DFRWS

Digital Forensic Doctoral Symposium

Proceedings of the inaugural Digital Forensics Doctoral Symposium (DFDS 2025)

held as part of the 12th Digital Forensics Research Conference Europe (DFRWS EU)

Brno, Czech Republic, 1st April 2025

In Cooperation with

DFRWS
DIGITAL FORENSICS RESEARCH CONFERENCE

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

The Association for Computing Machinery
1601 Broadway, 10th Floor
New York, New York 10019, USA

ACM ISBN: 979-8-4007-1076-6

Welcome to the Digital Forensics Doctoral Symposium 2025!

We are delighted to introduce the inaugural Digital Forensics Doctoral Symposium (DFDS) 2025, a dedicated forum for early-career researchers in digital forensics. DFDS is designed for current PhD students, master's students potentially considering a PhD, and early-stage researchers eager to share their work, refine their ideas, and engage with the wider academic and professional community. Participants may present doctoral research, research conducted during their master's theses, or other early-stage research projects that contribute to the advancement of digital forensics.

DFDS 2025 takes place on April 1, 2025, in Brno, Czech Republic and is co-located with the DFRWS EU conference. This symposium is designed to foster collaboration, mentorship, and academic growth, helping doctoral students refine their research and establish connections within the broader digital forensics community.

Our Mission

The Digital Forensics Doctoral Symposium aims to:

- Support the next generation of digital forensics scholars by providing a structured environment for academic exchange,
- Facilitate interdisciplinary collaboration and bridge the gap between research and practical application,
- Encourage innovative thinking and scientific rigor in tackling contemporary and emerging challenges in digital forensics,
- Strengthen the digital forensics research community by building lasting professional networks among young scholars and established experts.

In response to this year's call for papers, we received 12 research paper submissions from researchers across various institutions. Each submission underwent a rigorous peer-review process, with at least three reviewers evaluating the work based on originality, scientific contribution, and relevance to the digital forensics community. Following careful discussions by the Technical Program Chairs, 7 papers were accepted for presentation at DFDS 2025. Additionally, 3 articles originally submitted to the main DFRWS EU track were moved to DFDS, enriching the symposium with further high-quality research. This results in a total of 10 presentations, highlighting the symposium's role in fostering early-stage research and academic exchange in digital forensics.

We extend our sincere appreciation to the conference organisers from DFRWS EU and Brno University of Technology, reviewers, and participants who have contributed to making DFDS 2025 a reality. We look forward to an insightful, engaging, and inspiring symposium that will help shape the future of digital forensics research!

Organising Committee

Symposium Chair	Aikaterini Kanta, Ph.D. (University of Portsmouth)
Symposium Co-Chair	Frank Breitinger, Ph.D (University of Augsburg)
Program Chair	Mark Scanlon, Ph.D. (University College Dublin)

Technical Program Committee

- Frank Adelstein, Ph.D. (NFA Digital, LLC)
- Stefan Axelsson, Ph.D. (Stockholm University)
- Edita Bajramovic, Ph.D. (Siemens Energy)
- Frank Breitinger, Ph.D. (University of Augsburg)
- Femi Fasunlade Ph.D. (University of Portsmouth)
- Virginia Franqueira, Ph.D. (University of Kent)
- Milan Čermák, Ph.D. (Masaryk University)
- Felix Freiling, Ph.D. (Friedrich-Alexander-Universität Erlangen-Nürnberg)
- Jan Gruber, Ph.D. (Friedrich-Alexander-Universität Erlangen-Nürnberg)
- Christopher Hargreaves, Ph.D. (University of Oxford)
- Martin Lambertz (Fraunhofer FKIE)
- Sean McKeown, Ph.D. (Edinburgh Napier University)
- Jan-Niclas Hilgert (Fraunhofer FKIE)
- Aikaterini Kanta, Ph.D. (University of Portsmouth)
- Jenny Ottmann, Ph.D. (Friedrich-Alexander-Universität Erlangen-Nürnberg)
- Jan Pluskal, Ph.D. (Brno University of Technology)
- Ricardo J. Rodríguez, Ph.D. (Universidad de Zaragoza)
- Jens-Petter Sandvik, Ph.D. (Norwegian University of Science and Technology)
- Asanka Sayakkara, Ph.D. (University of Colombo School of Computing)
- Mark Scanlon, Ph.D. (University College Dublin)
- Janine Schneider, Ph.D. (Friedrich-Alexander-Universität Erlangen-Nürnberg)
- John Sheppard, Ph.D. (South East Technological University)
- Hannes Spichiger, Ph.D. (University of Applied Sciences Luzern)
- Harm van Beek, Ph.D. (Netherlands Forensic Institute)
- Wietse Venema, Ph.D. (Google)

Table of Contents

AutoDFBench: A Framework for AI Generated Digital Forensic Code and Tool Testing and Evaluation

Akila Wickramasekara
School of Computer Science
University College Dublin
Dublin, Ireland
akila.wickramasekara@ucdconnect.ie

Alanna Densmore
Florida State University
Tallahassee, FL, USA
amd22c@fsu.edu

Frank Breitinger
Institute of Computer Science
University of Augsburg
Augsburg, Germany
frank.breitinger@uni-a.de

Hudan Studiawan
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
hudan@if.its.ac.id

Mark Scanlon
School of Computer Science
University College Dublin
Dublin, Ireland
mark.scanlon@ucd.ie

Abstract

Generative AI (GenAI) and Large Language Models (LLMs) show great potential in various domains, including digital forensics. A notable use case of these technologies is automatic code generation, which can reasonably be expected to include digital forensic applications in the not-too-distant future. As with any digital forensic tool, these systems must undergo extensive testing and validation. However, manually evaluating outputs, including generated DF code, remains a challenge. AutoDFBench is an automated framework designed to address this by validating AI-generated code and tools against NIST's Computer Forensics Tool Testing Program (CFTT) procedures and subsequently calculating an AutoDFBench benchmarking score. The framework operates in four phases: data preparation, API handling, code execution, and result recording with score calculation. It benchmarks generative AI systems, such as LLMs and automated code generation agents, for DF applications. This benchmark can support iterative development or serve as a comparison metric between GenAI DF systems. As a proof of concept, NIST's forensic string search tests were used, involving more than 24,200 tests with five top-performing code generation LLMs. These tests validated the output of 121 cases, considering two levels of user expertise, two programming languages, and ten iterations per case with varying prompts. The results also highlight the significant limitations of the DF-specific solutions generated by generic LLMs.

CCS Concepts

• **Applied computing** → **Computer forensics**; *Evidence collection, storage and analysis*; • **Software and its engineering** → Software verification and validation.

DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712718

Keywords

Digital Forensics, Large Language Models, Investigative Process, Automation, Challenges

ACM Reference Format:
Akila Wickramasekara, Alanna Densmore, Frank Breitinger, Hudan Studiawan, and Mark Scanlon. 2025. AutoDFBench: A Framework for AI Generated Digital Forensic Code and Tool Testing and Evaluation. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712716.3712718

1 Introduction

Digital forensics (DF) relies on software tools to gather and analyse digital data, which can range from small single-function scripts to advanced software suites [2, 18]. With recent advances in generative artificial intelligence (GenAI), including Large Language Models (LLMs) [12], and the growing interest in the application of AI to digital forensics [4], it is reasonable to expect GenAI to be used for digital forensic purposes in the not-too-distant future. This has the potential to streamline workflows and allow practitioners to develop bespoke solutions faster for specific DF tasks [16]. However, this evolution raises critical questions about the evaluation of these tools, i.e., scalable and robust testing.

To address these challenges, this article introduces a novel framework, AutoDFBench (Automatic Digital Forensic Code and Tool Benchmarking), along with its corresponding AutoDFBench score. The framework provides a structured approach to evaluate AI-generated DF code, comparable to unit testing in software development, where specific functions are validated against expected results. A forensic task is defined with a dataset containing a 'ground truth'. The task is executed through a pipeline, enabling comparison of the tool's output with the ground truth. This methodology is exemplified in the National Institute of Standards and Technology's (NIST) Computer Forensic Tool Testing Programme (CFTT)[1].

The framework is validated using forensic string search, a common task in digital investigations [7]. This validation uses AutoDFBench to query five LLMs to generate executable string search code,

[1] https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt

execute it, and evaluate the generated code. The CFTT string search dataset serves as the ground truth.

This work makes the following contributions:

(1) Design and implementation of an open-source benchmarking framework, AutoDFBench: A robust framework for evaluating AI-generated scripts in digital forensic use cases, accessible at the following GitHub link: https://github.com/akila-UCD/AutoDFBench.
(2) Demonstration of the framework: A comparison of five state-of-the-art LLMs performing string search, resulting in the most extensive known comparison of generated DF code with 24,200 unique tests.
(3) Insights on LLM performance: The findings reveal that none of the LLMs performed satisfactorily, highlighting the need for practitioners to validate the LLM output and the importance of well-tested tools in forensic investigations.

2 Background

2.1 Computer Forensics Tool Testing Program (CFTT)

The typical phases of the DF process commonly rely on both proprietary and open-source tools for accurate data acquisition and analysis, leading to evidence discovery. Ensuring the reliability of these tools is critical, and the National Institute of Standards and Technology (NIST) addresses this through the Computer Forensics Tool Testing Program (CFTT). The program establishes a framework to test forensic tools by defining specifications, criteria, procedures, and test sets to validate their effectiveness and reliability[2]. CFTT provides protocols for various subfields, including Windows registry forensics, deleted file recovery, disk imaging, file carving, mobile devices, cloud data extraction, SQLite forensics, string search, and write blockers.

2.2 Large Language Models

An LLM is a neural network-based model with billions of parameters trained on extensive text datasets. These models understand and generate human language by recognising relationships between words and phrases [5, 15]. LLMs have revolutionised natural language processing (NLP), excelling in various tasks rather than being limited to specific functions.

Fine-tuned LLMs are adapted for specialised tasks in domains such as security, medicine, engineering, and business. This involves retraining the model on domain-specific datasets, enabling them to perform tasks such as threat detection, clinical decision-making, and business process automation [16].

2.3 HumanEval

HumanEval is a benchmark for assessing the code generation capabilities of generative AI systems, including LLMs. It consists of Python programming tasks, where each task includes a problem description and a test suite to verify the generated code [3].

The evaluation metric, pass@k, measures the percentage of correctly solved problems, with pass@1 indicating success on the first attempt. HumanEval is widely used to evaluate and compare LLMs such as GPT-4 and StarCoder for their accuracy and functionality in code generation.

[2]https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt

3 Related Work

The admissibility of electronic evidence requires a rigorous and scientific approach to validate DF tools. Guo et al. [6] proposed a functionality-orientated paradigm for tool testing, focussing on search functions. Their methodology includes mapping functions, specifying requirements, and developing reference test cases to assess tools' performance against standardised criteria, thereby aiding in the design of new validation frameworks.

The use of LLMs in DF is nascent but promising, as highlighted in prior studies. ChatGPT has been explored for tasks such as programming, recovering encryption keys, file carving, and keyword searching, demonstrating the potential for investigation and education [13]. Henseler and van Beek [8] showed ChatGPT's utility in the analysis phase, acting as a copilot to examine artefacts and provide instructions. However, they noted inherent issues with hallucinations in LLMs, suggesting that Augmented Language Models (ALMs) might mitigate these shortcomings [8]. Other studies have explored the potential for automating script generation, question answering, and sentiment analysis while acknowledging risks such as hallucinations, bias, and legal concerns [14].

