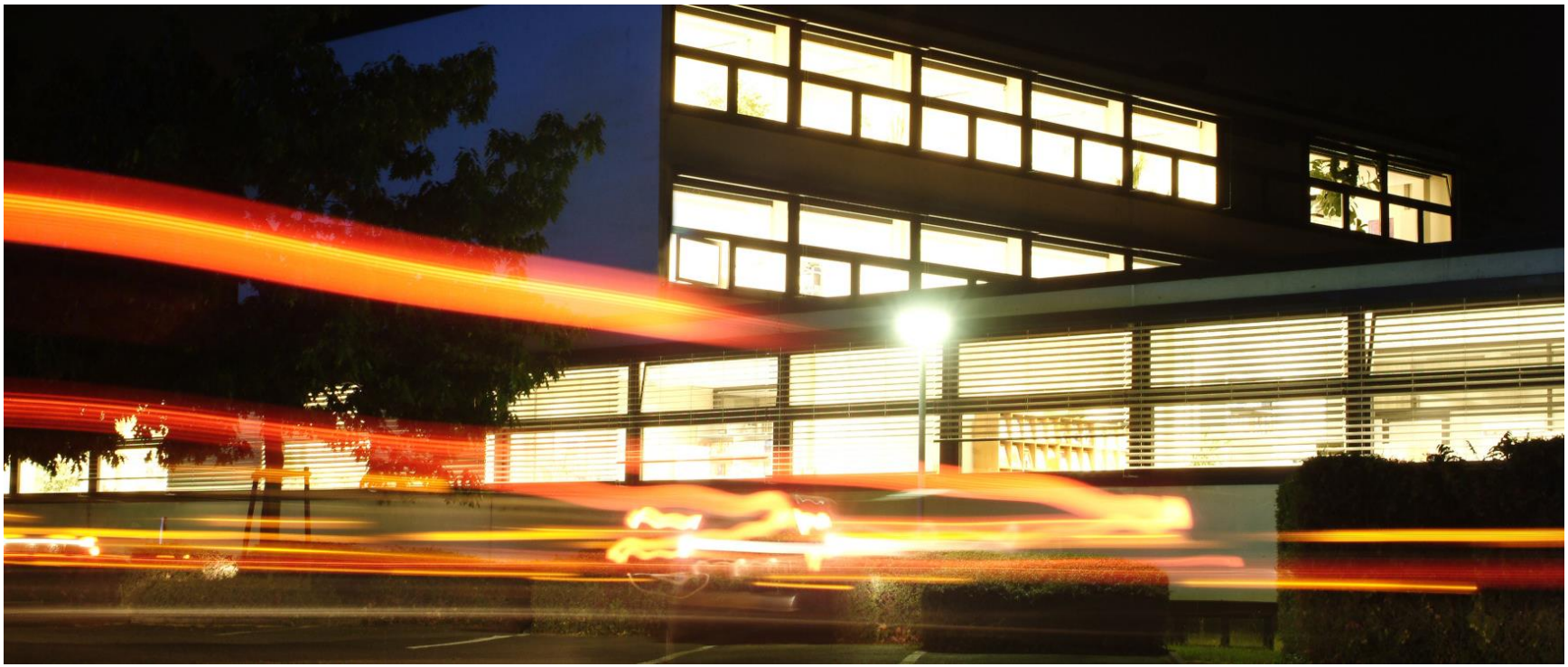
Effiziente Automatisierung von IT-forensischen Standardverfahren

York Yannikos
Media Security and IT Forensics, Fraunhofer SIT



CASED

Fraunhofer SIT

Motivation

© Thierry Gregorius

"Your recent Amazon purchases, Tweet score and location history makes you 23.5% welcome here."

