Análise Forense de Documentos Digitais

Prof. Dr. Anderson Rocha

anderson.rocha@ic.unicamp.br

http://www.ic.unicamp.br/~rocha

Reasoning for Complex Data (RECOD) Lab.
Institute of Computing, Unicamp

Av. Albert Einstein, 1251 – Cidade Universitária
CEP 13083-970 • Campinas/SP – Brasil

*** Slides preparados baseados em apresentação de *Tiago Daitx*, MO447 (2010s2)**

File Carving e Smart File Carving

Baseado em "The evolution of file carving – the benefits and problems of forensics recovery." **Anandabrata Pal and Nasir Memon.** IEEE Signal Processing Magazine, 26(2):59–71, March 2009.

Organização

Organização

- Introdução e conceitos gerais
- Recuperação de arquivos
- Smart Carving
- Conclusão
- Referências

Introdução e Conceitos Gerais

O que é file carving?

- É a extração e recuperação de arquivos baseada somente em sua estrutura

Por que File Carving?

A enorme massa de dados no formato digital está sujeita à:

- Corrupção do sistema de arquivos
- Formatação de um dispositivo
- Dados em formato proprietário ou desconhecido
- Arquivos removidos/apagados (intencionalmente ou não)

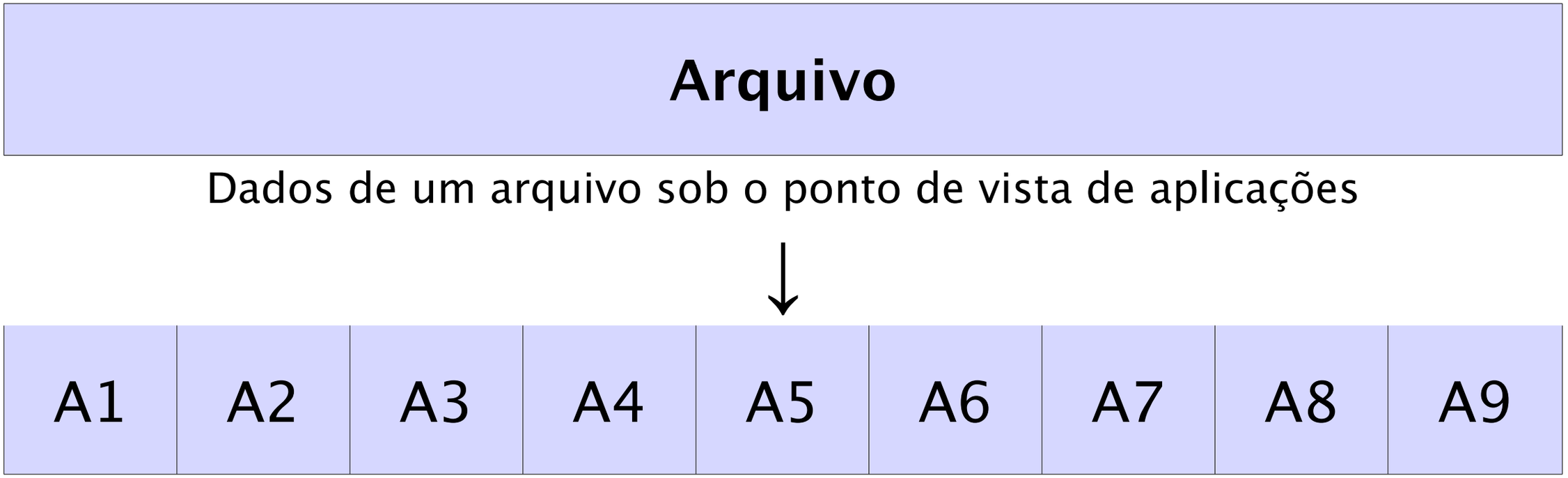
Armazenamento (1)

- Discos Rígidos e SSD's são divididos em Clusters
- Clusters são compostos por Setores e representam a menor quantidade que é possível gravar em uma única vez
- Setores costumam ter 512 ou 4K bytes

