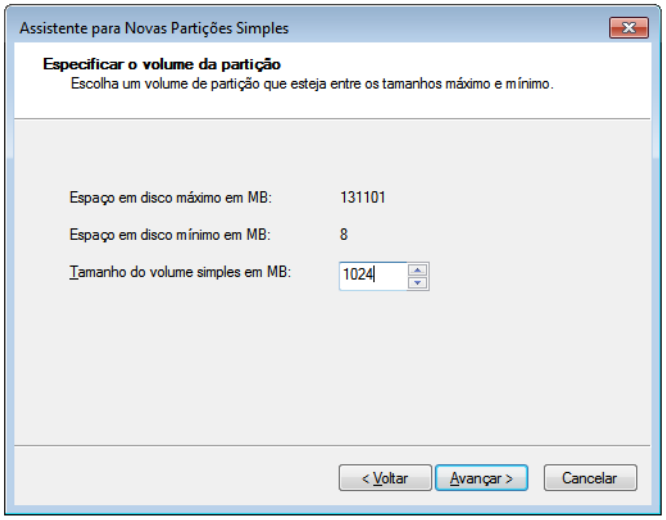


Figura 5.15: Escolha do tamanho do volume

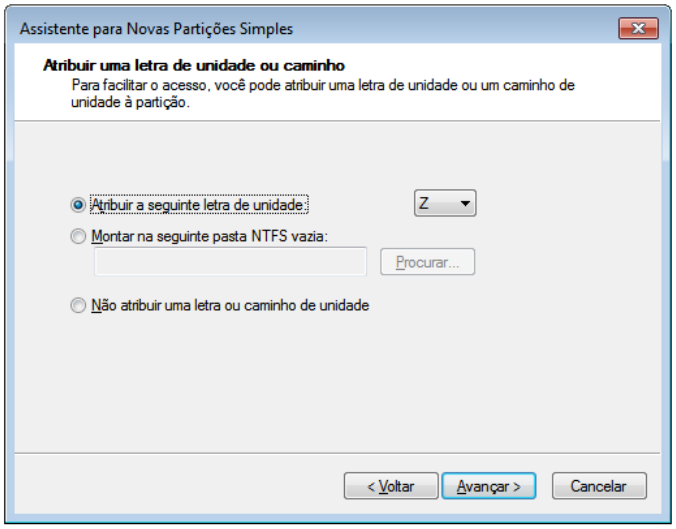


Figura 5.16: Atribuição de uma letra ao volume

Foram selecionados os mesmos arquivos dos experimentos dos Cenários 1 e 2, ou seja, os da Tabela 5.1. Esses foram copiados para o novo volume como ilustra a Figura 5.19.

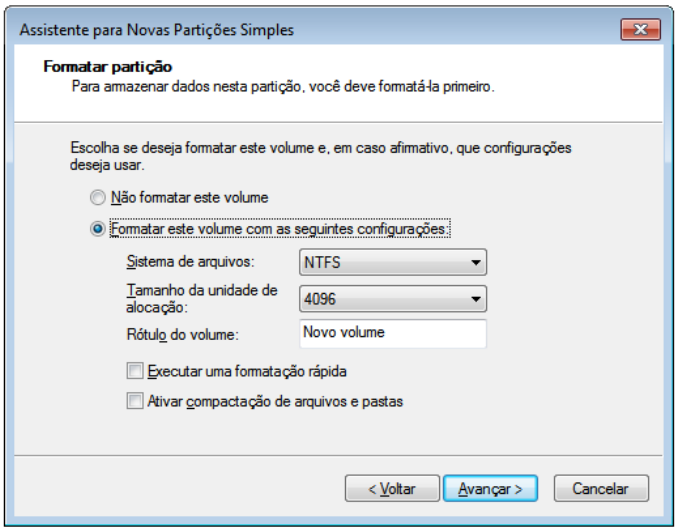


Figura 5.17: Escolha do tamanho do bloco e formatação do volume

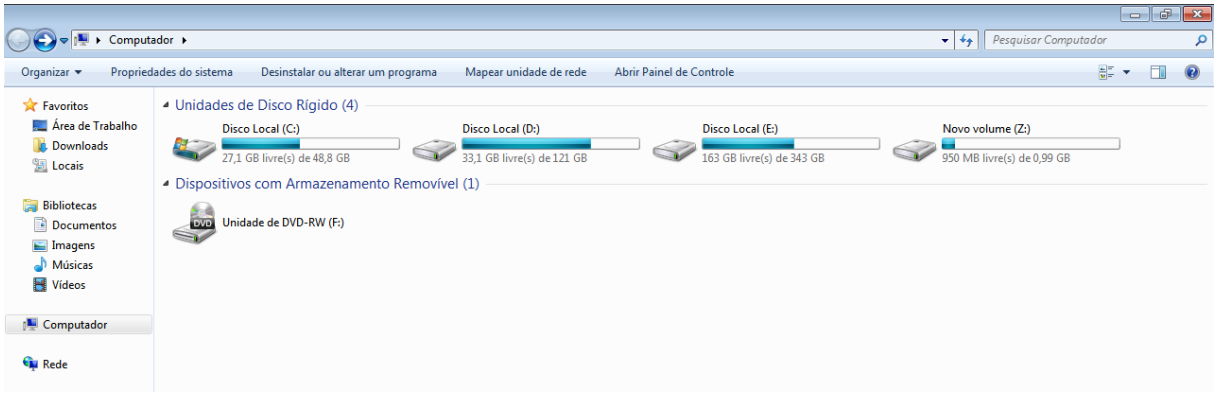


Figura 5.18: Volume Z montado

Cópia dos arquivos para o volume

Para a cópia dos arquivos para o volume Z foi utilizado o utilitário **xcopy** do *Windows*, como por exemplo:

```
xcopy /s E:\Arquivos_de_teste Z:\
```

que copia todos os arquivos do volume E para o volume Z.