Wickramasekara et al. [16] introduced a usability matrix categorising DF phases as low, medium, or high potential for LLMs. They highlighted Multimodal Large Language Models (MLLMs) for investigations and analysis. Similarly, Michelet and Breitinger [10] demonstrated LLMs' ability to generate DF reports using Llama2 and ChatGPT 3.5. By inputting data from a Universal Forensic Extraction Device (UFED) Physical Analyser, the researchers demonstrated that while LLMs can automate report generation, the quality depends on the model size, and hallucinations remain a drawback, stressing the need for standardised reports.

The potential of LLMs for generating forensic intelligence graphs (FIGs) was explored by Xu et al. [19]. Using GPT-4-turbo, they reconstructed FIGs from data extracted from mobile devices. Their approach achieved 91.7% coverage for evidence entities and 93.8% for relationship coverage, showcasing the utility of LLMs in building evidence networks.

Wickramasekara and Scanlon [17] proposed a framework that uses Microsoft AutoGen for DF investigations. This framework utilises AI agents with multiple pretrained LLMs to perform DF tasks, bridging knowledge gaps among investigators and enhancing efficiency. However, the researchers highlighted language proficiency as a dependency that affects agent performance.

volGPT, a Volatility 3 plugin, employs LLMs to evaluate ransomware in memory dumps through prompt-based techniques [11]. It uses JSON-formatted process data to identify ransomware. Limitations include challenges in detecting fileless malware or obscured processes [11].

Despite these advancements, Breitinger et al. [2] emphasised the rapid growth of AI in DF in recent years and the need for new research avenues to effectively leverage AI and LLMs. Although prior work has explored LLM applications in DF, a structured mechanism to evaluate their results remains absent. This work addresses

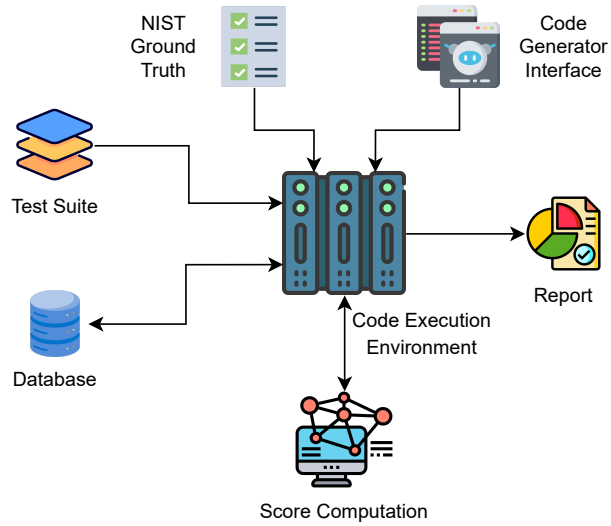


Figure 1: Overview of the proposed framework

this gap by proposing a framework to evaluate AI-assisted contributions, allowing the evaluation of LLM effectiveness in forensic scenarios.

4 Framework Design

This section outlines the design considerations and provides a detailed technical explanation of the framework.

4.1 Design Considerations

The framework is built for extensibility, enabling customisation for diverse experiments. Parameters such as the base prompt, testing disk image types, and the number of cross-validations can be tailored to specific needs. Its flexibility also includes seamless integration with various LLMs, as the framework is LLM agnostic. It supports local models such as Llama 3, StarCoder 2, and WaveCoder, as well as remote models such as GPT-4o and Claude 3.5 Sonnet, accessed via APIs integrated through the LLM Python library[3]. Key variables and configurations are stored in a MySQL database and a configuration file, allowing the smooth incorporation of new models or updates and ensuring adaptability to future technological advancements.

The framework also prioritises reproducibility and architecture agnosticism. By maintaining all variables, prompts, and results in a MySQL database, experiments can be reliably replicated under identical conditions, enabling consistent validation of LLMs in DF. The architecture-agnostic design further enhances its versatility, ensuring compatibility across various computing environments. Whether hosted on GPU-enabled local servers or deployed on cloud platforms, Ollama Docker containers ensure smooth operation regardless of hardware or software infrastructure. These features, illustrated in Figure 1, establish the framework as a flexible and robust solution for a wide range of forensic validation tasks.

[3]https://llm.datasette.io/en/stable/

4.2 Database

The AutoDFBench database is designed to support the management and analysis of experiments. Key components include tables for dynamic configurations, experiment management, and results storage. The configuration table tracks parameters such as API URLs, API keys, and disk paths, while the jobs table manages experiment definitions, including test parameters such as model type, disk image type, and script settings.

Prompt generation and test outcomes are recorded, including metadata such as response times, token counts, and execution statuses. These details enable a comprehensive performance tracking and facilitate analysis in various test scenarios. The results are categorised by hit types (active, deleted, and unallocated), allowing for a detailed performance evaluation and accuracy scoring.

4.3 Software

The framework operates in four key phases: API handling, code preparation, code execution, and summary generation, as illustrated in Figure 3 and detailed in Section 6.

In the API handling phase, the framework retrieves job details and base prompts, combines them with additional input, and generates responses using specified language models. API calls for remote models like GPT-4o and Claude 3.5 Sonnet adhere to rate limits and token usage policies, ensuring reliable operation.

During code preparation, the system processes generated prompts, extracts, and saves code snippets from the LLMs' responses, and organises them into structured output folders for traceability. Once prepared, these scripts are queued for execution.

The code execution phase runs the scripts, verifying disk paths and permissions to ensure valid test environments. A fallback mechanism addresses corrupted or inaccessible disk paths. The results of the execution, including any errors, are recorded for analysis, with a timeout limit ensuring testing efficiency for badly formatted code.

Finally, test results are validated against ground truth data in the summary phase and metrics such as precision, recall, and F1 scores are calculated. These results are stored for detailed performance analysis and benchmarking.

The framework is modular and configurable, allowing each phase to run independently. This flexibility enables users to adapt the system to specific needs and test cases, facilitating comprehensive and scalable evaluations.

4.4 Ground Truth

The framework relies on ground truth data from the NIST CFTT program for validation. These data are formatted and stored in the `ground_truth` table within the database. The framework compares search results with the ground truth during validation to ensure accuracy. More details on the usability of these data are provided in Section 5.1.

4.5 Score Calculation

The framework uses ground truth data to compute precision, recall, and F1 scores for each test run by analysing true positives, false positives, and false negatives.

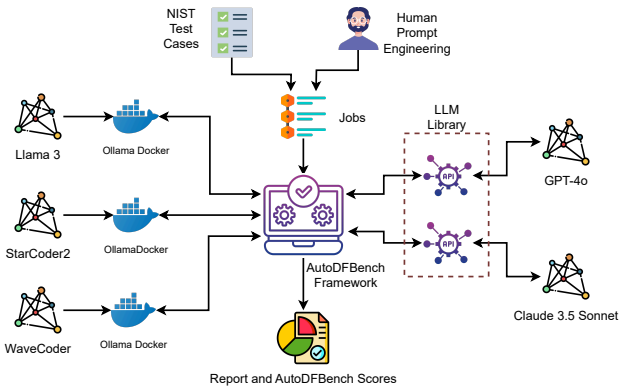


Figure 2: Overview of the proof of concept and evaluation of the framework

True Positives are matches between test results and ground truth data for a specific test case. False Positives are hits in the test results that do not align with the ground truth, indicating incorrect detections. False Negatives are expected outcomes from the ground truth not identified in the test results, representing missed detections.

Based on these metrics, precision, recall and F1 scores are calculated for each subtest. The database stores these scores, and the average F1 score is computed across subtests. This average, referred to as the AutoDFBench score, provides a benchmark to evaluate the performance of different language models and to suggest strategies in the forensic string search.

5 Proof of Concept

This section demonstrates the capabilities of the framework and its practical usability. The evaluation used the Forensic String Search test cases provided by CFTT, employing three open-source LLMs and two closed-source LLMs. Figure 2 provides an overview of the evaluation process for code generated by the LLMs, including Llama 3, StarCoder2, WaveCoder, GPT-4o, and Claude 3.5 Sonnet. The framework integrates human-engineered prompts and NIST test cases as input for testing and validation. It interacts with the LLMs to generate task-specific code, executes the resulting scripts, evaluates outputs against ground truth data, and computes the AutoDFBench Scores to benchmark the models' performance.

5.1 Forensic String Search

String-based evidence is critical in investigations, encompassing data such as natural language text, financial transactions, logs, emails, and other text-based information [1]. String searching is one of the most commonly performed tasks by practitioners [7], making it a practical choice for proof-of-concept testing.

The CFTT defines two core requirements for string search tools: returning exact matches for a given keyword and supporting searches using specific character representations. In addition, 15 non-functional requirements are outlined, including search area specification, stemming, and synonym searches. Based on these criteria, NIST has validated numerous forensic tools [18].

The ground truth data, sourced from NIST's 'Expected Results'[4], lists four-digit numbers each tool must locate, with their file locations specified as Allocated, Unallocated, or Deleted. For Linux, only Allocated and Deleted spaces are covered. Approximately 1,900 test cases for Windows and Linux were added to the `ground_truth` table, focussing on these two locations for Linux experiments.

5.2 LLM Selection

According to Jiang et al. [9], the pass@k = 1 scores for code generation are 84.1% for GPT-4, 82.9% for Claude 3 Opus, 72.6% for StarCoder2-Instruct, 74.4% for CodeFuse, and 81.7% for Llama 3, based on the largest versions of each model. These high HumanEval scores make them suitable for this proof-of-concept.

Anthropic recently reported that Claude 3.5 Sonnet achieves a pass@k = 1 score of 92%, surpassing Claude 3 Opus, while GPT-4o achieves 90.2%, outperforming GPT-4. These evaluations highlight the advances in the code generation capabilities of Claude 3.5 Sonnet and GPT-4o.

For this demonstration, five LLMs were selected: GPT-4o, Claude 3.5 Sonnet, WaveCoder, StarCoder2-Instruct, and Llama 3. These models were chosen for their high HumanEval benchmark performance and compatibility with the Ollama framework, ensuring robust code generation for the framework's tasks [9].

5.3 Implementation

The framework and testing environment were deployed in a Docker-enabled setup. Separate Docker containers were used for the software and MySQL database. The testing server featured a 3.6 GHz Intel Core i7 CPU with 8 cores and 192 GB of RAM, along with NVIDIA GeForce RTX 4090 and RTX 3090 GPUs, each equipped with 24 GB of VRAM, for LLM response generation.

API keys for GPT-4o and Claude 3.5 Sonnet were securely stored in the `config` table. For local LLMs, three individual Ollama Docker containers were configured using `docker-compose.yml` files. These files defined settings such as Docker IP addresses, container names, ports, and GPU preferences. An internal Docker network facilitated communication within the server, and the IP addresses of the configured containers were recorded in the `config` table.

The test disk images provided by NIST, approximately 2 GB each in raw format for Windows and Linux[5], were used for evaluation. These disk images were placed adequately with appropriate permissions to ensure seamless access and execution by the framework.

5.4 Base Prompts

The framework requires base prompts to guide LLMs in structuring their outputs and formulating their approaches. For this experiment, two distinct base prompts were used, each designed to simulate different levels of forensic expertise.

The first base prompt was comprehensive and detailed, providing extensive guidance on conducting a string search. It encompassed 418 words and included suggestions for libraries, Linux commands, and examples. This prompt aimed to replicate the instructions that an advanced forensic investigator might typically provide.