Stark ansteigendes Datenvolumen führt schnell zur Überlastung

- Speicherbedarf bei Datensicherung
- Zeitaufwand durch Notwendigkeit verschiedener Analysen
- Häufig nur strikt sequentielle Ausführung von Arbeitsschritten
- Keine Vermeidung redundanter Arbeitsschritte
- Fehlerraten einzelner Analysemethoden
- Manuelle Nachbearbeitung fast immer notwendig

➔ Bereits einfache Optimierungskonzepte können Aufwände stark verringern

Motivation

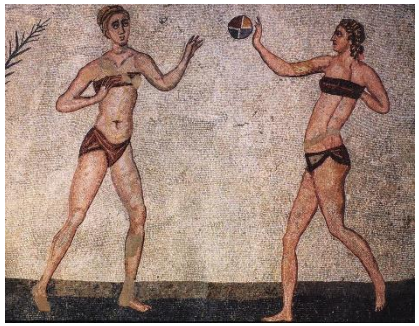
Typische Schritte einer IT-forensischen Untersuchung

- Erstellen einer forensisch exakten Kopie der Originaldaten
- Rekonstruktion gelöschter Dateien (soweit möglich)
- Filterung bekannter Dateien mittels Black-/Whitelists
- Kategorisierung relevanter Daten (Bilder, Videos, …)
- Spezifische Analysen
  - Zeitlinienanalyse
  - Robuste Erkennung bekannter Multimediadateien
  - Ähnlichkeitssuche
  - …
- Manuelle Sichtung vom „Rest“

Motivation

Wesentliche Probleme

- Ineffizienz
  - Mehrfache Verarbeitung derselben Daten
  - Schlechte Nutzung verfügbarer Hardware-Ressourcen
  - Nicht einheitliches Format für Ergebnisdaten
    - ➔ Oftmals Nachbearbeitung notwendig
- Ineffektivität
  - Bspw. Filterung *ausschließlich* mittels kryptographischer Hashes
    - ➔ False Negatives sind sehr einfach zu erzeugen
  - Eingesetzte Analysemethoden teilweise nutzlos



e1eb417354cc79d5cf9fcdbcb005ee5e

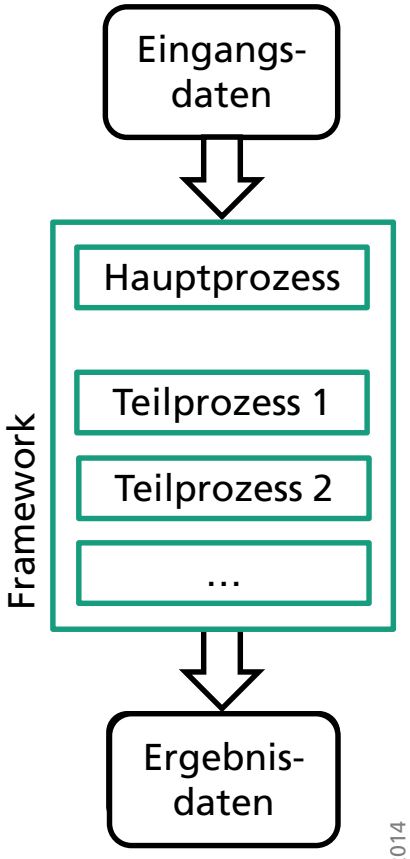
a01e1f145b9f0dcee4b3d894b9612256

Mosaik aus der
Villa Romana del Casale, Sizilien,
ca. 4. Jahrhundert n. Chr.