Para exclusão dos arquivos foi utilizada a exclusão manual "permanente" do *Windows*. Para a criação da imagem de volume foi utilizado o programa *ProDiscover Basic* versão 7.0 por The ARC Group. Já para a recuperação dos arquivos foram utilizadas 2 ferramentas:

- *The Sleuth Kit* versão 4.2.0 por Brian Carrier [6];
- *FTK Imager* versão 3.4.2 por ACCESS DATA. [16]

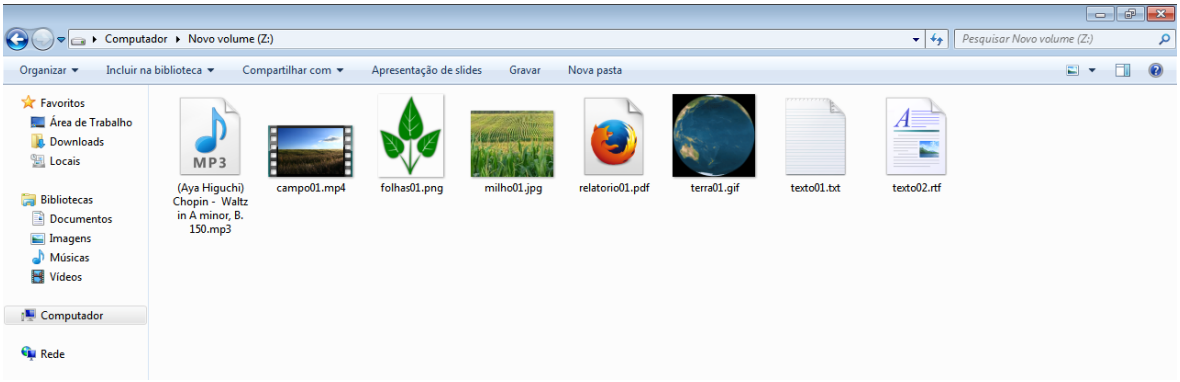


Figura 5.19: Arquivos no volume Z

Por motivos de consistência com os experimentos anteriores foram selecionadas apenas ferramentas que possibilitam a recuperação sob uma imagem de disco, volume ou partição. Não foram utilizadas ferramentas de recuperação que recuperam arquivos diretamente do disco e não de uma imagem de disco.

Para verificar a integridade dos arquivos recuperados, esses foram abertos e/ou executados em seus programas padrão, como visualizador de texto, imagem etc. Para todos os testes, todos os arquivos foram abertos e/ou executados. Em alguns casos também verificou-se seus conteúdos por meio de um editor hexadecimal. Vale ressaltar que os arquivos foram recuperados para um disco diferente (HDD).

Realização do experimento

Nesse experimento deseja-se verificar a atuação da coleta de lixo do SSD após a exclusão de arquivos, com o comando TRIM automático desabilitado. Para desativá-lo, foi feito o procedimento da Figura 5.20.

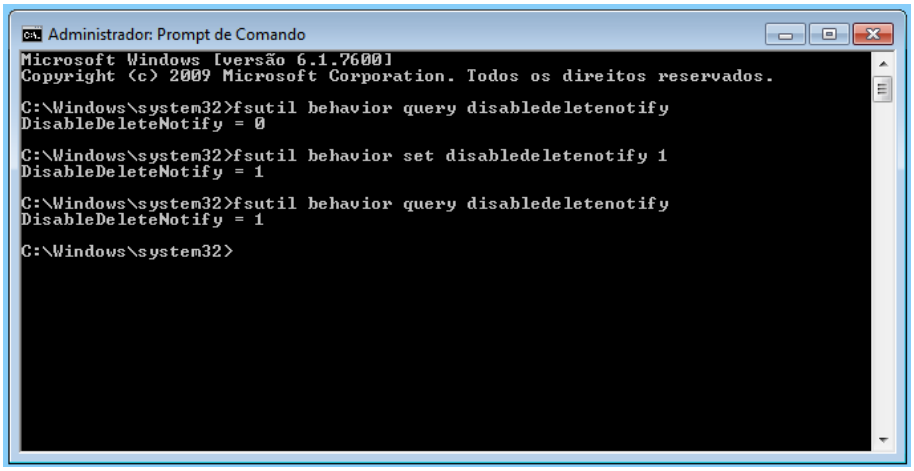


Figura 5.20: Desativação do envio automático do comando TRIM no *Windows 7*

O comando `fsutil query disabledeletenotify` requisita o estado da variável `disabledeletenotify`. Um valor 1 indica que a notificação de blocos excluídos está desativada, já um valor 0 indica o oposto. Já o comando `fsutil set disabledeletenotify valor` atribui o valor desejado a variável.

Foram encontradas dificuldades para automatização dos testes no *Windows*. Dessa forma, grande parte do procedimento de teste foi realizada de forma "manual". Um caso de teste nesse experimento consiste em:

1. Criação do volume Z conforme descrito na seção 5.5.2;
2. Cópia dos arquivos de teste de um diretório por meio do comando **xcopy**;
3. Exclusão dos arquivos de forma "permanente";
4. Após um tempo t foi criada uma imagem do volume Z por meio do programa *ProDiscover Basic*. Essa imagem é salva no HDD;
5. Tentou-se recuperar a partir das imagens, os arquivos excluídos. A saída dos programas de recuperação (os arquivos recuperados) são duas pastas contidas no HDD, uma para cada ferramenta;
6. Tentou-se abrir e/ou executar os arquivos recuperados de acordo com seu tipo. Em alguns casos foi utilizado um editor hexadecimal para verificar o conteúdo dos arquivos não abríveis ou não executáveis.

5.5.3 Resultados

Foram realizados 10 casos de teste e seus resultados constam na Tabela 5.4.

Os arquivos foram, em geral, recuperados. Para $t = 5s$ o arquivo 3 (`folhas01.png`) estava com seu conteúdo "zerado". Já o arquivo de vídeo 2 (`campo01.mp4`) apresentou 61.9% de seu conteúdo original, enquanto que o restante estava "zerado". O dado espúrio indica uma execução aparentemente aleatória das rotinas de coleta de lixo, sendo também possível um erro na formação da imagem de disco ou algum outro evento não previsto, como por exemplo um comando TRIM anteriormente na controladora.