[4]https://cfreds.nist.gov/all/NIST/StringSearch,V11
[5]https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-programme-cftt/federated-testing

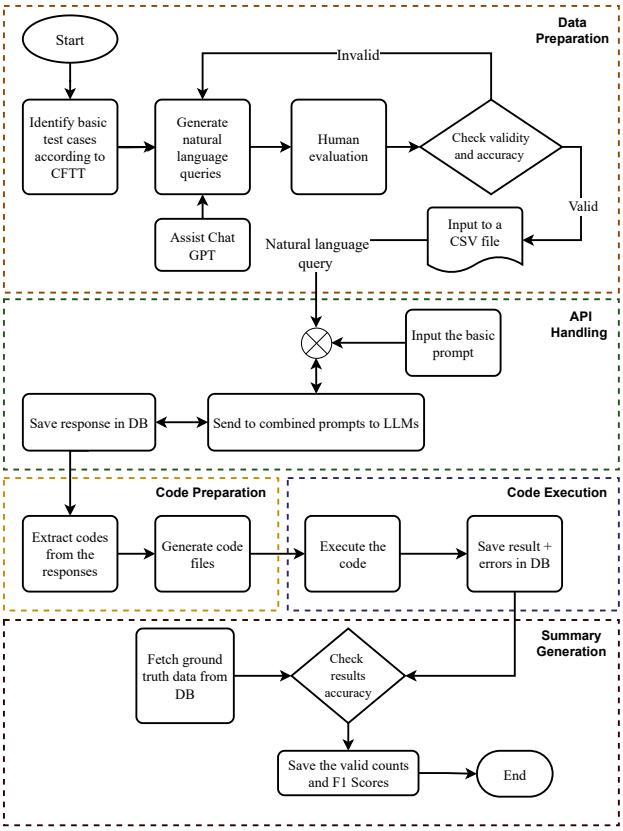


Figure 3: Flow diagram of the experimentation

The second base prompt was concise, containing 105 words. It instructed the LLM to act as an investigator and focus on delivering results without offering detailed procedural hints.

In both cases, the expected output format was standardised to ensure consistency in the results generated by the LLMs.

6 Experiment Flow

As shown in Figure 3, the test data was prepared for seamless integration into the framework. All test cases were compiled into a spreadsheet, and 1,330 human-like prompts were generated using ChatGPT. These prompts were manually reviewed to ensure accuracy before being input into the framework.

During the API Handling phase, these natural language prompts were combined with base prompts and sent to LLMs, generating responses stored in the database. The Code Preparation phase extracted code from these responses and generated the corresponding files. In the Code Execution phase, the framework executed the generated code and saved the outcomes in the database. The executions were performed in a Conda environment pre-configured with libraries for string extraction in Python and tools and dependencies essential for the process. This environment was defined in an `environment.yml` file containing approximately 180 libraries and dependencies. Finally, during the summary generation phase, the outputs were cross-validated against ground truth data, and summary statistics were recorded.

For this experiment, 40 job configurations were defined. These configurations included combinations of five LLMs, two base prompts, two script types, and two disk types, resulting in 40 jobs. Windows test cases included 59 cases, while Unix involved 62, each repeated 10 times for validation, generating 24,200 unique tests. Each job was run within the Conda environment and summary results were logged in the database.

The process of code generation, execution, and evaluation was computationally intensive. Local LLMs averaged approximately 1 minute for code generation and 2 minutes for execution per test, requiring about 484 hours. Cloud-based LLMs (Claude 3.5 Sonnet and GPT-4o) completed code generation in approximately 5 seconds per test but required 3 minutes for execution, totalling 496 hours. Overall, the experimentation took approximately 980 hours for all test cases.

7 Results

This section presents the findings from the proof-of-concept experiments, examining the performance of LLMs in string search tasks. The analysis focusses on the impact of different triggers, the capabilities of LLMs, and the factors influencing their accuracy, efficiency, and reliability. All experiments were conducted as 0-shot tests with the LLMs and results were obtained by averaging hits across subtests compared to the ground truth. Table A.1 details the results of the Windows and Linux test cases, classified by LLM, OS, coding language, and prompt level (beginner or advanced). The average F1 scores were also calculated for each test case. These F1 scores are then averaged across all test cases to produce the AutoDFBench forensic string search (FSS) score for each LLM.

Despite a poor performance, Claude 3.5 Sonnet and GPT-4o achieved the highest benchmarks, with values of 0.043 and 0.036, respectively. The results highlight that advanced prompts consistently produce higher F1 scores, demonstrating the critical role of input prompt detail in the generation of effective code. Among open-source LLMs, WaveCoder achieved the highest F1 score with advanced prompts, suggesting its potential for DF fine-tuning.

In particular, Claude 3.5 Sonnet and StarCoder2-Instruct successfully identified all social security numbers in the Linux environment. High hits in the phone number test case suggest that most LLMs, except Llama 3, generated an accurate code to identify numeric string values. For ASCII-related string searches, the accuracy in locating all search strings was also consistently high.

Furthermore, of the 4,840 test runs per LLM, it is found that GPT-4o and WaveCoder achieved an F1 score of 1 in just 11 instances, while Claude 3.5 Sonnet and StarCoder2-Instruct achieved it 9 and 3 times, respectively. However, Llama 3 failed to achieve an F1 score of 1 in any run.

To determine the best performance run among the ten trials for each test case, the highest F1 score was selected, as shown in Table A.1. The AutoDFBench score was calculated by averaging the F1 scores in all subtest cases, with equal weight assigned to each subtest. The results indicate that Claude 3.5 Sonnet achieved the highest AutoDFBench score of 0.421 using the advanced prompt,

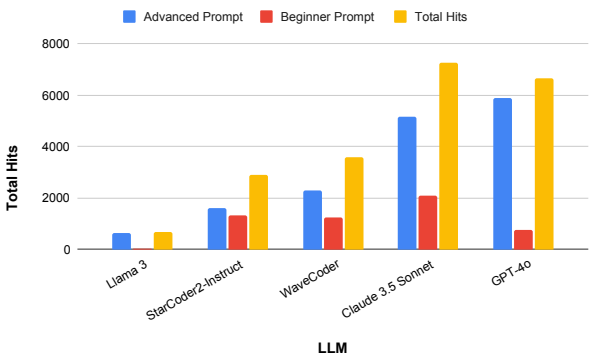


Figure 4: Summary of total keywords found

followed by GPT-4o. Among open-source models, WaveCoder outperformed Llama 3 and StarCoder2-Instruct in average F1 score, reaffirming its potential for AI-driven digital forensic applications.

The significant gap between the 'best run' scores and the average scores from the ten runs highlights the variability inherent in generative AI systems. Although the higher 'best run' scores suggest promising future possibilities for fine-tuned LLMs tailored for digital forensic code generation, the consistently low average scores underscore the current limitations of generic LLMs in delivering reliable and consistent performance for this use case.

The framework also facilitates evaluating the impact of user prompt quality/detail. In the evaluation performed as part of this paper, two levels of simulated expertise were analysed with each of the LLMs. Figure 4 summarises the total number of keyword search hits for each LLM. The results show that advanced prompts, which offer more detailed instructions, lead to higher hits than beginner prompts, which provide minimal guidance.

Although GPT-4o achieved the highest hit count with advanced prompts, Claude 3.5 Sonnet performed best considering total hits across all prompt levels. Among open-source LLMs, WaveCoder ranked highest in finding accurate keywords. Llama 3 had the lowest hit count, suggesting that it may not be a suitable choice for DF code generation fine-tuning.

7.1 Discussion

GPT-4o and Claude 3.5 Sonnet accurately identified only 5.5% and 4.5% of cases, respectively, demonstrating the limited capabilities of advanced commercial LLMs for digital forensic string searches. WaveCoder outperformed StarCoder2-Instruct and Llama 3 among open-source models, with 5.5%, 1.5%, and 0% keyword search accuracy, respectively. Llama 3 showed negligible utility for DF tasks.

Despite being 0-shot responses from generic LLMs, the models tested achieved relatively satisfactory keyword search hit rates and F1 scores for *some* of the individual test cases, indicating reasonable performance without specific training for those specific tasks. However, this performance was inconsistent across all forensic string search test cases used as part of the validation of the benchmarking framework. Significant improvement could be achieved through a combination of better prompt design, AI agents, and fine-tuned models tailored to digital forensics, improving both accuracy and reliability. This experiment used only two base prompts, suggesting that future work could explore a wider variety of prompts. Although character encoding was not specified or validated, the framework's flexibility allows such considerations to be incorporated into future evaluations.

8 Conclusion

This paper introduced AutoDFBench, a novel framework and benchmarking score to test and evaluate AI-generated DF code and tools. The framework encompasses four main components: data preparation, API handling, code execution, and summary and score generation. Built using a MySQL database and Python, AutoDFBench is designed for flexibility, allowing seamless integration with generative AI systems, including multiple LLMs via Ollama Docker containers or remote API calls.

AutoDFBench leverages ground truth data provided by NIST, using forensic string search as a proof-of-concept evaluation method to test the effectiveness of various open-source and commercial LLMs in DF applications. Beyond its current use case, the framework is capable of evaluating prompts and iterating fine-tuned DF-focused generative AI systems. Its modular nature also allows for future integration with an API, enabling centralised retrieval of results and the possibility of assessing non-AI-generated digital forensic tools and code.

To validate its capabilities, the framework was tested using NIST CFTT's forensic string search, encompassing 24,200 tests across five LLMs. These tests demonstrated the robustness of the framework, while revealing that state-of-the-art code-generation LLMs are not yet fully equipped to handle DF-specific tasks. GPT-4o and Claude 3.5 Sonnet achieved the highest performance of the tested models, but with modest F1 scores of 0.043 and 0.036, respectively. Open-source models such as Llama 3, WaveCoder, and StarCoder2-Instruct exhibited even more limited capabilities. However, a marked improvement was observed for all LLMs when provided with detailed "advanced user" prompts, underscoring the critical role of prompt engineering, AI agents and model fine-tuning in enhancing DF task performance.

In conclusion, AutoDFBench represents a significant step forward in the validation and evaluation of AI-generated DF tools. It addresses the growing demand for reliable and scalable solutions in the field, equipping forensic investigators to meet the challenges of the AI era.

The framework will be expanded for future work to evaluate a broader range of AI-assisted DF scenarios. This will include integrating all NIST CFTT test procedures to ensure complete coverage and alignment with established NIST tool testing and validation standards. Such enhancements will bolster the framework's applicability across diverse DF tasks, advancing its utility in identifying the most effective LLMs, prompts, and methodologies for future AI-assisted DF investigations.

A Appendix

Table A.1 details the results of the Windows and Linux test cases, classified by LLM, OS, coding language, and prompt level (beginner or advanced).

Table A.1: AutoDFBench score and the best performing F1-score for LLMs benchmarked over the 10 runs for each subtest

	Llama 3		StarCoder2 Instruct		WaveCoder		Claude 3.5 Sonnet		GPT-4o	
	Beginner	Advanced	Beginner	Advanced	Beginner	Advanced	Beginner	Advanced	Beginner	Advanced
AutoBFBench Score	0	0.001	0.005	0.006	0.011	0.013	0.005	0.036	0.006	0.043
Best Performing Run	0	0.165	0.143	0.197	0.369	0.379	0.354	0.427	0.305	0.411

References

[1] Nicole Lang Beebe and Jan Guynes Clark. 2007. Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results. *Digital Investigation* 4 (2007), 49–54. doi:10.1016/j.diin.2007.06.005

[2] Frank Breitinger, Jan-Niclas Hilgert, Christopher Hargreaves, John Sheppard, Rebekah Overdorf, and Mark Scanlon. 2024. DFRWS EU 10-year review and future directions in Digital Forensic Research. *Forensic Science International: Digital Investigation* 48 (2024), 301685. doi:10.1016/j.fsidi.2023.301685 DFRWS EU 2024 - Selected Papers from the 11th Annual Digital Forensics Research Conference Europe.