Armazenamento (2)

- Os sistemas de arquivos:
  - Gerenciam os arquivos
  - Alocam Blocos (Clusters)
- A alocação de blocos pode ou não ser sequencial
  - Alocação não sequencial indica fragmentação

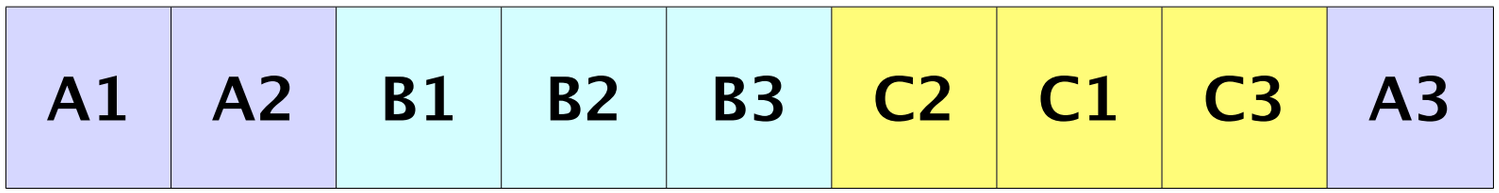
Exemplo: Armazenamento



Fragmentação

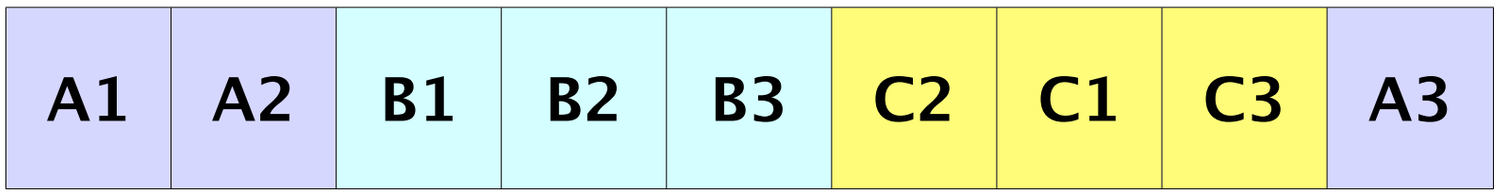
- O nível de fragmentação é dependente do:
  - Sistema de arquivos
  - Tamanho do arquivo
  - Tamanho do bloco
- Alocação não sequencial indica fragmentação
- Os fragmentos podem estar dispostos em qualquer ordem

Exemplo de Fragmentação



No exemplo cada célula representa um bloco. Estão representados 3 arquivos, cada um constituído por 3 fragmentos. Os fragmentos 1, 2 e 3 representam, respectivamente: o início, meio e fim do arquivo.

Nomenclatura



- Utilizando o arquivo A como exemplo:
  - A1 e A2 são o fragmento base
  - A2 é o ponto de fragmentação

Recuperação de arquivos

Recuperação Tradicional

- Dependente de estruturas do sistema de arquivos, ex.: tabelas de arquivos
  - Sistemas de arquivos costumam apenas "marcar" uma entrada como removida
- Permite rápida recuperação dos arquivos enquanto estiverem presentes na estrutura
- Evita buscas em áreas não alocadas do disco