5.5.4 Análise dos resultados

Nos testes realizados, também verificou-se que um dos arquivos foi recuperado de forma parcial e outro não foi recuperado. Nesse caso verificou-se a atuação da coleta de lixo, até então ausente nos experimentos realizados no *Linux*. Levanta-se a hipótese de que

		Arquivos							
t(s)	n	1	2	3	4	5	6	7	8
1	1	100%	100%	100%	100%	100%	100%	100%	100%
2	1	100%	100%	100%	100%	100%	100%	100%	100%
3	1	100%	100%	100%	100%	100%	100%	100%	100%
4	1	100%	100%	100%	100%	100%	100%	100%	100%
5	1	100%	**61.9%**	**0%**	100%	100%	100%	100%	100%
10	1	100%	100%	100%	100%	100%	100%	100%	100%
20	1	100%	100%	100%	100%	100%	100%	100%	100%
30	1	100%	100%	100%	100%	100%	100%	100%	100%
60	1	100%	100%	100%	100%	100%	100%	100%	100%
120	1	100%	100%	100%	100%	100%	100%	100%	100%
180	1	100%	100%	100%	100%	100%	100%	100%	100%
360	1	100%	100%	100%	100%	100%	100%	100%	100%
600	1	100%	100%	100%	100%	100%	100%	100%	100%
1200	1	100%	100%	100%	100%	100%	100%	100%	100%
1800	1	100%	100%	100%	100%	100%	100%	100%	100%
3600	1	100%	100%	100%	100%	100%	100%	100%	100%

Tabela 5.4: Resultados obtidos nos experimentos do Cenário 3
$t(s)$: tempo em segundos; n: número de vezes que o experimento foi realizado

a coleta de lixo nos Sitemas *Windows* com **NTFS** não depende diretamente do tempo de exclusão e sim de um mínimo de blocos excluídos, sendo que essa quantidade não foi determinada no experimento realizado.

A hipótese do TRIM estar desativado no sistema *Windows* é bastante improvável, dado que o padrão é que esteja habilitado nos SOs mais recentes da Microsoft. Logo, trata-se de um resultado de pouca relevância para a Forense Digital, pois argumenta-se da mesma forma que no Cenário 1 no que se refere a usuários comuns: dificilmente um usuário comum desses sistemas iria desativar a execução automática do comando. Mesmo que o faça, os dados ainda poderiam ser irrecuperáveis por conta da atuação de uma coleta de lixo independente.

5.6 Cenário 4

5.6.1 Descrição

Experimento similar ao anterior, sendo que a diferença reside na ativação do TRIM automático antes da realização dos testes. Dessa forma o caso de teste é o mesmo do Cenário 3, alterando-se apenas alguns valores de t. Os resultados constam na Tabela 5.5.

5.6.2 Procedimentos

Realização do experimento

Os procedimentos realizados para os testes correspondentes a este cenário foram os mesmos do cenário anterior, com exceção de que o comando TRIM encontrou-se habilitado na forma automática (agendada).

5.6.3 Resultados

		Arquivos							
t(s)	n	1	2	3	4	5	6	7	8
1	1	100%	100%	100%	100%	100%	100%	100%	100%
2	1	100%	100%	100%	100%	100%	100%	100%	100%
3	1	100%	100%	100%	100%	100%	100%	100%	100%
4	1	**0%**	100%	**0%**	**0%**	100%	100%	**0%**	100%
5	1	100%	**46.7%**	100%	100%	100%	100%	100%	100%
6	1	100%	**5.4%**	100%	100%	100%	100%	100%	100%
7	1	100%	100%	100%	100%	100%	100%	100%	100%
8	1	100%	100%	100%	100%	100%	100%	100%	100%
9	1	**0%**	100%	**0%**	**0%**	100%	100%	**0%**	100%
10	1	100%	100%	100%	100%	100%	100%	100%	100%
15	1	**0%**	100%	**0%**	**0%**	100%	100%	**0%**	100%
20	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
30	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
60	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
120	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
180	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
360	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
600	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
1200	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
1800	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
3600	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**

Tabela 5.5: Resultados obtidos nos experimentos do Cenário 4
$t(s)$: tempo em segundos; n: número de vezes que o experimento foi realizado

Da Tabela 5.5 nota-se que:

- Até 3 segundos após a exclusão, todos os arquivos foram recuperados
- Para $t = 5,6$ o arquivo 2 (vídeo) foi recuperado parcialmente

- Para $t = 4, 9, 15$ os arquivos 1 (`(Aya Higuchi) Chopin - Waltz in A minor B. 150.mp3`), 3 (`folhas01.png`), 4 (`milho01.jpg` e 7 (`texto02.rtf`) encontravam-se zerados enquanto que os demais, íntegros. No caso do arquivo 7, foi possível abri-lo com programas usuais por conta de sua extensão
- Para $t \geq 20$, não foi possível recuperar os arquivos, exceto o arquivo 6 (`texto01.txt`)
- O arquivo 6 (`texto01.txt`) foi recuperado em todos os testes

5.6.4 Análise dos resultados

Nesse experimento verificou-se que a execução automática do TRIM pelo *Windows* surtiu efeitos consideráveis na recuperabilidade dos arquivos. Ocorreram casos de teste onde entre conjuntos de arquivos apenas alguns são recuperados, outros onde um arquivo foi truncado (parcialmente "zerado"), até um tempo $t = 15s$. A partir de $t = 20$ tem-se a irrecuperabilidade de todos os arquivos, exceto um arquivo de texto de 61 *bytes*.

A interação dos arquivos muito pequenos com o TRIM e a coleta de lixo provavelmente se dá de forma peculiar. Conjectura-se que tais arquivos sejam armazenados em locais especiais nas estruturas de dados internas do SSD não sendo sucestíveis aos efeitos do TRIM. Também é possível a hipótese de que arquivos desse tipo sejam acumulados para exclusão de forma distinta dos arquivos de tamanho de outras ordens de grandeza.

No que tange à Forense Digital, trata-se de um resultado negativo, pelos motivos já citados na análise do Cenário 3. Janelas de oportunidade menores que 20s ou a recuperação de arquivos com tamanhos nas casas dos *bytes* são resultados nada otimistas para a Forense Digital dos SSDs em sistemas *Windows*.

5.7 Cenário 5

5.7.1 Descrição

Nesse experimento deseja-se avaliar a influência do TRIM manual, sem a presença do TRIM automático, na recuperação de arquivos excluídos. Não existe forma nativa de execução manual do TRIM no *Windows 7*, diferentemente do *Linux*. Dessa forma, foi necessário utilizar um programa (Samsung Magician) próprio do fabricante que veio num CD acompanhado do SSD, sendo esse *software* de fácil obtenção na rede. A Figura 5.21 ilustra uma tela do programa, essa de maior importância para o experimento, por ser responsável pela execução manual do TRIM.