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. *CoRR* abs/2107.03374 (2021). arXiv:2107.03374 https://arxiv.org/abs/2107.03374

[4] Xiaoyu Du, Chris Hargreaves, John Sheppard, Felix Anda, Asanka Sayakkara, Nhien-An Le-Khac, and Mark Scanlon. 2020. SoK: Exploring the State of the Art and the Future Potential of Artificial Intelligence in Digital Forensic Investigation. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (Virtual Event, Ireland) *(ARES '20)*. Association for Computing Machinery, New York, NY, USA, Article 46, 10 pages. doi:10.1145/3407023.3407068

[5] Lizhou Fan, Lingyao Li, Zihui Ma, Sanggyu Lee, Huizi Yu, and Libby Hemphill. 2023. A Bibliometric Review of Large Language Models Research from 2017 to 2023. *CoRR* abs/2304.02020 (2023). doi:10.48550/ARXIV.2304.02020 arXiv:2304.02020

[6] Yinghua Guo, Jill Slay, and Jason Beckett. 2009. Validation and Verification of Computer Forensic Software Tools—Searching Function. *Digital Investigation* 6 (2009), S12–S22. doi:10.1016/j.diin.2009.06.015 The Proceedings of the Ninth Annual DFRWS Conference.

[7] Christopher Hargreaves, Frank Breitinger, Liz Dowthwaite, Helena Webb, and Mark Scanlon. 2024. DFPulse: The 2024 digital forensic practitioner survey. *Forensic Science International: Digital Investigation* 51 (2024), 301844. doi:10.1016/j.fsidi.2024.301844

[8] Hans Henseler and Harm van Beek. 2023. ChatGPT as a Copilot for Investigating Digital Evidence. In *Proceedings of the Third International Workshop on Artificial Intelligence and Intelligent Assistance for Legal Professionals in the Digital Workplace (LegalAIIA 2023) co-located with the 19th International Conference on Artificial Intelligence and Law (ICAIL 2023), Braga, Portugal, June 19, 2023 (CEUR Workshop Proceedings, Vol. 3423)*, Jack G. Conrad, Daniel W. Linna Jr., Jason R. Baron, Hans Henseler, Paheli Bhattacharya, Aileen Nielsen, Jyothi K. Vinjumur, Jeremy Pickens, and Amanda Jones (Eds.). CEUR-WS.org, 58–69. https://ceur-ws.org/Vol-3423/paper6.pdf

[9] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A Survey on Large Language Models for Code Generation. arXiv:2406.00515 [cs.CL] https://arxiv.org/abs/2406.00515

[10] Gaëtan Michelet and Frank Breitinger. 2024. ChatGPT, Llama, can you write my report? An experiment on assisted digital forensics reports written using (local) large language models. *Forensic Science International: Digital Investigation* 48 (2024), 301683. doi:10.1016/j.fsidi.2023.301683 DFRWS EU 2024 - Selected Papers from the 11th Annual Digital Forensics Research Conference Europe.

[11] Dong Bin Oh, Donghyun Kim, Donghyun Kim, and Huy Kang Kim. 2024. volGPT: Evaluation on triaging ransomware process in memory forensics with Large Language Model. *Forensic Science International: Digital Investigation* 49 (2024), 301756. doi:10.1016/j.fsidi.2024.301756 DFRWS USA 2024 - Selected Papers from the 24th Annual Digital Forensics Research Conference USA.

[12] Partha Pratim Ray. 2023. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems* 3 (2023), 121–154. doi:10.1016/j.iotcps.2023.04.003

[13] Mark Scanlon, Frank Breitinger, Christopher Hargreaves, Jan-Niclas Hilgert, and John Sheppard. 2023. ChatGPT for digital forensic investigation: The good, the bad, and the unknown. *Forensic Science International: Digital Investigation* 46 (2023), 301609. doi:10.1016/j.fsidi.2023.301609

[14] Mark Scanlon, Bruce Nikkel, and Zeno Geradts. 2023. Digital forensic investigation in the age of ChatGPT. *Forensic Science International: Digital Investigation* 44 (03 2023), 301543. doi:10.1016/j.fsidi.2023.301543

[15] Yiqiu Shen, Laura Heacock, Jonathan Elias, Keith D. Hentel, Beatriu Reig, George Shih, and Linda Moy. 2023. ChatGPT and Other Large Language Models Are Double-edged Swords. *Radiology* 307, 2 (2023), e230163. doi:10.1148/radiol.230163 PMID: 36700838.

[16] Akila Wickramasekara, Frank Breitinger, and Mark Scanlon. 2024. Exploring the Potential of Large Language Models for Improving Digital Forensic Investigation Efficiency. arXiv:2402.19366 [cs.CR] https://arxiv.org/abs/2402.19366

[17] Akila Wickramasekara and Mark Scanlon. 2024. A Framework for Integrated Digital Forensic Investigation Employing AutoGen AI Agents. In *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*. 01–06. doi:10.1109/ISDFS60797.2024.10527235

[18] Tina Wu, Frank Breitinger, and Stephen O'Shaughnessy. 2020. Digital forensic tools: Recent advances and enhancing the status quo. *Forensic Science International: Digital Investigation* 34 (2020), 300999. doi:10.1016/j.fsidi.2020.300999

[19] Hanxiang Xu, Shenao Wang, Ningke Li, Kailong Wang, Yanjie Zhao, Kai Chen, Ting Yu, Yang Liu, and Haoyu Wang. 2024. Large Language Models for Cyber Security: A Systematic Literature Review. arXiv:2405.04760 [cs.CR] https://arxiv.org/abs/2405.04760

Fast Synthetic Data Generation for Case-Specific Entity Extraction in Criminal Investigations

Mads Skipanes
The National Criminal Investigation Service (Kripos)
Oslo, Norway
madsls@stud.ntnu.no

Nardiena Pratama
School of Electrical Engineering and Computer Science
The University of Queensland
Brisbane, Australia
uqnprata@uq.edu.au

Kyle Porter
Department of Information Security and Communication Technology
The Norwegian University of Science and Technology
Gjovik, Norway
kyle.porter@ntnu.no

Gianluca Demartini
School of Electrical Engineering and Computer Science
The University of Queensland
Brisbane, Australia
g.demartini@uq.edu.au

Abstract

In major criminal investigations, the manual analysis of police reports for the categorization of entities is a resource-intensive task prone to human error. Recent advances in Named Entity Recognition (NER) models offer promising solutions for automating this process, potentially reducing both time and error rates.

This paper demonstrates the effectiveness of fine-tuning a NER model using a publicly shared synthetic dataset inspired by real case files. Notably, we leverage a large language model (LLM) for generating both the synthetic data and the annotations used for training. This approach enables investigators to rapidly develop case-specific models tailored to ongoing investigations. To structure this effort, we propose an ontology for entity extraction in criminal cases, focusing on key entities, such as persons, seized items, communication profiles, vehicles, locations, organizations, and financial profiles. Our model achieves an average weighted F1-score of 94.2% on the synthetic dataset.

For further validation, we manually annotated a small dataset of confidential data from two homicide cases, achieving an average weighted F1-score of 81.6%. Our results demonstrate that our approach can at times generalize well to real case files.

CCS Concepts

• **Applied computing** → **Document searching**; **Evidence collection, storage and analysis**.

Keywords

Criminal Investigation, Artificial Intelligence, Named Entity Recognition, Large Language Models, Synthetic Data, Data Management, Information Analysis.

DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712719

ACM Reference Format:
Mads Skipanes, Nardiena Pratama, Kyle Porter, and Gianluca Demartini. 2025. Fast Synthetic Data Generation for Case-Specific Entity Extraction in Criminal Investigations. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3712716.3712719

1 Introduction

Criminal investigations are information work [10]. Investigators collect, analyze, and synthesize information for knowledge discoveries to answer the question of 'What has happened?'. Within this workflow, the analytical stage consists of thematization and categorization of collected information [25]. We have previously addressed the challenge of thematization [26]. However, in this paper, we focus on the task of categorization.

In this context, the term 'categorization' refers to assigning information to entity classes. This task is performed as investigators often do not understand the meaning of the collected data. However, regardless of meaning, the text in the data can be grouped into classes for later knowledge discoveries. For instance, information about a person's user account cancellation might seem insignificant when first received. Yet, when combined with other data sources, such as transactions of a weapon purchase, the process of categorizing these entities can lead to valuable insights. By categorizing relevant text into entity classes, investigators would be capable of uncovering relationships, such as temporal, relational, and geographical connections that are crucial for them to solve criminal cases. Thus, categorization is assigned to specific roles such as the Document Reader, Indexer, and Indexing Supervisor [18]. This highlights the task's dual nature: it is resource-intensive and a high priority in criminal investigations.

Within major criminal investigations, thousands of police documents, such as incident reports, interviews, warrants, statements, and crime scene reports, are generated leading to large volumes of unstructured text. Currently, the Document Reader, Indexer, and Indexing Supervisor categorize information manually. Within two homicide cases applied in this study, we revealed 13,379 manually created connections between entities and documents. This number

demonstrates the workload, and highlights its vulnerability to human errors. With such an overwhelming number of entity involved in generation and linkage, there is a risk that information may be overlooked. This underlines the need for more efficient and reliable methods for the task of categorization.

Entity extraction has a long tradition within the legal domain, aiming to support investigators in categorizing information [5]. In recent years, researchers have proposed systems for NER in criminal investigations [23] based on Bidirectional Encoder Representations from Transformers (BERT) [6]. These methods have shown great results over traditional methods such as lexical lookup and rule-based entity extraction.

After the release of generative Large Language Models (LLM), more and more powerful approaches to NER have emerged [21, 30]. However, these advancements have primarily been applied to open or constructed datasets. The confidentiality of real criminal investigative data presents a significant challenge for utilizing LLMs in real investigations, as well as for the fine-tuning of BERT NER models [23, 31]. The restricted access to data results in a lack of labeled data for fine-tuning NER models, making it difficult to develop NER systems.

Our current work makes the following contributions. First, we present an ontology for named entities in criminal investigations. Second, we present an approach leveraging an LLM for fast synthetic data generation for fine-tuning of a case-specific NER model. Third, we share our *Annotated Synthetic* dataset used for fine-tuning, and report our results. Finally, we present our findings from applying our model to real criminal investigations. Through this work, we seek to answer the following research question: "*To what extent can NER models fine-tuned on fast synthetic data generalize to real case files?*"

The remainder of this paper is as follows: Section 2 presents related work, followed by Section 3 which describes our methodology, including an overview of the three datasets applied in our study, the proposed ontology, and our annotation and fine-tuning process. Section 4 presents our experimental results, while Section 5 discusses our findings. Finally, Section 6 concludes the paper and suggests directions for future work.

2 Related work

NER in Criminal Investigations. The application of NER in criminal investigations encompasses multiple objectives, purposes, and practical applications. Early research by Chau et al. [5] aptly highlighted the necessity of processing both structured databases and unstructured police reports in criminal investigations. Louis and Engelbrecht [15] and Al-Zaidy et al. [1] highlighted the tedious process of manual extraction of useful information in unstructured text. Pollitt [22] explored the application of NER in investigations to enhance forensic analysis and emphasized NER's value for discovering unknown entities by comparing it to string search where entities must be known search terms. This is a key factor in criminal investigations as investigators often lack clarity on what entities to look for and which ones are relevant.

Al-Zaidy et al. [1] proposed a method for discovering criminal networks using NER. Similar work was performed by Iqbal et al. [11] for clique detection in chat logs - an exploratory approach into criminal investigations with the potential to discover entities of importance. Taking a different approach from Al-Zaidy and Iqbal's network analysis, Schraagen et al. [24] explored how out-of-the-box NER models could give police investigators a head start by automatically identifying relevant entities in user-submitted reports. Others have highlighted the importance of document analysis by applying NER to provide an overview of *Person* entities and their linkage to documents [7]. As investigators continuously need to retrieve information from documents, such an approach can benefit them by providing an overview of persons and documents.