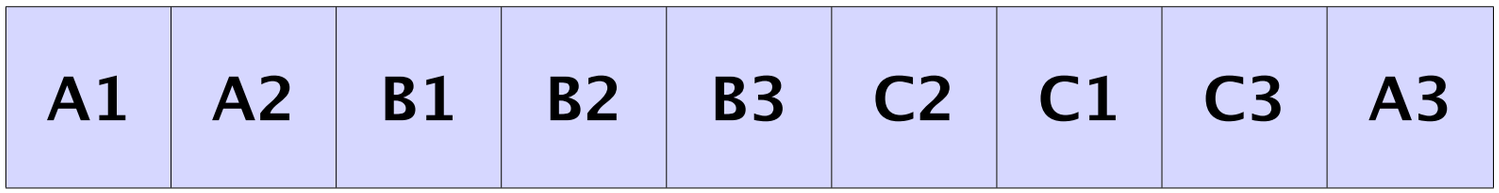
File Carving: Regras gerais

- Não utiliza a estrutura ou informações do sistema de arquivos de forma direta
- Busca em blocos não alocados somente quando existe alguma confiança na informação do sistema de arquivos
- Costumam identificar arquivos muito comums através de hashes (MD5) e palavras chave

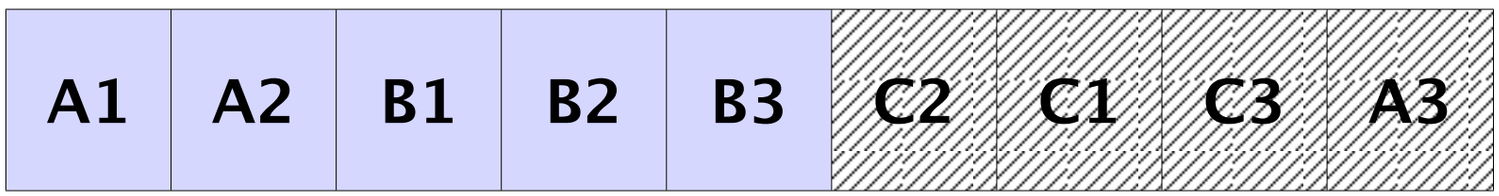
File Carving: Recuperação baseada em estrutura

- Busca pelos “numeros mágicos” (ie. sequência de bytes em pontos conhecidos)
  - Cabeçalho e rodapé, ou
  - Cabeçalho + tamanho do arquivo
- Técnicas mais avançadas também utilizam o conteúdo do arquivo
- O arquivo é formado por todos os blocos que se encontram entre o setor do cabeçalho e o setor do final

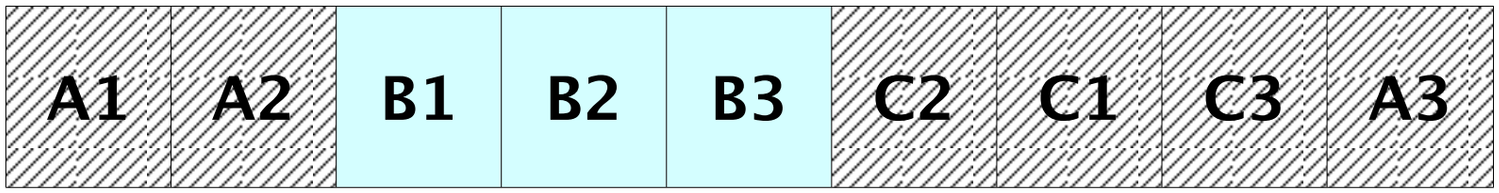
Exemplo: Recuperação baseada em estrutura



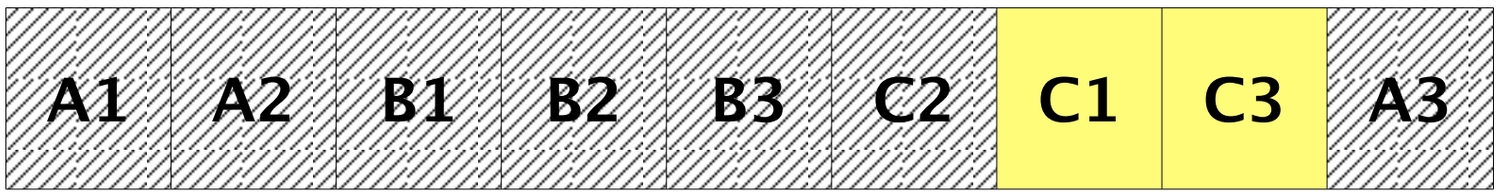
Arquivo A: A1+A2+B1+B2+B3+C2+C1+C3+A3



Arquivo A: A1+A2+B1+B2+B3



Arquivo B: B1+B2+B3



Arquivo C: C1+C3

File carving: Recuperação baseada em grafos

- Tenta resolver o problema de recuperação baseada em estrutura para arquivos fragmentados
- Os blocos representam os vértices
- Arestas representam a verossimilhança entre blocos através de um peso
- Como definir a verossimilhança?