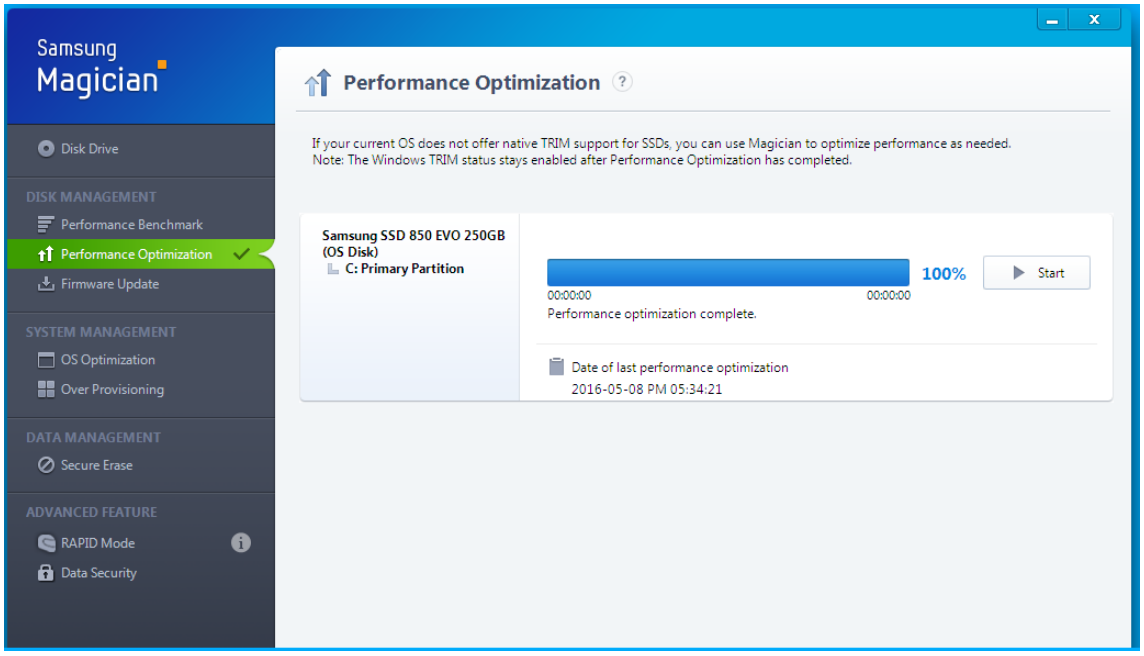


Figura 5.21: Tela do programa Samsung Magician

5.7.2 Procedimentos

Realização do experimento

O caso de teste é similar ao dos Cenários 3 e 4 porém difere pela execução do TRIM após a exclusão dos arquivos, no passo 4:

1. Criação do volume Z conforme descrito na seção 5.5.2;
2. Cópia dos arquivos de teste de um direction por meio do comando **xcopy**;
3. Exclusão dos arquivos de forma "permanente";
4. Após um tempo t foi enviado o comando TRIM (*Performance Optimization*) por meio do programa Samsung Magician;
5. Logo após o passo 4, foi criada uma imagem do volume Z por meio do programa *ProDiscover Basic*. Essa imagem é salva no HDD;
6. Tentou-se recuperar a partir das imagens, os arquivos excluídos. A saída dos programas de recuperação (os arquivos recuperados) são duas pastas contidas no HDD, uma para cada ferramenta;
7. Tentou-se abrir e/ou executar os arquivos recuperados de acordo com seu tipo. Em alguns casos foi utilizado um editor hexadecimal para verificar o conteúdo dos arquivos não abríveis ou não executáveis.

5.7.3 Resultados

Os resultados do exeperimento constam na Tabela 5.6.

		Arquivos							
t(s)	n	1	2	3	4	5	6	7	8
1	1	**0%**	100%	**0%**	**0%**	100%	100%	**0%**	100%
2	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
3	1	**0%**	100%	**0%**	**0%**	100%	100%	**0%**	100%
4	1	**0%**	100%	**0%**	**0%**	100%	100%	**0%**	100%
5	1	**0%**	100%	**0%**	**0%**	100%	100%	**0%**	100%
10	1	**25.2%**	100%	**0%**	**0%**	100%	100%	**0%**	100%
20	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
30	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
60	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
120	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
180	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
360	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
600	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
1200	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
1800	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**
3600	1	**0%**	**0%**	**0%**	**0%**	**0%**	100%	**0%**	**0%**

Tabela 5.6: Resultados obtidos nos experimentos do Cenário 5
$t(s)$: tempo em segundos; n: número de vezes que o experimento foi realizado

Da Tabela 5.6 nota-se que:

- Para $t = 10$ tem-se a recuperação parcial do arquivo 1 (`Aya Higuchi) Chopin - Waltz in A minor, B. 150.mp3`). Foi possível executá-lo, porém a música estava truncada;
- Para $t = 1, 3, 4, 5, 10$ os arquivos 2 (`campo01.mp4`), 5 (`terra01.gif`) e 8 (`relatorio01.pdf`) foram recuperados totalmente;
- Para $t \geq 20$, similarmente ao Cenário 4, não foi possível recuperar os arquivos, exceto o arquivo 6 (`texto01.txt`);
- O arquivo 6 (`texto01.txt`) foi recuperado em todos os testes.

5.7.4 Análise dos resultados

Nesse experimento verifica-se que a execução manual do comando TRIM com a sua forma automática desabilitada levou à irrecuperabilidade da maioria dos arquivos para

$t \geq 20s$, de forma similar à do experimento 4. Para tempos menores que $20s$, alguns arquivos foram também excluídos. O arquivo de texto de 61 *bytes* foi recuperado em todos os testes, contribuindo para as hipótestes levantadas na análise do experiment anterior.

No que se refere à Forense Digital, trata-se de um resultado negativo, e também otimista (no sentido de que a situação na prática poderia ser pior ainda), pois para esse experimento o TRIM automático foi desativado. Nada impede um usurious de manter o TRIM automático e enviar um comando manual utilizando um *software* próprio do fabricante.