Yang and Chow [31] developed a framework for both entity and relation extractions in digital forensic investigations. This is comparable to the work by Banerveld et al. [27], who developed a tool for NER, relation and information extraction to enhance white-collar crime, and Batini et al. [2], who proposed a semantic data integration platform built on NER and relation extraction. Kejriwal et al. [13] also focused on end-to-end systems allowing for question input for the automatic identification of entities towards building knowledge graphs.

To the best of our knowledge, there does not exist any NER model for Norwegian criminal investigations, and there only seems to have been one previous study around fine-tuning for criminal investigations by Ørke [32]. Ørke manually annotated a dataset and fine-tuned the 'nb-bert-base-ner' with an F1-score of 90.6%. For reference, as of the time of writing, the best Norwegian NER model has obtained a micro-average F1-score of 90.13% on the NorNE-Nb Norwegian benchmark dataset,[1] and the model with the best results for the CoNLL 2003 English benchmark obtained a 94.6% F1-score [29].

Collectively, these studies demonstrate NER's fundamental role in enabling knowledge discovery and seek to facilitate more efficient information analysis compared to manual methods. However, very few work with real data due to confidentiality, and several highlight the challenge of manual annotation.

Synthetic Data Generation for NER. Due to confidentiality restrictions, the vast majority of studies have been unable to utilize real-world police reports in their research as recognized by Schraagen et al. [24]. Thus, we explore the workflow of anonymising confidential data and use this data as an example input to an LLM for both data generation and annotation. Oliveira et al. [19] explored the use of language models as an alternative strategy to human-based annotations. They found that human annotation was the best approach. However, a combination of human and LLM annotations showed positive results. Others, such as Park et al. [21] and Xing et al. [30], have also used LLMs for dataset construction, evaluating information extraction techniques on constructed police reports. In contrast to Park and Xing, we present an approach for synthetic data generation for fine-tuning a case-specific BERT NER model in alignment with existing ontology used by Norwegian criminal investigators and we share our synthetic generated dataset.

Transformers and LLMs. Recent work has shifted to applying transformers like BERT [28] and LLMs for information extraction and NER. Rodrigues et al. [23] developed a forensic pipeline for information extraction with transformers showing how NER and

[1] https://scandeval.com/norwegian-nlu/

relation extraction can enable comprehensive knowledge graph construction for forensic analysis. Park et al. [21] recently fine-tuned two language models for information extraction on open verdicts, developing a hybrid classification model that could automatically extract 18 key types of information. The application of LLMs in digital forensics has also begun to expand, with researchers like Henseler and van Beek [8] exploring the potential of using ChatGPT as a copilot in criminal investigations. While not explicitly testing ChatGPT for entity extraction, they investigated the capability of person and role identification. Their results were promising, but they emphasized the need for careful validation and highlighted challenges in implementing such models securely within law enforcement infrastructure.

In this study, we chose to use a BERT-based model for NER due to its relatively low computational requirements compared to LLMs. BERT's adaptability, learning capacity, and ease of updating and expansion made it more suitable for our purposes. These factors influenced our decision not to utilize an LLM for NER in criminal investigations.

3 Methodology

In the following section, we present our methodology. First, we outline the selection of our applied datasets, followed by an overview of our suggested ontology. Last, we elaborate on our annotation and fine-tuning process. All resources including datasets, and fine-tuning code are publicly available in our repository.[2]

3.1 Data

In this study, we utilized three datasets: A *Confidential Case File* dataset, an *Annotated Synthetic* dataset and an *Annotated Confidential* dataset.

The *Confidential Case File* dataset consists of 3,729 police documents within two case files. These documents are listed in a relation database linked to manually extracted entities. The total number of entities sums up to 4,321, with a total of 13,379 unique links. We utilized this dataset for three distinct purposes: (1) ontology generation, (2) synthetic data generation, where selected anonymized sentences served as prompt input for 'Claude 3.5 Sonnet', and (3) evaluation of our fine-tuned model on unseen real-world data, which we discuss in Section 5.1. By the use of 'Claude 3.5 Sonnet' we generated the *Annotated Synthetic* dataset containing 1,458 entities for training and testing. Further, we manually annotated 285 entities from our *Annotated Confidential* dataset for testing. Both datasets use the BIO annotation format (later explained in section 3.3).

3.2 Ontology

Within criminal investigations, a well-defined ontology plays a crucial role in structuring and categorizing information. Ontologies are the main mechanism for domain-specific knowledge representation as stated by Bruckschen et al. [3]. Therefore, we highlight the importance of establishing a unified ontology alongside case-specific entity classes.

Within digital forensics, the need for standardized ontologies has been addressed by initiatives like the Cyber-Investigation Analysis Standard Expression (CASE) and its underlying Unified Cyber Ontology (UCO). CASE provides a comprehensive specification language for representing digital forensic information [4].

This drive towards standardization is also evident in traditional criminal investigations, where the National Police Chief's Council (NPCC) in the UK recently shared a 'Minimum POLE Data Standards Dictionary,' with the intent to support consistent and accurate recording of police information. The dictionary is organized around POLE entity classes: Person, Object, Location, and Event.

Similar ontologies have long existed within the domain of operational analysis in Norway proposed by Nissen [16, 17], namely ORE entities: Objects, Relations, and Events. These are further split into Person, Item, Vehicle, Communication Profile, Organization, Location, and Financial Profile. These entity classes serve as a standard within Norwegian criminal investigations. These classes are split into case-specific entities depending on the type of case as investigators must adapt and work with dynamic entity classes depending on the case.

Upon analyzing the *Confidential Case File* dataset, we have mapped out the frequency of entity classes that the investigators have manually created within these cases. The entity classes and their frequencies are listed in Table 1.

Table 1: Frequency of entities identified in the *Confidential Case Files.*

Entity Class	Frequency
Person	2043
Seized Item	1146
Communication Profile	439
Vehicle	406
Location	234
Organization	50
Financial Profile	3

Building upon Nissen's established framework, we propose **PIVCOLF** - a unified ontology for entity extraction in criminal investigations. PIVCOLF preserves Nissen's seven fundamental classes: **P**erson, **I**tem, **V**ehicle, **C**ommunication Profile, **O**rganization, **L**ocation, **F**inancial Profiles.

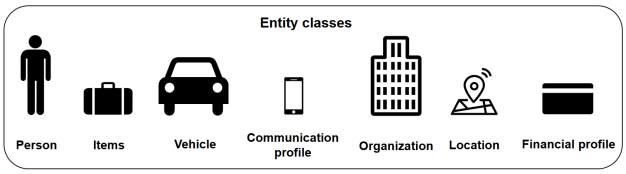


Figure 1: PIVCOLF ontology for entity extraction in criminal investigations

From analyzing entity classes within our *Confidential Case File* dataset, we found that several of these classes had subclasses. *Persons* contained both names and personal identification numbers. For *Items*, this class can consist of any item, however, if an item is seized, it is referenced to by its unique identification reference. For this reason we focus on the seized item reference, not any item.

[2] https://anonymous.4open.science/r/fastner-1650/

Vehicles contained both names and license plate numbers. *Communication profiles* consisted of phone numbers, e-mail addresses, and user accounts. *Organizations* had names and organizational identification numbers. *Location* contained only addresses. *Financial profiles* contained both credit card numbers and bank account numbers. An overview of the different classes and examples are listed in Table 2.

Table 2: PIVCOLF entities with subclasses, labels and examples for criminal investigations.

Entity	Type	Label	Example
Person	Named Entity	PER	First Lastname
	Numeric Entity	PID	112233 45678
Seized Item	Alphanumeric Entity	ITM	2024 / 121 - A2
Vehicle	Named Entity	VEH	BMW i3
	Alphanumeric Entity	LIC	VEH123
Communication profile	Named Entity	EMA	name@online.com
	Numeric Entity	PHO	12 34 56 78
	Named Entity	USR	JohnDoe_12
Organization	Named Entity	ORG	Coca Cola
	Numeric Entity	OID	123 456 789
Location	Named Entity	LOC	Main Street 1, 0101 Oslo
Financial profile	Numeric Entity	FIN	1234 56 78901

The above listed entity classes have been selected for annotation and training in our study.

3.3 Data Annotation and Model Development

With regards to the annotation method, we derived inspiration from authentic police reports within our *Confidential Case File* dataset to ensure relevance to the target domain. We manually generated example sentences to mimic the style and content of the source documents and instructed 'Claude 3.5 Sonnet' to both generate and annotate a series of similar sentences, each series containing between 10-20 sentences. We validated each sentence to ensure quality. These example sentences contained BIO-schema annotations and were used as prompt input. This is a common format used for Named Entity Recognition. B (Beginning): Indicates the first token of a named entity. I (Inside): Indicates a token that is inside a named entity (i.e., not the first token). O (Outside): Indicates a token that is not part of a named entity. An example of the BIO-schema can be viewed in Table 3. Our annotation style followed a space-based tokenizaton, where text was split into individual units based on spaces between words. This 'human way' of labeling allowed for faster annotation.

We sent the following prompt to 'Claude 3.5 Sonnet': *Please generate look-alike annotated sentences using the BIO (Beginning-Inside-Outside) schema. Each line should contain a token and a classification. Make sure there is a new line for each new sentence. Follow this example for annotation style and structure: <Example sentence>. Generate sentences within the domain of <domain>.*

We tracked the time spent on annotation and validation. Each set of 20 sentences required approximately 2 minutes for generation and validation. In total, we produced 817 annotated sentences, consuming 162 minutes.

Table 3: Tokens and their corresponding annotation based on the space-based BIO annotation style.

Token	Annotation
The	O
witness	O
JOHNSON,	B-PER
John	I-PER
uses	O
e-mail:	O
john@online.com.	B-EMA

Additional time was necessary to craft the initial prompt generation for sets of entity classes. We had to adjust the prompt input at times. This step took between 4-7 minutes per prompt, as this involved anonymizing data and manually annotating an example sentence. We spent between 140-168 minutes on this task.

We manually annotated 25 sentences with a total of 285 entities of the *Confidential Case File* dataset following the same style. This took 55 minutes. This annotated dataset, containing unseen confidential data was used to evaluate the performance of our fine-tuned model. The synthetic generation of the annotated datasets was 5.45 times faster to generate and annotate compared the manual annotation of confidential text.

3.4 Fine-Tuning

Our *Annotated Synthetic* dataset was partitioned into training and test sets using k-Fold Cross-Validation (CV), where the dataset is split into five equal-sized partitions (k=5). The model was fine-tuned with four of the partitions (training set) and tested on the final remaining partition (test set). This process was performed k=5 times using different combinations of the partitioned data - each partition will act as the test set once. The scores of the evaluation metrics, such as precision, recall, and F1-scores, for each of the five models are derived and used to calculate the weighted averages.

We use the following standard metrics. *Precision*: The ratio of correctly predicted entities. This can be mathematically defined as: $TP/(TP + FP)$, where TP represents the number of correctly predicted entities for a given class (True Positives) and FP represents the number of entities incorrectly predicted to belong to that class (False Positives). *Recall*: The ratio of correctly predicted entities to the actual entities in the dataset. This can be mathematically defined as: $TP/(TP + FN)$, where TP represents the number of correctly predicted entities for a given class (True Positives) and FN represents the number of entities that actually belong to that class that have been incorrectly predicted as not belonging to that class (False Negatives). *F1-Score*: The harmonic mean of precision and recall (a score that balances precision and recall). In addition to individual entity performance, we report three types of average scores. The *micro average* provides an overall score weighted by the frequency of each class. The *macro average* gives equal weight to each class, regardless of its frequency. The *weighted average* is similar to the macro average but weighted by the support size of each class. These scores offer an overall performance across all entity classes.