Grafos: Caminho Hamiltoniano (1/2)

- Técnica apresentada por Shanmugasundaram et al.
- Computa a permutação de um conjunto de **n** blocos pertencentes ao arquivo A que representa a estrutura original de A
  - Os pesos entre dois blocos representa a probabilidade de serem adjacentes
  - A permutação correta é provavelmente a que maximiza da soma dos pesos
- O conjunto de todos os pesos forma uma matriz de adjacência de um grafo completo de **n** vértices.
  - A sequência correta é um caminho no Grafo.

Grafos: Caminho Hamiltoniano (2/2)

- Como determinar o peso entre dois blocos?
  - Prediction by parcial matching (PPM) para textos (Kulesh et al.)
  - Comparação de bordas entre blocos para imagens (Pal et al.)

Grafos: k-Vertex Disjoint Path (1/2)

- Refinamento do método do caminho Hamiltoniano por Pal et al.
  - Em casos reais, múltiplos arquivos estão fragmentados juntos
  - Utiliza a estatística destes múltiplos arquivos
- Cada vértice representa um bloco

Grafos: k-Vertex Disjoint Path (2/2)

- **k** arquivos identificados pelo cabeçalho
  - Existem apenas **k** caminhos disjuntos, pois (usualmente) cada bloco pertence unicamente à um arquivo
- É um problema NP difícil
- Foram propostos algoritmos de caminho único (UP – unique path)

Algoritmos Unique Path (UP)

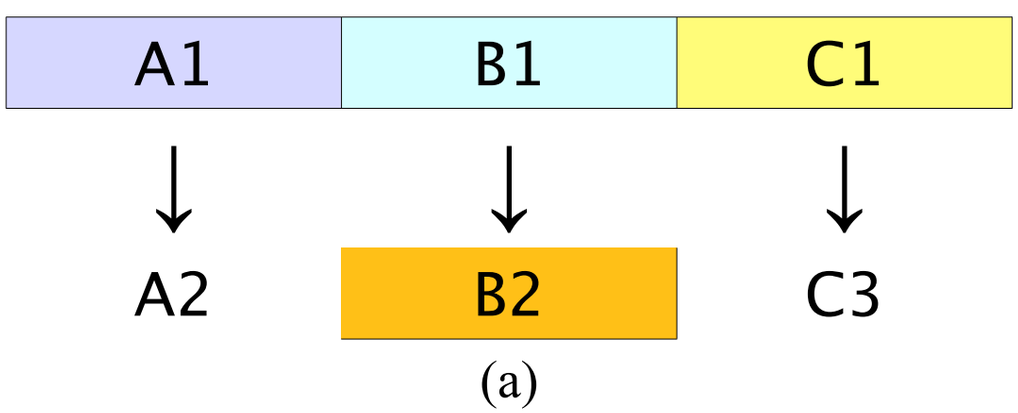
- É realista: cada bloco costuma pertencer a um único arquivo
- Erros se propagam em cascata
  - Um bloco incorreto leva à reconstrução errada de 2 arquivos