5.8 Análise dos resultados entre os diferentes cenários

Ns Cenários 1 e 2, no *Linux*, não houve recuperação parcial. No Cenário 1 a taxa de recuperação foi de 100% e no Cenário 2 foi de 0% para $t \geq 6s$. Nos experimentos no *Windows*: o Cenário 3 apresentou em geral recuperação total dos arquivos, com exceção de um caso de teste em que houve a recuperação parcial de um arquivo e a não recuperação de outro. A hipótese provável é que a coleta de lixo tenha entrado em ação.

No Cenário 4 verifica-se uma diferença na janela de oportunidades para a recuperação em contraste com a do Cenário 2. No último, a janela de oportunidades é menor que $6s$ (recuperação total), enquanto que nos primeiros, é menor que $3s$ (recuperação parcial). Para $4 \leq t < 20$ a recuperação é incerta. No Cenário 5 verifca-se situação ainda pior que a do Cenário 4. Naquele, teoricamente a janela de oportunidades é menor que $1s$, porém não é possível afirmar o que acontece nesses casos.

Uma diferença fundamental entre os dois grupos de cenários (1,2) e (3,4,5) é que nos experimentos no *Windows* (3,4 e 5), foi possível recuperar o arquivo de texto mesmo após o TRIM, em todos os casos. No Cenário 2 do *Linux* esse arquivo não foi possível ser recuperado para $t \geq 6s$ após execução manual do TRIM, pois no *Linux* o comando TRIM atuou de forma diferente, "zerando"todos os arquivos.

5.9 Comparação com outros trabalhos

Quanto aos resultados de Bell e Boddington [2], esses não testaram a influência do comando TRIM, pois utilizaram o SO *Windows XP*, testando apenas a influência da coleta

de lixo. Os experimentos realizados pela dupla não foram confirmados pelos resultados dos Experimentos referentes aos cenários 1 e 3 realizados, que excluem a atuação do TRIM.

Quanto aos resultados de Gebremaryam [22], os resultados obtidos nos experimentos realizados se alinham, exceto para o SSD Kingston SSDNow V 100, com os do teste do TRIM.

Os resultados de King e Vidas [31] apresentaram grande variância. Os resultados obtidos nos experimentos do Cenário 4 quando considerados tempos $t > 20$ se alinham aos da dupla para SSDs da marca Intel, OCZ1, Corsair e Crucial. Quanto aos experimentos realizados no *Linux*, foi testado um SO que não suporta o TRIM. Nesse ponto os resultados obtidos nos experimentos do Cenário 1 divergem dos obtidos para o SSD da marca PQI1, RiData, Patriot, Kingstom e Intel. Vale ressaltar que foram utilizados diferentes tamanhos de arquivos, 900 kB e 650 MB nos experimentos da dupla do referido trabalho.

Os resultados de Bonetti et al. [21] se alinham aos resultados obtidos para o TRIM no *Windows* e no *Linux* (Cenários 2 e 4), para os SSDs Samsung S470 e Crucial M4. Já para o SSD Corsair F60 a taxa de recuperação foi de 70.79%, no caso do *Windows*. No caso da análise da recuperabilidade na ausência do TRIM, os resultados obtidos nos experimentos dos Cenário 1 e 3 se alinham aos obtidos pelos autores, se excluído o dado espúrio dos resultados para o Cenário 3. Os dois SSDs que alegadamente suportavam coleta de lixo não a realizaram mesmo horas após a exclusão, segundo os autores.

Por fim, quanto aos resultados de Nisbet et al. [38], não cabe determinar ou não a divergência nos resultados pois não é especificado pelos autores os tempos exatos da tentativa de recuperação no primeiro conjunto de resultados apresentados. Pode-se afirmar o alinhamento ou a divergência dependendo da escolha do tempo. Os demais resultados tratam-se de tentativas de recuperação após 1 hora e após 5 horas.

É importante fazer algumas considerações sobre as confirmações ou contradições apresentadas. A primeira delas é que, obviamente, tratam-se de diferentes dispositivos, com diferentes características. Constatemente são desenvolvidas novas tecnologias de *hardware* para os SSDs, assim como são elaborados algoritmos mais eficientes para prolongar o tempo de vida dos dispositivos. Os Sistemas Operacionais se alinham a esse cenário, interagindo com os SSDs de forma cada vez mais próxima.

A segunda é que diferentes meios de análise Forense e de recuperação foram empregados e mesmo que tenham sido empregados métodos similares, podem se diferenciar quanto às ferramentas utilizadas e quanto ao ambiente computacional alvo dos experimentos.

A discussão realizada nessa Seção foi sumarizada na Tabela 5.7.

Autores	Resultados obtidos pelos autores	Comparação com o trabalho realizado
Bell & Boddington	Experimentos de recuperabilidade realizados no sistema *Windows XP* sem o TRIM, sem sucesso	Resultados obtidos pela dupla refutados pelos resultados dos experimentos dos Cenários 1 e 3
Gebremaryam	Experimentos no sistema *Windows 7* na presença (sem sucesso) e na ausência do TRIM (bem sucedidos)	Resultados se alinharam
King & Vidas	Experimentos nos sistemas *Windows* e *Linux*, vasta gama de SSDs com resultados variados	Confirmados para SSDs de algumas marcas e refutados por outros de outras
Bonetti et al.	Experimentos nos sistemas *Windows* e *Linux*, na presença do TRIM (sem sucesso) e na ausência do TRIM (bem sucedidos)	Resultados se alinharam, exceto para um modelo de SSD
Nisbet et al.	Experimentos nos sistemas *Windows*, *Linux* e no *Mac OS X* com resultados variados	Não coube comparação ($1hr \leq t \leq 5hrs$)

Tabela 5.7: Comparação com outros trabalhos

Capítulo 6

Conclusão

Neste trabalho foram estudadas as características e mecanismos internos dos SSDs que podem inviabilizar ou dificultar a prática usual de recuperação forense e de forense computacional em dois Sistemas Operacionais, *Windows* e *Linux*.

Os resultados foram variados, sendo que o fator determinante em ambos os sistemas é o comando TRIM, que após ser manualmente enviado pelo usuário e executado pela controladora, inviabiliza a recuperação de dados. Já no sistema *Windows*, tanto a execução automática quanto a manual do comando TRIM apresentaram resultados negativos para a recuperação forense, à exceção de um arquivo de texto de tamanho na faixa dos *bytes*.