To see how well the fine-tuned models perform on the *Annotated Confidential* dataset, the models trained by each of the folds are saved and used for further evaluation on this dataset. We obtained the same evaluation metrics as in the *Annotated Synthetic* dataset evaluation and used these to calculate the weighted averages. Testing on the models from the k-Fold CV process ensures that the model evaluation on the test set and the *Annotated Confidential* dataset is performed fairly.

We leveraged the ScandiNER,[3] a NER model for Danish, Norwegian, Swedish, Icelandic, and Faroese which is available under the MIT License. It is a fine-tuned version of the Nasjonalbiblioteket AI Lab's (NBAILab) "NoTram", short for "Norwegian Transformer Model", available under the Apache License, Version 2.0 [14]. The model has been fine-tuned on datasets containing Scandinavian languages, such as the DaNE, NorNE, SUC 3.0[4] and the Icelandic and Faroese sections of the WikiANN dataset [9, 12, 20]. Due to this, the model performs well on Scandinavian languages, which makes it specific enough when used for Norwegian documents, but not overly generalizable to non-Norwegian documents, making it well-suited for our use-case.

For the fine-tuning process, we utilised NBAILab's Colaboratory Notebook on "How to finetune a NER/POS-model", which is available on their Github Project page.[5] The code has been adapted so that it fine-tunes the ScandiNER model on the *Annotated Synthetic* dataset. Fine-tuning is a process where the weights of a foundation model is updated to make it more adjusted to new data. This is unlike using a pre-trained model directly, which does not involve further adaptations. Moreover, BERT models, which the ScandiNER was fine-tuned on, were trained to understand natural languages and require fine-tuning for particular Natural Language Processing (NLP) tasks. In this work, the ScandiNER model is further fine-tuned to allow it to identify new entity classes within the context of Norwegian criminal investigations. The model was trained using a learning rate of 3e-05 over 4 epochs and 750 warm-up steps, which is the default set-up in the NBAILab notebook. The tokenizer in the model used no padding.

4 Results

In this section, we present our results. We first present the results of applying the ScandiNER model 'out of the box'. We then present the results of our fine-tuned model on our *Annotated Synthetic* dataset, and last we present the results of our fine-tuned model on the *Annotated Confidential* dataset.

Today's existing NER models have not been trained on the entity classes presented in our suggested ontology, PIVCOLF, except for *Person*, *Location*, and *Organization*. Thus, we could only apply the ScandiNER model to predict these entity classes. We evaluated the ScandiNER model both 'out of the box' and after fine-tuning using our *Annotated Synthetic* dataset. Table 4 presents the performance comparison between the ScandiNER model and our fine-tuned version. The Support column indicates the occurrences for each entity presented in the data used for evaluation. Both models were evaluated on our synthetic dataset. As indicated, our fine-tuned model outperformed ScandiNER in F1-scores across every entity class.

Table 4: Scandi Base vs. Scandi FT

Entity	Model	Precision	Recall	F1-Score	Support
PER	Scandi Base	0.834	0.954	0.890	285
	Scandi FT	0.976	0.982	0.979	285
ORG	Scandi Base	0.745	0.913	0.820	115
	Scandi FT	0.869	0.913	0.888	115
LOC	Scandi Base	0.688	0.778	0.730	207
	Scandi FT	0.891	0.894	0.892	207

Performance metrics of our fine-tuned model, evaluated across all entity classes on our *Annotated Synthetic* dataset, are presented in Table 5. Our fine-tuned model achieved strong performance with an F1 weighted average score of 94.2% and F1-scores >90% for most entity classes, with F1-scores of 89.2% and 88.8% observed for *Location* and *Organization*.

Table 5: 5-fold Cross Validation - Synthetic Data

Entity	Precision	Recall	F1-Score	Support
PER	0.976	0.982	0.979	285
PID	0.910	0.952	0.930	83
ITM	0.973	1.000	0.986	106
VEH	0.949	0.948	0.948	77
LIC	0.944	0.875	0.908	80
EMA	0.988	0.988	0.988	167
PHO	0.936	0.962	0.949	105
USR	0.908	0.908	0.907	98
ORG	0.869	0.913	0.888	115
LOC	0.891	0.894	0.892	207
FIN	0.953	0.943	0.940	87
Micro avg	0.937	0.947	0.942	1458
Macro avg	0.935	0.948	0.939	1458
Weighted avg	0.939	0.947	0.942	1458

The lower performance on *Location* and *Organization* entities reflects the domain characteristics in criminal investigation data. Unlike traditional location entities (e.g., 'Oslo'), our data includes detailed addresses (e.g., 'Main Street 1, postcode 0901 Oslo, Norway'). Similarly, our *Organization* entities can often be comprised of lesser-known criminal groups (e.g., 'DarkSide', 'Bandidos') rather than well-known company names. Additionally, we found that *User Account* entities were challenging to identify due to their highly variable formats (e.g., 'John_12', 'JohnZ'), suggesting a need for additional training data in this category.

For the evaluation of the *Annotated Confidential* dataset, results are listed in Table 6. We see a drop in performance compared to the *Annotated Synthetic* dataset. However, from synthetic generation, we see an overall weighted average F1-score of 81.6% for all entity classes. The performance metrics for the entity classes *Financial Profile*, *Organization*, *Personal Identification Number*, and *User Account* indicate a need for refinement and additional training data.

[3] https://huggingface.co/saattrupdan/nbailab-base-ner-scandi
[4] https://spraakbanken.gu.se/en/resources/suc3
[5] https://github.com/NbAiLab/notram#user-content-colab-notebooks

Table 6: 5-fold Cross Validation - Confidential Data

Entity	Precision	Recall	F1-Score	Support
PER	0.897	0.908	0.902	260
PID	0.511	0.846	0.637	65
ITM	0.842	0.909	0.872	55
VEH	0.905	0.739	0.813	115
LIC	0.878	0.867	0.871	90
EMA	1.000	0.933	0.965	120
PHO	0.965	0.925	0.944	265
USR	0.698	0.400	0.497	40
ORG	0.624	0.600	0.610	75
LOC	0.629	0.797	0.702	315
FIN	0.592	0.560	0.543	25
Micro avg	0.792	0.833	0.812	1425
Macro avg	0.776	0.771	0.760	1425
Weighted avg	0.813	0.833	0.816	1425

5 Discussion

The effort required for manual entity extraction from unstructured police reports is comparable to that of BIO-annotation tasks—both being highly time-intensive processes. This considerable time investment motivated our exploration of automated approaches, leading us to utilize 'Claude 3.5 Sonnet' for both the data generation and the annotation task. We have saved a significant amount of time compared to manual annotations. While the generation and annotation of synthetic data proved 5.45 times more time-efficient compared to manual annotation of our confidential data, we needed to validate how well models trained on such synthetic data would generalize to real criminal case files. To evaluate this, we manually annotated a confidential dataset from actual case files to serve as a test bed for our synthetically trained model.

From a relatively small dataset of 1,458 entities in total, we find inspiring results showing that there is a capability for fine-tuning by applying our proposed method. Within our synthetic dataset, we see strong performance with a weighted average of 94.2%. Notably, this performance exceeded that reported by Ørke [32], who used manual annotation for fine-tuning the 'nb-bert-base-ner' model. However, it's important to note that these results are not directly comparable due to differences in datasets and underlying model architectures.

However, our model's performance dropped on the confidential dataset. It is important to emphasize that our model was trained exclusively on synthetic data, with no exposure to confidential information. While we generated synthetic data to closely mirror investigative texts, the model's performance showed a decrease when applied to unseen, confidential real-world data. This performance gap suggests that enhancing the similarity between our synthetic training data and confidential data could improve the model's generalization capabilities.

Even though the performance decreased, we ultimately obtained a weighted average F1-score of 81.6%. We argue this shows that fast synthetic generation for criminal investigations generalizes well and can be a method for developing case-specific NER models. The model's performance for the entity classes *Financial Profile*, *Organization*, *Personal Identification Number*, and *User Account*, fell below our initial expectations. Specifically, these classes exhibited lower precision and recall compared to other entity classes in our dataset.

Our results indicate two key areas for improvement: increasing the volume of training data and enhancing the quality of entity representation in our synthetic dataset. Despite our efforts to generate representative data, there seems to remain a gap between the linguistic patterns in synthetic texts and those found in confidential police reports. This disparity can be attributed to 'Claude 3.5 Sonnet's' training limitations—specifically, its lack of exposure to confidential investigative documents and potential inherent biases in the model that may affect its ability to generate authentic investigative language and terminology. This results in text that differs from the characteristic formulations used by criminal investigators. Thus, we want to explore the generation of synthetic data by an LLM and criminal investigators similar to Oliveira et al. [19] for fine-tuning and testing.

We must underline that we worked with entities the same way investigators do. This means we applied a space-based annotation method as shown in Table 3. This may also affect the results, though we found this to be a faster and easier way to annotate.

If we take into account the weighted average scores of 94.2% for the *Annotated Synthetic* dataset and 81.6% for the *Annotated Confidential* dataset, respectively, and the fact that our generation of training sets was 5.45 times faster than manual annotation, we suggest that investigators should consider utilizing this method in major criminal investigations.

While several of our entity classes within PIVCOLF could potentially be identified using rule-based approaches like regular expressions, these methods face challenges when dealing with the broad variability in police documents. Our experience from working with police documents has shown that investigators formulate entities with a broad variety. Even with a comprehensive dictionary of regex patterns, ambiguity remains a challenge. For instance, distinguishing between *phone numbers* (PHO), case file numbers, and *organizational identification numbers* (OID) requires contextual understanding, as they can share identical numerical patterns (e.g., sequences of 8-10 digits). Additionally, a *financial profile* (FIN) can be identical to a *personal identification number* (PID). Thus, we opted for BERT as it can learn contextual patterns and disambiguate between similar numeric formats based on their surrounding context, making it more suitable for handling such overlapping patterns and variations in entity representation. While recent LLMs have shown impressive capabilities, BERT offers several advantages for our specific NER task. It requires significantly less computational resources, enabling wide spread local deployment. It can process sensitive police data locally without requiring external API calls, thereby maintaining data privacy and security and BERT provides consistent entity labeling without the formatting variations and hallucinations often seen in generative models.

Our study has concentrated on fine-tuning the ScandiNER model for case-specific entity extraction under the ontology we named PIVCOLF. While this approach has shown promise, we recognize the need to evaluate its performance across a broader spectrum of investigative scenarios. Future work should explore the application

of this fine-tuning methodology to cases involving other entities such as weapons, drugs, cryptocurrency, geographical coordinates, and IP addresses.

It is crucial to underline the potential risk of data leakage inherent in our proposed method. If investigators decide to use this approach, they must ensure not to include any confidential data in the example sentences they provide for annotations to the externally hosted LLM.

5.1 Lessons Learned from Real Use-Cases

We applied our model to the *Confidential Case File* dataset, comparing its output to entities manually extracted by the investigator through string-matching rules. This analysis revealed an important aspect of human investigative work: investigators are selective when categorizing information, prioritizing the relevant entities. For example, they tend to focus on the mentioning of the suspect (PER) rather than defense attorneys (PER). In contrast, our model does not make these nuanced distinctions, resulting in more comprehensive but less focused extracted entities compared to the investigator's manual work.

A successful extraction model should focus solely on relevant investigative information within case file reports, disregarding metadata and administrative details. For instance, the location where police interviews were conducted is typically not pertinent to the ongoing investigation. To achieve this, we need to define specific sections of the documents for entity extraction, allowing us to concentrate on the relevant content while ignoring extraneous administrative information.