File Carving: PUP

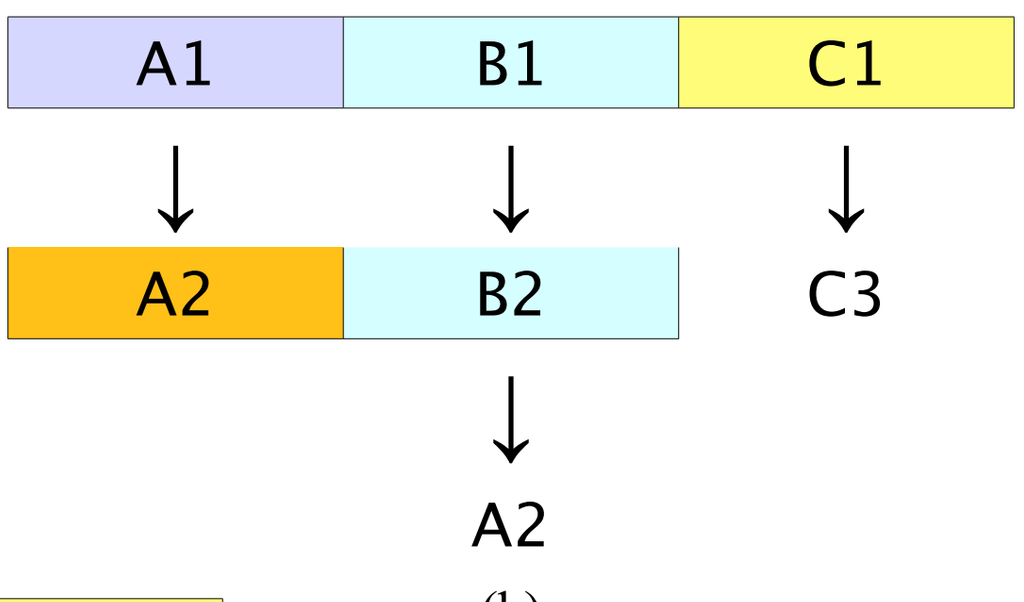
PUP: Parallel Unique Path

1. Começa com um conjunto **S** com **k** Headers (s_1,s_2,...,s_k), referentes a **k** arquivos.
2. Encontra um conjunto **T** com **k** setores, onde t_i é a melhor correspondência ao s_i. Seleciona o t_i com correspondência mais alta dentre todos.
    i. Adiciona o t_i ao caminho do i-ésimo arquivo
    ii. Substitui o setor corrente em **S** para o i-ésimo elemento ($s_i = t_i$)

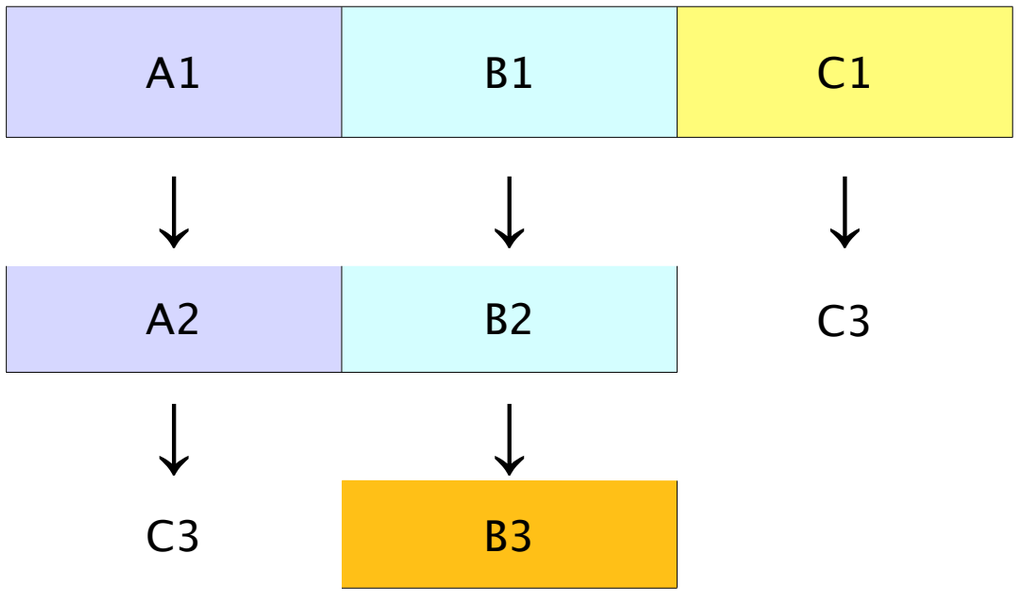
Exemplo: PUP



(a)