Optimierungskonzept

Vorschläge zur Effizienzverbesserung

- Sinnvolle Kombination existierender Lösungen
- Einfaches Optimierungskonzept
  - Automatisierung von Teilprozessen
  - Parallelisierung von Teilprozessen
  - Effektive Filterung bekannter Dateien
- Fokus auf Entwicklung eines Frameworks



Automatisierung von Teilprozessen

- Erfordert spezielle Konfiguration einzelner Teilprozesse
- Einheitliche Schnittstellen zur Teilprozesssteuerung/-kommunikation
- Einheitliches Format für Ergebnisdaten

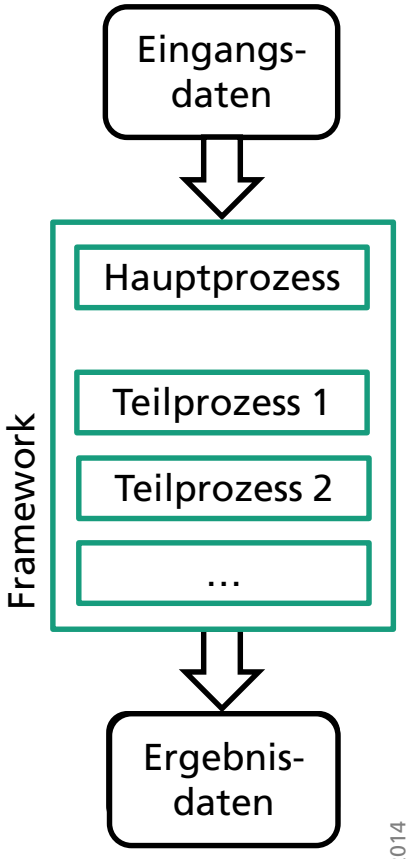
Optimierungskonzept

Parallelisierung von Teilprozessen

- Vermeidung sequentieller Ausführung redundanter Teilschritte
- Identifikation von Teilprozessen mit identischer Datenbasis
- Ausnutzung geeigneter Hard- und Software, bspw. Multithreading, effiziente Bibliotheken, …

Effektive Filterung bekannter Daten

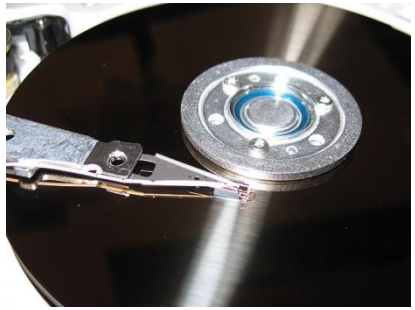
- Reduzierung des Datenvolumens
- Priorisierung von Daten für manuelle Nachbearbeitung
- Anwendung von Verfahren zur Ähnlichkeitserkennung
- Erweiterung der bekannten Datenbasis durch Verwendung zusätzlicher Quellen



Parallelisierung

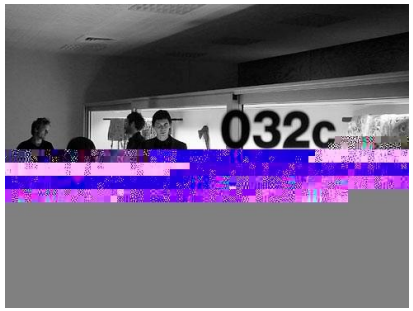
Beispielszenario

1. Erstellen einer forensisch exakten Kopie eines Datenträgers
2. Berechnen der Prüfsummen (kryptographische Hashes)
3. Rekonstruktion gelöschter Dateien



© Inklein, Wikipedia

```
8d858f56b81d22294bed3f8efc0fefcc
ea488f2feabe060965be117f664b49f15cba3431
...
```



Problem

- Datenträger bzw. sein Abbild wird jeweils erneut vollständig eingelesen (worst case)