A key differentiation between humans and computational categorizations lies in the interpretations within the context. Investigators apply their knowledge, such as creating a location by its street name for the crime scene, even though this is not explicitly mentioned in the text. This is a dominant approach we see from the investigators. While this type of inference falls beyond the scope of traditional NER, which focuses on explicit entity mentions, it could potentially be addressed through co-reference solutions, relation extraction models, or rule-based systems that capture domain-specific relationships and contextual patterns.

We motivate investigators to rethink the way they perform their categorization of entities by applying a new method such that machine learning methods can be utilized. By structuring their annotations in a format that supports both case-specific knowledge discovery and machine learning requirements, investigators can contribute to the continuous improvement of machine learning models, such as NER. This dual-purpose approach would ultimately enable more efficient semi-automated entity recognition in future investigations.

6 Conclusion and Recommendations

Criminal investigators face significant challenges with information overload, where manual entity annotation remains a resource-intensive task prone to human error, yet crucial for knowledge discovery in investigations.

Our approach has demonstrated the potential of leveraging an LLM for generating synthetic training data to fine-tune a case-specific NER model. Our work has resulted in a shared synthetic dataset that builds upon the PIVCOLF ontology, enabling researchers and law enforcement to fine-tune models within the domain of Norwegian criminal investigations.

Our method achieved a fivefold increase with respect to time compared to manual annotation while maintaining promising performance with weighted average F1-scores of 94.2% on the *Annotated Synthetic* dataset and 81.6% on the *Annotated Confidential* dataset.

Our findings suggest that criminal investigators can implement this method to gain computational support for information categorization in major criminal investigations. While the performance drop between synthetic and confidential data indicates a gap in linguistic patterns, this presents clear directions for future research. Specifically, we propose developing a hybrid dataset that combines LLM-generated content with investigator annotations, potentially bridging the current performance gap and improving the model's effectiveness on real data.

We motivate investigators to start performing their categorization of information in a format suitable for machine learning, thus enabling semi-automated entity categorization in future investigations.

7 Limitations

While our approach shows promise, several limitations should be noted. We have chosen to apply what we call 'Space-based tokenization' for BIO annotations. This might have impacted the results. However, this way is fast and interpretable, thus we see this as an applicable way of annotating for criminal investigations. Our synthetic dataset, though useful, shows performance differences when applied to real cases, indicating room for improvement in data generation quality. Another limitation of our research is that we did not experiment with different LLM parameters nor assess their effects when generating the synthetic dataset and annotations. Additionally, further experiments with fine-tuning different models using the same dataset could have better demonstrated the approach's generalization. We also acknowledge that evaluating the representativeness of our synthetic data could have provided valuable insights into its quality. While acknowledging these constraints, our approach shows promising results, and we will continue to refine and improve this work in future research and development.

8 Ethics and Data Protection

The Norwegian Director of Public Prosecutions and The Norwegian National Police Directorate have permitted access to data for this study. All confidential case data remained within secured police servers throughout the study, adhering to data protection protocols. No confidential data was transferred to external services or systems.

The confidential data served as inspiration for generating synthetic data that mimicked the style and content, without revealing any actual case information. The manually annotated confidential data for evaluation remained within its secured police server. The synthetic dataset and fine-tuned model are available for research purposes and transparency.

Acknowledgments

This project is financed by the Norwegian Research Council (project number 333898), Kripos, and NTNU. Thanks to the Center for Information Resilience (CIRES) at the University of Queensland and to Dr. Lei Han for your initial exploration and work. Thanks to The Norwegian Police IT Unit for technical facilitation and colleagues at Kripos for valuable feedback.

References

[1] Rabeah Al-Zaidy, Benjamin C.M. Fung, Amr M. Youssef, and Francis Fortin. 2012. Mining criminal networks from unstructured text documents. *Digital Investigation* 8, 3-4 (Feb. 2012), 147–160. https://doi.org/10.1016/j.diin.2011.12.001

[2] Carlo Batini, Valerio Bellandi, Paolo Ceravolo, Federico Moiraghi, Matteo Palmonari, and Stefano Siccardi. 2021. Semantic Data Integration for Investigations: Lessons Learned and Open Challenges. In *2021 IEEE International Conference on Smart Data Services (SMDS)*. 173–183. https://doi.org/10.1109/SMDS53860.2021.00031

[3] M Bruckschen, Caio Northfleet, D Silva, Paulo Bridi, Roger Granada, Renata Vieira, Prasad Rao, and Tomas Sander. 2010. *Named entity recognition in the legal domain for ontology population.*

[4] Eoghan Casey, Sean Barnum, Ryan Griffith, Jonathan Snyder, Harm van Beek, and Alex Nelson. 2017. Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language. *Digital investigation* 22 (2017), 14–45.

[5] Michael Chau, Jennifer J Xu, and Hsinchun Chen. 2002. Extracting Meaningful Entities from Police Narrative Reports. (2002), 5.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. http://arxiv.org/abs/1810.04805 arXiv:1810.04805 [cs].

[7] Alaa El-Halees and Nael Elyezjy. 2015. Investigating Crimes using Text Mining and Network Analysis. *International Journal of Computer Applications* 126 (Sept. 2015), 19–25. https://doi.org/10.5120/ijca2015906134

[8] Hans Henseler and Harm van Beek. 2023. ChatGPT as a Copilot for Investigating Digital Evidence.. In *LegalAIIA@ ICAIL*. 58–69.

[9] Rasmus Hvingelby, Amalie Brogaard Pauli, Maria Barrett, Christina Rosted, Lasse Malm Lidegaard, and Anders Søgaard. 2020. DaNE: A Named Entity Resource for Danish. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (Eds.). European Language Resources Association, Marseille, France, 4597–4604. https://aclanthology.org/2020.lrec-1.565/

[10] Martin Innes. 2003. *Investigating murder : detective work and the police response to criminal homicide.* Oxford University Press, Oxford ; New York.

[11] Farkhund Iqbal, Benjamin CM Fung, Mourad Debbabi, Rabia Batool, and Andrew Marrington. 2019. Wordnet-based criminal networks mining for cybercrime investigation. *Ieee Access* 7 (2019), 22740–22755.

[12] Fredrik Jørgensen, Tobias Aasmoe, Anne-Stine Ruud Husevåg, Lilja Øvrelid, and Erik Velldal. 2020. NorNE: Annotating Named Entities for Norwegian. arXiv:1911.12146 [cs.CL] https://arxiv.org/abs/1911.12146

[13] Mayank Kejriwal, Pedro Szekely, and Craig Knoblock. 2018. Investigative Knowledge Discovery for Combating Illicit Activities. *IEEE Intelligent Systems* 33, 1 (Jan. 2018), 53–63. https://doi.org/10.1109/MIS.2018.111144556 Conference Name: IEEE Intelligent Systems.

[14] Per E Kummervold, Javier De la Rosa, Freddy Wetjen, and Svein Arne Brygfjeld. 2021. Operationalizing a National Digital Library: The Case for a Norwegian Transformer Model. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*. Linköping University Electronic Press, Sweden, Reykjavik, Iceland (Online), 20–29. https://aclanthology.org/2021.nodalida-main.3

[15] Al Louis and Andries P Engelbrecht. 2011. Unsupervised discovery of relations for analysis of textual data. *digital investigation* 7, 3-4 (2011), 154–171.

[16] Alf Bernt Nissen. 2018. Informasjonsbehandling i store etterforskningssaker. (2018).

[17] Alf Bernt Nissen. 2019. Politiets system for Operativ kriminalanalyse.

[18] Homicide Working Group NPCC. 2021. Major Incident Room Standardised Administrative Procedures (MIRSAP 2021). (2021).

[19] Vitor Oliveira, Gabriel Nogueira, Thiago Faleiros, and Ricardo Marcacini. 2024. Combining prompt-based language models and weak supervision for labeling named entity recognition on legal documents. *Artificial Intelligence and Law* (2024), 1–21.

[20] Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual Name Tagging and Linking for 282 Languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, Vancouver, Canada, 1946–1958. https://doi.org/10.18653/v1/P17-1178

[21] Yerin Park, Ro Seop Park, and Hansoo Kim. 2024. Key Information Extraction for Crime Investigation by Hybrid Classification Model. *Electronics* 13, 8 (2024), 1525.

[22] Mark Pollitt. 2013. The hermeneutics of the hard drive: Using narratology, natural language processing, and knowledge management to improve the effectiveness of the digital forensic process. (2013).

[23] Fillipe Barros Rodrigues, William Ferreira Giozza, Robson de Oliveira Albuquerque, and Luis Javier García Villalba. 2022. Natural Language Processing Applied to Forensics Information Extraction With Transformers and Graph Visualization. *IEEE Transactions on Computational Social Systems* (2022), 1–17. https://doi.org/10.1109/TCSS.2022.3159677 Conference Name: IEEE Transactions on Computational Social Systems.

[24] Marijn Schraagen, Matthieu Brinkhuis, Floris Bex, M P Schraagen, M J S Brinkhuis, and F J Bex. 2017. Evaluation of Named Entity Recognition in Dutch online criminal complaints. (2017).

[25] Mads Skipanes, Katrin Franke, Gianluca Demartini, and Alf Bernt Nissen. 2025. Information Analysis in Criminal Investigations: Methods, Challenges, and Computational Opportunities Processing Unstructured Text. *Policing: A journal of Policy and Practise.* (2025).

[26] Mads Skipanes, Tollef Emil Jørgensen, and Katrin Franke. 2023. Advancing Knowledge Discoveries in Criminal Investigations with Semantic Textual Similarity. (2023).

[27] Maarten van Banerveld, Nhien-An Le-Khac, and M-Tahar Kechadi. 2014. Performance Evaluation of a Natural Language Processing Approach Applied in White Collar Crime Investigation. In *Future Data and Security Engineering*, Tran Khanh Dang, Roland Wagner, Erich Neuhold, Makoto Takizawa, Josef Küng, and Nam Thoai (Eds.). Vol. 8860. Springer International Publishing, Cham, 29–43. https://doi.org/10.1007/978-3-319-12778-1_3 Series Title: Lecture Notes in Computer Science.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[29] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2020. Automated concatenation of embeddings for structured prediction. *arXiv preprint arXiv:2010.05006* (2020).

[30] Xintao Xing and Peng Chen. 2024. Entity Extraction of Key Elements in 110 Police Reports Based on Large Language Models. *Applied Sciences* 14, 17 (2024), 7819.

[31] Min Yang and Kam-Pui Chow. 2015. An Information Extraction Framework for Digital Forensic Investigations. In *Advances in Digital Forensics XI*, Gilbert Peterson and Sujeet Shenoi (Eds.). Vol. 462. Springer International Publishing, Cham, 61–76. https://doi.org/10.1007/978-3-319-24123-4_4 Series Title: IFIP Advances in Information and Communication Technology.

[32] Anne Mosvold Ørke. 2023. Giving the Police a Head Start: Norwegian Named Entity Recognition Dataset and Model Development for Investigative Purposes. (2023).

Received 4 November 2024

Concealing targeted attacks on the TLSH similarity digest scheme

Gábor Fuchs
CrySyS Lab
Budapest University of Technology and Economics
Budapest, Hungary
gfuchs@crysys.hu

Tamás Fülöp
CrySyS Lab
Budapest University of Technology and Economics
Budapest, Hungary
tfulop@crysys.hu

Roland Nagy
CrySyS Lab
Budapest University of Technology and Economics
Budapest, Hungary
rnagy@crysys.hu

Abstract

TLSH is one of the most widely used similarity digest schemes in the field of binary analysis. Its primary use cases in digital forensics and malware analysis require the scheme to be robust against attacks that create files designed to mislead it in evaluating their similarity to other files. Recently a set of practical targeted attacks were proposed against the scheme. One of these attacks could modify existing binaries in ways to make the scheme find them very different from files they were (and still are) similar to, while another attack achieved the scheme finding the modified binaries very similar to files, which they are actually very different from. Both attacks achieved their goals by relatively small modifications consisting of injecting specially crafted byte sequences to different places in the binary. A weakness of this approach is that the user of the scheme can easily identify and remove the added pieces of data, and thus eliminate the attack. We propose a set of modifications to the attacks that disguise the added data as legitimate machine code, in a way that is much harder to detect, identify and remove. We evaluated our methods on a set of ARM malware samples, and found that around 70% to 80% of the samples can be successfully modified by these new versions of the previously mentioned attacks.