A partir das análises realizadas conclui-se que o cenário das práticas de Forense Digital aplicadas aos discos de estado sólido ainda é incerto, porém pessimista.

O fato de os SSDs terem ganhado e continuarem ganhando cada vez mais espaço no mercado onde tinha-se predominância dos discos rígidos, aliado ao constante avanço tecnológico na área, torna árdua a tarefa da Forense Digital de acompanhar a nova realidade dos dispositivos de armazenamento secundário.

Diante da pergunta: *É possível recuperar arquivos excluídos em um SSD?* uma resposta adequada seria: *Depende. Provavelmente não.* Não com as técnicas atuais comumente aplicadas aos discos rígidos. Inúmeros fatores orbitam a questão: modelos e marcas específicas, sistemas computacionais, Sistemas Operacionais, Sistemas de Arquivos e usuários são alguns desses fatores. Combinados entre si, as possibilidade são diversas.

O usuário do sistema computacional desempenha um papel-chave nessa problemática. Um usuário com um mínimo de conhecimento sobre a tecnologia que envolve os SSDs bem

como sobre Sistemas de Arquivos pode aliar técnicas de anti-forense às funcionalidades e aos mecanismos de um SSD de modo a tornar o processo de exclusão permanente de arquivos um processo relativamente simples e eficiente.

A prática de crimes informáticos, que utilizam os SSDs como meio ou como fim para sua realização, inegavelmente é beneficiada com os Joseito dos mecanismos internos de um SSD, concebidos para aumentar o tempo de vida e uso do dispositivo. Embora já existam métodos consolidados de remoção segura e de anti-forense, o uso por si só de um SSD já pode ser considerado, até certo ponto, como uma medida antecipada de anti-forense.

Há esperança de que fatores como algoritmos, mecanismos e funcionamento dos discos de estado sólido se uniformizem no futuro, à medida que o mercado se expande, e à medida que são feitos avanços na área de engenharia reversa, levando a uma melhor elucidação das características dos SSDs e consequentemente à avanços nas práticas forenses digitais. Acredita-se, entretanto, que o futuro destas resida no desenvolvimento de estudos e técnicas voltadas para a parte de *hardware*, em vez do caminho via *software*.

6.1 Trabalhos Futuros

Como já mencionado, acredita-se que o caminho a ser seguido para a recuperação forense envolvendo SSDs seja via *hardware*. No entanto, para uma maior elucidação dos efeitos do uso dos SSDs nas práticas forenses, foram levantadas algumas possibilidades de trabalhos futuros com enfoque na parte de *software*.

Como a execução do comando TRIM levou a expressivos resultados negativos para a recuperação dos arquivos, especialmente no sistema *Linux*, em trabalhos futuros poderá ser estudada como se dá a interação do Sistema Operacional com a controladora do SSD nos instantes que se seguem após o envio do comando, por meio do monitoramento da camada HIL (*Host Interface Layer*) que implementa a comunicação entre o SSD e o Sistema Operacional. Dessa forma poderão ser estudadas formas de impedir o envio do comando, retirá-lo da fila de comandos, impedir a sua execução, anular ou amenizar seus efeitos.

Dado que os resultados da execução do comando TRIM se deram de forma diferenciada nos sistemas *Windows* e *Linux*, trabalhos futuros poderão ser direcionados no estudo das diferenças entre as implementações do TRIM em cada sistema, de modo elucidar de que forma a controladora do SSD trata o comando em cada um deles.

Trabalhos futuros também poderão ser voltados ao estudo da granularidade dos arquivos e seus efeitos na recuperação forense, no sistema *Windows* com Sistema de Arquivos **NTFS**, dado que nesse sistema foi possível recuperar, em todos os testes, um arquivo com tamanho na faixa dos *bytes*. Esses estudos poderão levar a uma compreensão mais aprofundada da relação entre tamanho de arquivos, exclusão e recuperabilidade.

Referências

[1] C. J. Antonellis. *Solid State Disks and Computer Forensics*, 2008 ISSA Journal, pgs 36-38. Artigo em formato eletrônico. Disponível em: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.7020&rep=rep1&type=pdf` Última visualização em 23.04.2016. 3

[2] G.B. Bell e R. Boddington. *Solid state drives: The beginning of the end for current practice in digital forensic recovery?*, Journal of Digital Forensics, Security and Law, 5(3):1–20. Australia, 2010. 2, 3, 21, 22, 62

[3] A. Brouwer. *History of BIOS and IDE limits*, 2004. Large Disk HOWTO v2.5. Sítio eletrônico. Disponível em: `http://tldp.org/HOWTO/Large-Disk-HOWTO-4.html` Última visualização em 02.04.2016. 11

[4] D. A. Koufaty C. Feng e X. Zhang. *Understanding intrinsic characteristics and system implications of flash memory based solid state drives*, 2009. In ACM SIGMETRICS Performance Evaluation Review, volume 37, pages 181–192. ACM. 11, 12, 19, 21, 23

[5] B. Carrier. *autopsy - digital forensics platform and graphical interface to the Sleuth Kit® and other digital forensics tools*, Disponível em: `http://www.sleuthkit.org/autopsy/` Última visualização em 25.06.2016. 35, 47

[6] B. Carrier. *The Sleuth Kit - a library and collection of command line tools to investigate disk images*, Disponível em: `http://www.sleuthkit.org/sleuthkit/` Última visualização em 25.06.2016. 35, 54

[7] B. Carrier. *File system forensic analysis*, Upper Saddle River, New Jersey. 2005, Addison-Wesley. 1, 9, 10, 11, 14, 15, 28, 29

[8] N E Case. *extundelete - utility that can recover deleted files from an ext3 or ext4 partition*, Disponível em: `http://extundelete.sourceforge.net/` Última visualização em 25.06.2016. 35, 47

[9] Clker.com. *Green Leaves clip art*, 2016 Sítio eletrônico. Disponível em: `http://www.clker.com/clipart-green-leaves-5.html` Última visualização em 26.07.2016. 45

[10] Microsoft Corporation. *Windows 7 Enhancements for Solid-State Drives*, 2008 Documento em formato eletrônico. Disponível em: `http://download.microsoft.com/download/F/A/7/FA70E919-8F82-4C4E-8D02-97DB3CF79AD5/COR-T558_Shu_Taiwan.pdf` Última visualização em 09.04.2016. 25