Lösungsansatz

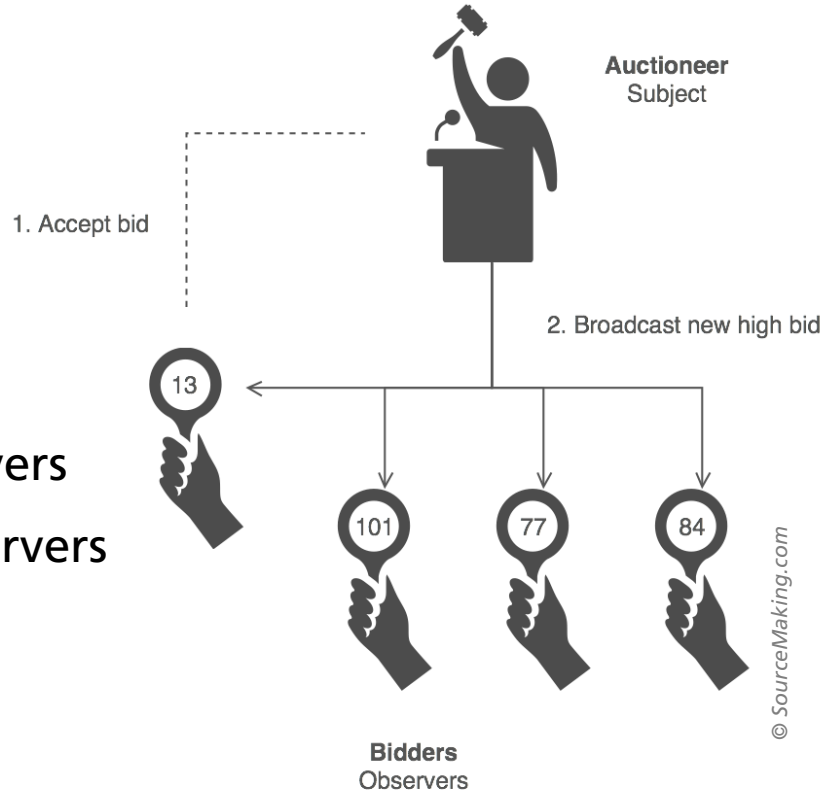
- Observer Pattern für Einleseprozess verwenden

➔ Einmaliges Einlesen, mehrfache parallele Weiterverarbeitung

Parallelisierung

Verwendung des Observer Patterns

- Software Design Pattern
- Dient zur Kommunikation und Verarbeitung von Ereignissen
- *Subject*
  - Pflegt Liste von registrierten Observers
  - Benachrichtigt alle bekannten Observers über Zustandsänderungen
- *Observers*
  - Registrieren sich beim Subject, um Änderungsbenachrichtigungen zu erhalten
  - Reagieren auf Änderungsbenachrichtigung vom Subject



Parallelisierung im Framework

Subject

- Modul zum Einlesen der Eingangsdaten (bspw. Festplatten, optische Datenträger), benachrichtigt Observers über Änderungen im Lese-Buffer

Observer

- Module zur Verarbeitung der Eingangsdaten (bspw. File Carving, Hashing)
- Werden über Änderungen des Buffer-Inhalts vom Subject benachrichtigt

Prozess für Beispielszenario

1. Subject liest Chunk der Eingangsdaten in den Buffer, benachrichtigt alle Observers, dass neue Daten im Buffer liegen, wartet auf Rückmeldung
2. Observers reagieren auf Benachrichtigung, verarbeiten parallel Daten-Chunk, melden an Subject zurück, wenn fertig
3. Schritte 1 und 2 werden wiederholt, bis Eingangsdaten vollständig gelesen

Effektive Filterung

Beispielszenario

- Dateien mit unbekanntem Inhalt auf Datenträger
- Black-/Whitelisting soll mittels vorliegender Hashlisten durchgeführt werden

Problem

- Garantierte Unvollständigkeit von Hashlisten
- Extrem geringe Robustheit kryptographischer Hashes

➔ Hoher manueller Nachbearbeitungsaufwand

Lösungsansatz

- Suche nach Hashes im Web und in P2P-Netzen
- Robust Hashing für Multimediadaten
- Fuzzy Hashing für beliebige Daten

Effektive Filterung durch Hash-Suche im Internet

Framework zur Suche von (kryptographischen) Hashes

- P2P-Netzwerke und Web-Suchmaschinen als Quellen
- Zusätzlich Verwendung lokaler Hashlisten
- Anreicherung unbekannter Dateien mit gefundenen Metadaten