(b)



(c)

File Carving: SPF (1/2)

SPF: Shortest Path First

1. Para cada uma das imagens a serem reconstruídas:
    i. A partir do conjunto de todos os blocos disponíveis, reconstrói o caminho da imagem e calcula o custo médio do caminho
2. Encontra dentre todos os caminhos aquele com o menor custo e reconstrói essa imagem
3. Remove dos outros caminhos os blocos utilizados na reconstrução do passo (2)
4. Repete o passo (1) até que todas as imagens sejam reconstruídas

File Carving: SPF (2/2)

- Reconstrução de até 88% dos arquivos contra 83% do PUP
- Tem performance e escalabilidade muito mais baixas que o PUP
- Os pesos das arestas são pré-computados para facilitar a busca, porém este passo tem complexidade $O(n^2 \log n)$

Smart Carving

Proposto por *Pal et al.*

- Visa resolver problemas de escalabilidade
- Leva em consideração o comportamento típico da fragmentação em discos
- Etapas:
  - Pré processamento
  - Comparação/classificação
  - Reconstrução

Smart Carving: Pré-processamento

- Aplicada em dados com compressão ou encriptação
- Opcionalmente pode remover os blocos reconhecidamente alocados

Smart Carving: Comparação e Classificação

- Classifica os blocos pelo tipo de arquivo
  - Palavras chave ou padrões
  - Frequencia de caracteres ASCII
  - Entropia
  - “Impressão digital” de arquivos

Smart Carving: "Impressão digital" (1/3)

- *McDaniel and Heydari* propuseram 3 algoritmos:
  - Frequencia de distribuição de bytes (BFD): média dos histogramas de diversos exemplos e correlação dos bytes
  - Frequencia de distribuição de correlação cruzada (BDC): correlação entre os bytes
  - Inclusão de cabeçalho e rodapé
- Baixa acurácia: 30% (BFD), 45% (BFC) e 95%
- Não funciona bem para blocos

Smart Carving: "Impressão digital" (2/3)

Proposto por *Wang and Stalfo*

- Utiliza um conjunto de modelos BFD e desvio padrão
- Melhor acurácia: entre 75% e 100%
- Acurácia decresce com o número de bytes disponíveis

Smart Carving: "Impressão digital" (3/3)

Karresand et al. propuseram o método Oscar

- Utiliza um modelo de centróide baseado na média e desvio padrão de cada byte
  - Obteve uma acurácia de 97%
- Melhorado ao incorporar uma medida do ordenamento dos bytes utilizando a diferença absoluta entre bytes adjacentes
  - Acurácia de 99% para JPEG

Smart Carving: Reconstrução

- Visa localizar o ponto de fragmentação de um arquivo
- Garfinkel mostrou que em geral os arquivos raramente se fragmentam em mais de 3 fragmentos
- A reconstrução consistem em encontrar o fragmento-base e localizar seu ultimo bloco

Smart Carving: SHT-PUP (1/2)

- Modificação do PUP por *Pal et al.*
- SHT: Sequential Hypotesys Testing
- Cada tipo de arquivo possui uma hipótese
- Blocos são anexados ao caminho até que a Hipótese seja confirmada ou refutada
- Só foi implementado para imagens JPEG

1. Começa com um conjunto **S** com **k** Headers (s_1,s_2,...,s_k), referentes a **k** arquivos.

2. Encontra um conjunto **T** com **k** setores, onde t_i é a melhor correspondência ao s_i. Seleciona o t_i com correspondência mais alta dentre todos.

   i. Adiciona o t_i ao caminho do i-ésimo arquivo

   ii. Substitui o setor corrente em **S** para o i-ésimo elemento ($s_i = t_i$)

   iii. Analiza sequencialmente os setores imediatamente depois de t_i até detectar um ponto de fragmentacao t_f ou o arquivo estar completo. Consiste de um teste de hipótese que é executado para cada setor subsequente adicionado ao fragmento. O teste verifica se a sequencia completa pertence ou não ao fragmento inteiro. Se o resultado for inconclusivo o próximo setor é adicionado.