[11] Microsoft Corporation. *Solid State Drive Deployment*, 2013 Sítio eletrônico. Disponível em: https://technet.microsoft.com/en-us/library/hh825197.aspx Última visualização em 21.04.2016. 27

[12] Microsoft Corporation. *Basic and Dynamic Disks*, 2016 Sítio eletrônico. Disponível em: https://msdn.microsoft.com/en-us/library/windows/desktop/aa363785%28v=vs.85%29.aspx Última visualização em 26.07.2016. 15

[13] Samsung CT Corporation. *Understanding SSDs: A Peek Behind The Curtain*, 2015 Documento em formato eletrônico. Disponível em: http://www.samsung.com/global/business/semiconductor/minisite/SSD/M2M/download/04_Understanding_SSDs.pdf Última visualização em 09.04.2016. 23

[14] Samsung CT Corporation. *SSD 850 EVO Overview*, 2016. Sítio eletrônico. Disponível em: http://www.samsung.com/global/business/semiconductor/minisite/SSD/global/html/ssd850evo/overview.html Última visualização em 09.04.2016. 19, 41

[15] T. Cunha. *Remoção Segura de Arquivos em EXT3: técnicas para evitar a recuperação de dados*, Brasília: UnB, 2014. 42

[16] ACESS DATA. *FTK Imager - disk imaging program*, Disponível em: http://accessdata.com/product-download/digital-forensics/ftk-imager-version-3.4.2 Última visualização em 25.06.2016. 35, 54

[17] K. Dydecka. *Earth Spinning Animated Gifs*, 2016 Sítio eletrônico. Disponível em: http://bestanimations.com/Earth&Space/Earth/Earth.html Última visualização em 26.07.2016. 45

[18] M. A. C. C. Júnior e R. J. G. B. Queiroz. *Forense em Solid State Drive: entendendo os mecanismos internos que podem inviabilizar a recuperação de dados*, Universidade Federal de Pernambuco (UFPE) 2015. Documento em formato eletrônico. Disponível em: http://sbseg2015.univali.br/anais/WFC/artigoWFC01.pdf Última visualização em 08.04.2016. 20

[19] M. A. C. C. Júnior e R. J. G. B. Queiroz. *Após um processo de formatação, é possível recuperar dados em um SSD?*, Universidade Federal de Pernambuco (UFPE) 2015. Documento em formato eletrônico. Disponível em: http://sbseg2015.univali.br/anais/WFC/artigoWFC03.pdf Última visualização em 23.04.2016. 6

[20] fishhawk. *Corn field*, 2010 Sítio eletrônico. Disponível em: https://www.flickr.com/photos/16502322@N03/4806634131 Última visualização em 26.07.2016. 45

[21] A. Frossi F. Maggi S. Zanero G. Bonetti, M. Viglione. *A Comprehensive Black-box Methodology for Testing the Forensic Characteristics of Solid-state Drives*, 2011 Journal of Computer Virology and Hacking Techniques Artigo em formato eletrônico. Disponível em: http://www.syssec-project.eu/m/page-media/3/ssdforensics-ACSAC-final.pdf Última visualização em 23.04.2016. 4, 6, 36, 63

[22] F. Y. Gebremaryam. *Solid State Drive (SSD) Digital Forensics Construction*, 2011 Master of Science in Computing System Engineering, Politecnico di Milano. Documento em formato eletrônico. Disponível em: https://www.politesi.polimi.it/bitstream/10589/37402/3/SSD%20Digital%20forensics%20Construction.pdf Última visualização em 23.04.2016. 4, 63

[23] Y. Gubanov e O. Afonin. *Recovering Evidence from SSD Drives in 2014: Understanding TRIM, Garbage Collection and Exclusions*, 2014 Sítio eletrônico. Disponível em: https://articles.forensicfocus.com/2014/09/23/recovering-evidence-from-ssd-drives-in-2014-understanding-trim-garbage-collection Última visualização em 20.04.2016. 23, 24, 26, 37

[24] L. Hutchinson. *Latest OS X update allows you to enable TRIM for third-party SSDs*, 2015 Sítio eletrônico. Disponível em: http://arstechnica.com/apple/2015/06/latest-os-x-update-allows-you-to-enable-trim-for-third-party-ssds/ Última visualização em 21.04.2016. 27

[25] G. G. Richard III. *scalpel - an open source data carving tool*, Disponível em: https://github.com/sleuthkit/scalpel Última visualização em 25.06.2016. 35, 47

[26] Corsair Components Inc. *How to check that TRIM is active?*, 2010 Documento em formato eletrônico. Disponível em: http://www.corsair.com/media/cms/manual/How_To_Check_That_TRIM_is_Active.pdf Última visualização em 26.07.2016. 24

[27] K. Kendall e N. Mikus J. Kornblum. *foremost - a console program to recover files based on their headers, footers, and internal data structures*, Disponível em: http://foremost.sourceforge.net/ Última visualização em 25.06.2016. 35, 47

[28] J. Jensen. *magirescue - tool to scan block device for file types it knows how to recover and calls an external program to extract them*, Disponível em: http://www.itu.dk/people/jobr/magicrescue/ Última visualização em 25.06.2016. 35, 47

[29] H. Jones. *Partitioner och filsystem 1*, 2013 Documento em formato eletrônico. Disponível em: http://users.du.se/~hjo/cs/dt1059/presentation/CF1_5_partitions.pdf Última visualização em 26.07.2016. 10

[30] Kbolino. *GUID Partition Table Scheme*, 2007 Sítio eletrônico. Disponível em: https://en.wikipedia.org/wiki/GUID_Partition_Table#/media/File:GUID_Partition_Table_Scheme.svg Última visualização em 26.07.2016. 15

[31] C. King e T. Vidas. *Empirical analysis of solid state disk data retention when used with contemporary operating systems*, 2011 Digital Investigation Journal. Artigo em formato eletrônico. Disponível em: https://www.dfrws.org/2011/proceedings/17-349.pdf Última visualização em 23.04.2016. 4, 5, 63