Sample file	Source	Total number of results	Most common file names found
Picture	Google	1100	`Lighthouse.jpg` `8969288f4245120e7c3870287cce0ff3.jpg`
	Yahoo	7	`Lighthouse.jpg`
	eDonkey	7	`Lighthouse.jpg` `sfondo hd (3).jpg`
	ISC	1	`Lighthouse.jpg`

Audio file	eDonkey	31	`Madonna feat. Nicki Minaj M.I.A. Give(...).mp3`
			`Madonna - Give me all your lovin [ft.(...).mp3`
			`Madonna ft. Nicki Minaj & M.I.A. - G(...).mp3`

Video file[1]	eDonkey	130	`La.Loca.De.la.Luna.Llena.Spanish.XXX.(...).mpg`
			`La Loca De La Luna Llena (Cine Porno (...).avi`
			`MARRANILLAS TORBE...avi`
			`Pelis porno - La Loca De La Luna Llen(...).mpg`

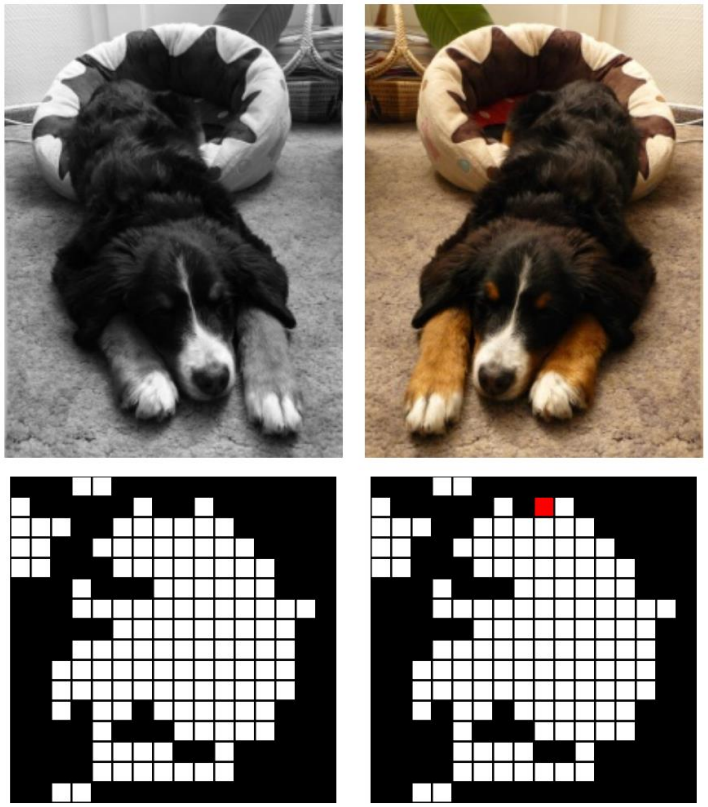
[1] local file name: `Pulp Fiction 1994.avi`

Beispielergebnisse

Effektive Filterung durch Robust Hashing

Fehlertolerante Erkennung ähnlicher Multimediadaten

- Hash basierend auf menschlicher Wahrnehmung
- Daten werden als *ähnlich* wahrgenommen → ähnlicher Hash
- Großer Unterschied zu kryptographischen Hashes
- Sehr niedrige Fehlerraten, performant
- Lösungen existieren für Bilder, Videos, Audiodateien