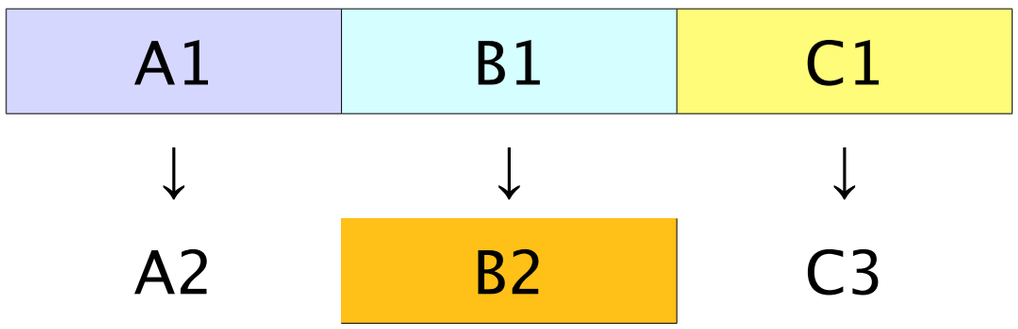
   iv. Substitui o setor corrente em **S** com t_f ($s_i = t_i$)

   v. Encontra um novo conjunto **T** das melhores correspondencias

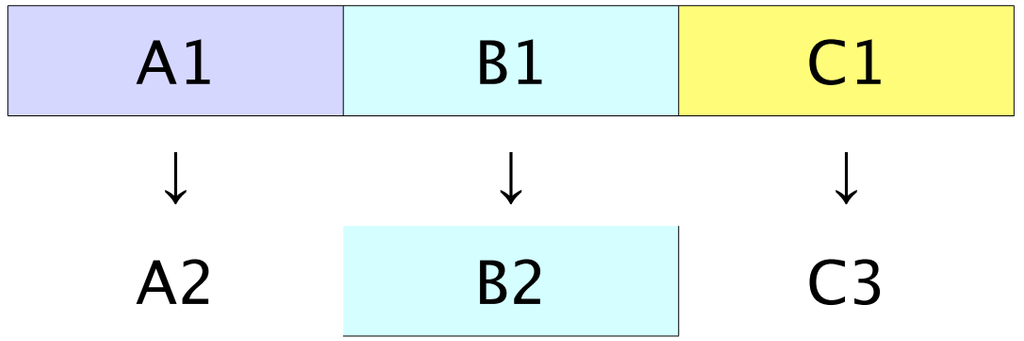
   vi. Seleciona o elemento t_i com a melhor correspondencia

   vii. Repete passo (i) até todos os arquivos estarem completos.