[32] B. Klug. *Android 4.3 Update Brings TRIM to All Nexus Devices*, 2013 Sítio eletrônico. Disponível em: http://www.anandtech.com/show/7185/android-43-update-brings-trim-to-all-nexus-devices Última visualização em 09.04.2016. 25

[33] F. Knight. *TRIM – DRAT / RZAT clarifications for ATA8 - ACS2*, 2010. Documento em formato eletrônico. Disponível em: `http://www.t13.org/Documents/UploadedDocuments/docs2010/e09158r2-Trim_Clarifications.pdf` Última visualização em 20.04.2016. 24

[34] M-Systems. *Two Technologies Compared: NOR vs. NAND. In White Paper*, 2003. Large Disk HOWTO v2.5. Sítio eletrônico. Disponível em:`http://maltiel-consulting.com/Nonvolatile_Memory_NOR_vs_NAND.pdf` Última visualização em 02.04.2016. 11

[35] T. Nam e T. Kim. *Practical issues in designing of garbage collection for solid states drives.*international journal of future computer and communication , 2(5):451., 2013. Documento em formato eletrônico. Disponível em: `http://www.ijfcc.org/papers/204-S067.pdf` Última visualização em 08.04.2016. 20

[36] H. Nazarian e S. Dubois. *The drive for SSDs: What's holding back NAND flash?*, 2013 Sítio eletrônico. Disponível em: `http://www.edn.com/design/systems-design/4424905/2/The-drive-for-SSDs--What-s-holding-back-NAND-flash--` Última visualização em 26.07.2016. 22

[37] Kernel Newbies. *Linux 2 6 28*, 2008 Sítio eletrônico. Disponível em: `http://kernelnewbies.org/Linux_2_6_28#head-a1a9591f48fe0cf8fde1f0d1f637c7ae54ad0bfa` Última visualização em 09.04.2016. 25

[38] A. Nisbet e S. Lawrence. *A forensic analysis and comparison of solid state drive data retention with trim enabled file systems*, In Proceedings of the 11th Australian Digital Forensics Conference. SRI Security Research Institute, Edith Cowan University, Perth, Western Australia, 2013. 6, 7, 19, 63

[39] K. OBrien. *Seagate Enterprise Performance 10K HDD Savvio 10K.6 Review*, Sítio eletrônico. Disponível em: `http://www.storagereview.com/seagate_enterprise_performance_10k_hdd_savvio_10k6_review` Última visualização em 26.07.2016. 9

[40] A.R. Olson e D.J. Langlois. *Solid State Drives Data Reliability and Lifetime*, 2008. 21

[41] J. O'Reilly. *SSD Prices in A Free Fall*, Sítio eletrônico. Disponível em: `http://www.networkcomputing.com/storage/ssd-prices-free-fall/1147938888` Última visualização em 19.03.2016. 2

[42] Debian Org. *SSDOptimization*, 2016. Sítio eletrônico. Disponível em: `https://wiki.debian.org/SSDOptimization` Última visualização em 20.04.2016. 25, 26

[43] D.A. Patterson e J.L. Henessy. *Computer Organization and Design: The Hardware / Software Interface*, 4a Ed, 2012. Elsevier. 9

[44] KABUM COMERCIO ELETRONICO S/A. *SSD Samsung 850 EVO 2.5 250GB SATA III - MZ-75E250B/AM*, 2016 Sitio eletrônico. Disponivel em: http://www.kabum.com.br/produto/68073/ssd-samsung-850-evo-2-5-250gb-sata-iii-mz-75e250b-am/?tag=SD%20Samsung%20850%20EVO%202.5%B4%20250G. 40

[45] A. Shilov. *Shipment of hard disk drives hit multi-year low in Q1 2015*, Sítio eletrônico. Disponível em: http://www.kitguru.net/components/hard-drives/anton-shilov/shipments-of-hard-disk-drives-hit-multi-year-low-in-q1-2015/ Última visualização em 19.03.2016. 2

[46] E. Slivka. *Mac OS X 10.6.8 Brings TRIM Support for Apple SSDs, Graphics Improvements [Updated]*, 2011 Sítio eletrônico. Disponível em: http://www.macrumors.com/2011/06/27/mac-os-x-10-6-8-brings-trim-support-for-apple-ssds-graphics-improvements/ Última visualização em 09.04.2016. 25

[47] Partition Manager Software. *Understanding Partitions*, 2016 Sítio eletrônico. Disponível em: http://www.partition-magic-manager.com/partition-magic/partition-magic-help/UnderstandingPartitions.php Última visualização em 26.07.2016. 14

[48] Statista. *Global shipments of HDDs and SSDs in PCs from 2012 to 2017 (in millions*, Sítio eletônico. Disponível em: http://www.statista.com/statistics/285474/hdds-and-ssds-in-pcs-global-shipments-2 Última visualização em 09.04.2016. 2

[49] J. Stolfi. *Cryptographic Hash Function*, 2008 Sítio eletrônico. Disponível em: https://commons.wikimedia.org/wiki/File:Hash_function.svg Última visualização em 26.07.2016. 30

[50] L. Torvalds e T. Hao. *libata-core.c - helper library for ATA*, 2016. Código fonte de núcleo de Sistema Operacional. Documento em formato eletrônico. Disponível em: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/drivers/ata/libata-core.c Última visualização em 16.04.2016. 26

[51] Inc. Toshiba Americ Electronic Components. *NAND vs. NOR Flash Memory Technology Overview* , 2006. Disponível em: http://aturing.umcs.maine.edu/~meadow/courses/cos335/Toshiba\20NAND_vs_NOR_Flash_Memory_Technology_Overviewt.pdf Última visualização em 02.04.2016. 11

[52] Ask Ubuntu. *How to enable TRIM?*, 2015. Sítio eletrônico. Disponível em: askubuntu.com/questions/18903/how-to-enable-trim/19480#19480 Última visualização em 20.04.2016. 26

[53] M. West. *A Practical Guide to Learning Linux*, 2004 Sítio eletrônico. Disponível em: http://www.learnlinux.org.za/courses/build/fundamentals/fundamentals-all.html Última visualização em 26.07.2016. 17

[54] F. Liu X. Tao e D. Feng. *Fast Collision Attack on MD5*, 2013. Documento em formato eletrônico. Disponível em: `https://eprint.iacr.org/2013/170.pdf` Última visualização em 16.04.2016. 31