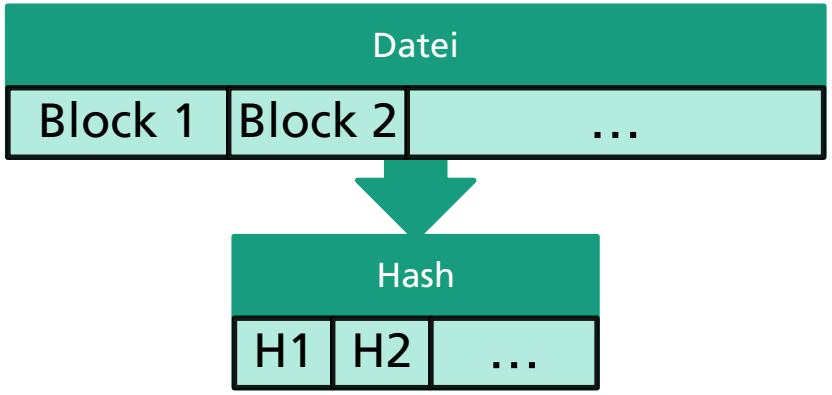


Darstellung der robusten Hashes zweier ähnlicher Bilder

Effektive Filterung durch Fuzzy Hashing

Erkennung ähnlicher Dateiinhalte

- Ähnliche Inhalte → ähnliche Hashwerte
- (Context Triggered) Piecewise Hashing
  - Unterteilen einer Datei in Segmente
  - Erkennen der Segmente
- Einsatzgebiete
  - Software-Erkennung über Versionsgrenzen hinweg
  - Erkennen von Textpassagen
- Ähnlichkeitssuche aufwendig
  - Effizienter Algorithmus zum Ähnlichkeitsabgleich



Praktische Umsetzung

Kopierstation *Delta* (LSK Data Systems GmbH)

© LSK Data Systems GmbH

- Leseroboter für große Mengen optischer Datenträger
- Datenträger werden direkt beim Einlesen auf enthaltene Bilder durchsucht
- Sicheres Protokoll gefundener Daten wird erzeugt
- Von gefundenen Bildern wird robuster Hash erzeugt
- Robuster Hash wird mit Datenbank abgeglichen
- Vorteile gegenüber manueller/sequentieller Durchführung
  - Gesamtprozess vollständig automatisiert
  - Kopierzeit nur marginal durch parallele Analyseschritte beeinflusst
  - Direkt nach Sicherung der optischen Datenträger liegt Ergebnisprotokoll zu ggf. erkanntem Bildmaterial vor

Praktische Umsetzung

File Carving & Content Identification Framework

- Framework zur IT-forensischen Sicherung und Erstanalyse
- Modularer Aufbau, leicht erweiterbar
- Module für individuelle Verarbeitungsschritte
  - Datensicherung
  - Sichere Protokollierung
  - File Carving
  - Berechnung kryptographischer Hashes
  - Robust Hashing
- Ausnutzung verfügbarer Hardware-Ressourcen

➔ Aktuell in Entwicklung

➔ Bereits existierende Module auch als Einzelprodukt verfügbar (bspw. Robust Hashing als Plugin für X-Ways Forensics)

Zusammenfassung

Automatisierte effiziente Sicherung und Erstanalyse gut realisierbar

- Anwendung effektiver Analyseverfahren wichtig
  - Aktuell in der Praxis eingesetzte Analyseverfahren liefern teilweise nur schlechte Ergebnisse
  - Verbesserte Verfahren existieren, sind praxistauglich
- Verfügbare Hardware-Ressourcen möglichst ausreizen
  - Parallelisierung geeigneter Analyseschritte
  - Ausnutzung von Multithreading
  - ggf. Verteilung auf mehrere Systeme
- Redundante Verarbeitungsschritte vermeiden
  - Geeignetes Software-Design

➔ Kostenintensive Allround-Lösungen liefern nicht zwangsläufig bessere (oder gleichwertige) Ergebnisse

Vielen Dank für Ihre Aufmerksamkeit.

York Yannikos

Fraunhofer SIT, Media Security and IT Forensics

Rheinstr. 75, 64295 Darmstadt

Tel.: +49 (0)6151 869-308

york.yannikos@sit.fraunhofer.de

www.sit.fraunhofer.de