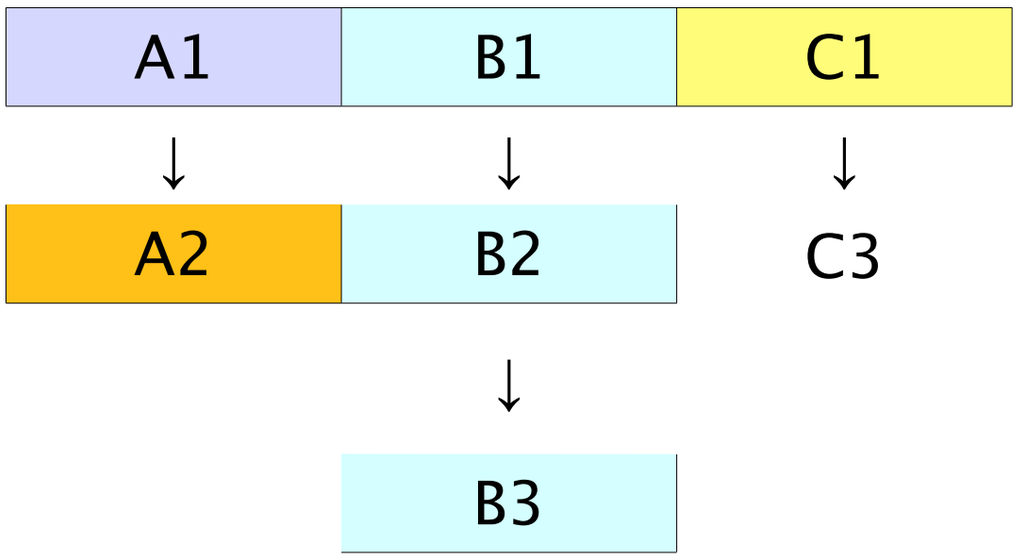
Exemplo: SHT-PUP



(a)



(b)



(c)

Conclusões

Conclusões

Foi feita uma análise abrangente das técnicas existentes para a recuperação de arquivos sem o uso de nenhuma meta-informação do sistema de arquivos.

Uma técnica de recuperação de arquivos de texto e imagens foi apresentada. Esta técnica, apesar de útil, precisa ser ampliada para incorporar recuperação de arquivos de vídeo, áudio, executáveis e outros formatos.

Referências

Referências

1. Anandabrata Pal, Husrev T. Sencar, and Nasir Memon. Detecting file fragmentation point using sequential hypothesis testing. Digital Investigation (DIIN), 5(1):S2–S13, September 2008.
2. Anandabrata Pal and Nasir Memon. The evolution of file carving – the benefits and problems of forensics recovery. IEEE Signal Processing Magazine, 26(2):59–71, March 2009.
3. Pal A, Memon N. Automated reassembly of file fragmented images using greedy algorithms. IEEE Transactions on Image processing February 2006:385–93.
4. K. Shanmugasundaram and N. Memon, "Automatic reassembly of document fragments via data compression," presented at the 2nd Digital Forensics Research Workshop, Syracuse, NY, July 2002
5. A. Pal, K. Shanmugasundaram, and N. Memon, "Reassembling image fragments," in Proc. ICASSP, Hong Kong, Apr. 2003, vol. 4, pp. IV–732-5.
6. A. Pal and N. Memon, "Automated reassembly of file fragmented images using greedy algorithms," IEEE Trans. Image Processing, vol. 15, no. 2, pp.385 – 393, Feb. 2006.
7. A. Pal, T. Sencar, and N. Memon, "Detecting file fragmentation point using sequential hypothesis testing," Digit. Investig., to be published.
8. M. McDaniel and M. Heydari, "Content based file type detection algorithms," in Proc. 36th Annu. Hawaii Int. Conf. System Sciences (HICSS'03)—Track 9, IEEE Computer Society, Washington, D.C., 2003, p. 332.1
9. K. Wang, S. Stolfo, "Anomalous payload-based network intrusion detection," in Recent Advances in Intrusion Detection, (Lecture Notes in Computer Science), vol. 3224. New York: Springer-Verlag, 2004, pp. 203 –222.
10. M. Karresand and N. Shahmehri, "Oscar file type identification of binary data in disk clusters and RAM pages," in Proc . IFIP Security and Privacy in Dynamic Environments, vol. 201, 2006, pp. 413 – 424.
11. M. Karresand and N. Shahmehri, "File type identification of data fragments by their binary structure," in Proc. IEEE Information Assurance Workshop, June 2006, pp. 140 –147.

NOTA: os artigos de A. Pal et al. podem ser obtidos em http://digital-assembly.com/technology/

Obrigado!