ACM Reference Format:
Gábor Fuchs, Tamás Fülöp, and Roland Nagy. 2025. Concealing targeted attacks on the TLSH similarity digest scheme. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712716.3712720

1 Introduction

A similarity digest scheme is a pair of algorithms, the first of which maps arbitrary length inputs (byte sequences, e.g. files) to small hashes called similarity digests, in a way that the second algorithm can reason about how similar two original inputs were, based on their corresponding similarity digests alone. This is usually achieved through the digests of similar inputs being similar in contrast to the behavior of cryptographic hash algorithms, where similar but non-identical inputs are mapped to completely different hashes. Similarity digest schemes are mostly used in digital forensics applications [1], spam filtering [2], malware clustering [6, 12], and malware detection [13].

DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712720

TLSH [7] (Trend Micro Locality Sensitive Hash) is one of the most widely used similarity digest schemes in the field of binary analysis, for example used by VirusTotal[1], a well known malware repository. TLSH proved to be more robust against random mutations introduced to files than its alternatives (notabliy Ssdeep [5] and Sdhash [11]). This is based on the observation that after introducing certain amounts of small changes to a files (possibly not even affecting their functional behavior), the alternatives failed to detect the similarity between the original and modified versions many times, when TLSH still succeeded, in a configuration where all the schemes were tuned to produce similar low false positive rates [8, 9]. However, [3] and its follow up, [4] showed that it is possible to make small modifications to a file to specially manipulate the TLSH algorithms. The authors propose two attacks in [4]. The first of the two attacks achieves that the scheme will find the original and modified versions of the data very different, while the second attack makes the scheme find the modified file very similar or identical to an arbitrary target, by creating a near or complete digest collision.

The research presented in [4] showed in detail, how the described methods can be used on software binaries of IoT devices, more specifically executable ELF (Executable and Linkable Format) binaries for the ARM architecture. The described framework removes parts of the executable file that are not required during execution (both inner regions and the tail of the file), and replaces them by specially crafted data that manipulates TLSH in the desired way. A notable weakness of this approach is that a more experienced user of TLSH can repeat the process of removing these non-required parts, now containing the data planted by the attacker, before similarity digest calculation, and thus eliminate the attack.

In this paper, we propose modifications to the attack framework to disguise the data introduced by the attacker as legitimate machine code in the binary, while still achieving the same kinds of manipulations to TLSH. First, we extend the component responsible for generating the previously mentioned specially crafted data with the ability of abiding by custom rules like generating only valid machine code instructions of a certain architecture, and define such a rule for the 32-bit ARM architecture. Second, we extend the component responsible for allocating the required space within the ELF binaries so that the inserted data will be loaded to the memory just like the legitimate instructions of the program (i.e. as part of a text segment). Finally, we evaluate our new concealed versions of the existing attacks achieved by using these two kinds of extensions together on a data set of ARM IoT malware samples. We note that we could similarly allocate space belonging to a data segment, and

[1] https://www.virustotal.com/ Last accessed: January 9, 2025

insert arbitrary bytes there without the additional challenge of generating valid instructions. We take on this challenge to show that the attacks can be efficient, even if they have to abide by certain syntactic requirements in their generated data, which might enable them to support other file formats.

The paper is organized as follows: Section 2 establishes the background required to discuss our additions to the attack framework, including overviews of the relevant details of the TLSH algorithms, the ELF file format, and the original attack framework itself. In Section 3, we discuss our modifications of the framework to be able to generate valid machine code instructions instead of arbitrary bytes; and in Section 4, we discuss our different strategies to obtain space in the existing executable file, where the generated instructions can be inserted to, in a way that they will be loaded to the memory along with the original parts of the file containing machine code, without actually changing the behavior of the program. Section 5 shows the results of our experiments carrying out the new concealed versions of the two existing attacks built upon the framework on samples from a malware data set, discussing how many samples enable the modifications required by each of the different strategies, and how much more generated machine code it takes to achieve the same desired effect on TLSH compared to when arbitrary bytes can be inserted. Finally, Section 6 concludes the paper.

2 Background

2.1 The TLSH algorithms

TLSH is short for Trendmicro Locality Sensitive Hash. As its name suggests, TLSH belongs to the family of similarity digest schemes called locality sensitive hashes. For this reason, the similarity digests produced by TLSH are usually referred to as TLSH hashes. The algorithm of TLSH calculating the similarity digest traverses the input data and extracts its so called 2-skip-3-grams, which are triads of almost subsequent bytes skipping at most 2 bytes in between. Each of these byte triads are mapped to a single byte hash value using the Pearson hash function [10] on the 4-byte input constructed from a salt value depending on the shape of the triad (i.e. the pattern of in between skipped bytes) and the values of the 3 extracted bytes. The algorithm counts the occurrences of each of the 256 different hash values throughout the file. With the analogy of bucket hashing, the elements of the resulting 256 long array are called *bucket counts*. The algorithm discards the second half of the array keeping only the first 128 bucket counts $b_0 \dots b_{127}$ corresponding to the Pearson hash values $0 \dots 127$. Apart from a single byte `checksum` field and another one byte field called `lvalue`, which represent the size of the input data on a logarithmic scale, all fields of the TLSH hash are calculated from these bucket counts. The algorithm calculates the q_1, q_2, and q_3 quartiles of the 128 values, which are the last values from the first three quarters of the sorted version of the array (i.e. $\hat{b}_{31}$, $\hat{b}_{63}$, and $\hat{b}_{95}$, where $\hat{b}_0$ would mark the smallest of the 128 kept bucket counts). `Q1ratio` and `Q2ratio` capture the ratios of the quartile values:

$$\texttt{Q1ratio} = \lfloor 100 \frac{q_1}{q_3} \rfloor \bmod 16$$

$$\texttt{Q2ratio} = \lfloor 100 \frac{q_2}{q_3} \rfloor \bmod 16,$$

and the so called `codes` capture the relations of the $b_0 \dots b_{127}$ bucket counts corresponding the Pearson hash values $0 \dots 127$ to the quartiles:

$$\texttt{code}_i = \begin{cases} 0 & \text{if } b_i \le q_1 \\ 1 & \text{if } q_1 < b_i \le q_2 \\ 2 & \text{if } q_2 < b_i \le q_3 \\ 3 & \text{if } q_3 < b_i \end{cases}$$

The second algorithm takes two TLSH hashes as input and outputs a so called difference score. Higher scores mean that the original pieces of input data the hashes represent were different, while scores close to zero mean that they were very similar. While there is a theoretical maximum difference score, the score is not purposefully limited. This is different from some other schemes that output a similarity score on a fixed scale like $0 \dots 100$, where usually higher values mean more similar inputs. The difference score is calculated as a sum of smaller scores calculated for all the different hash fields.

2.2 ELF binaries

Executable and Linkable Format (ELF) is the binary format used on Linux systems. Among others, ELF is used for executable files, shared libraries and kernel modules on many different architectures. Every ELF binary starts with the so called ELF header, which, along with describing the more specific type of the file, optionally addresses two header tables: the program header table and the section header table. Both are called header tables, as they are arrays of equal sized header entries called program headers and section headers respectively.

Program headers are required in an executable file, as they define the initial memory image of the program: among others, what regions of the file are to be loaded to what regions of memory. These different regions of memory are called segments. Program header entries contain the following fields:

- `type` - the type of the described segment;
- `offset` - the offset of the data within the file relative to the beginning of the file;
- `vaddr` - the memory address where the segment is loaded;
- `paddr` - physical memory address, not used for simple user space binaries;
- `filesz` - the size of data to be loaded from the file to (the beginning) of the segment;
- `memsz` - the size of the segment within the memory (filled with zeros after the first `filesz` bytes);
- `flags` - bits describing whether the memory region of the segment should be readable (`R`), writable (`W`) and or executable (`X`); and
- `align` - alignment of the segment, a power of 2, where

$$\texttt{offset} \equiv \texttt{vaddr} \pmod{\texttt{align}}.$$

The most basic and important segment type is `LOAD`, which instructs the loader to load a piece of the file to a given range within the memory. The alignment for `LOAD` entries is usually the memory page size of the given architecture, but can be larger as well. On architectures like ARM, where this is relevant, multiple page sizes can be supported by setting the largest as alignment for the `LOAD` entries. Two other segment types relevant in our paper are the

GNU_STACK and GNU_RELRO. Compilers like GCC often add these entries to the program header table, even when not strictly required.

GNU_STACK determines the flags associated with the stack (read, write and execute), and GNU_RELRO makes an originally writable part of the memory read-only after certain loading steps. If the GNU_STACK entry is not present in the program header table, the permissions default to RW (read and write), so an entry with these same flags contains no additional information and can, therefore, be freely overwritten. GNU_RELRO is a security measure, and overwriting it does not disturb the functionality of the program.

Section headers are used to address parts of the entire file including any other kind of metadata (e.g. debugging symbols, linkage information) that is not actually used during loading and execution of the file. The entire section header table is optional in executable files, and can be removed by setting the fields referencing it to zero in the ELF header without affecting the behavior of the program.

2.3 The TLSH attack framework

The attack framework originally described in [4], is composed of multiple types of modules with different responsibilities, with the idea that modules covering the same responsibilities are interchangeable to the support of different combinations of file formats, attack goals, and strategies.

The first type of module provides support for a file format. This kind of module takes the original form of the input file and modifies it to an equivalent version that has spaces within and/or at the end of the file that can be overwritten with arbitrary data without affecting the functionality of the file. This can be a simple marking of unused padding areas within the file, or marking the possibility that the given file format is not corrupted by appending arbitrary data to the end. However, it might also entail some editing of headers within the file to remove references to some optional metadata that now can be overwritten, or to allocate some kind of new region to write into. The framework contains an implementation supporting ELF binaries, which removes the reference to section header table if there is one, and strips the file keeping only its ELF header, program headers, and the content referenced by the program headers, marking all the regions in between and at the end available for overwriting.

The next type of module calculates the so called *bucket count goals*, that are target ranges for the 256 bucket counts with optional lower and higher bounds. The used algorithm is specific to what we would like to achieve regarding the bucket counts. The framework contains two attacks named anti-blacklisting and anti-whitelisting. The anti-blacklisting attack aims to achieve a large difference score from a given TLSH hash (e.g. the hash of the original version of the file), through the difference score gained from the `Q1ratio` and `Q2ratio` fields alone, by increasing the q_3 quartile, while leaving q_1 and q_2 unchanged. The anti-whitelisting attack aims to replicate the values of hash fields of a given target hash, thus creating a hash collision. First, the bucket count dependent fields are set up with appropriate bucket count goals. The `checksum` and `lvalue` fields are handled separately at the end of the entire patching process as an optional feature. Along with any parameters specific to the attack, the input to this step contains the existing bucket count contributions of the regions of data that is kept from the original file, calculated by the framework, as these serve as starting values that can only be increased later by the contributions of the data inserted by the later steps.

Finally, the so called *generator* type module actually generates the data to be inserted to the previously allocated and marked regions that manipulates the TLSH bucket counts to fit in the corresponding target ranges defined by the bucket count goals. The input to this step contains the layout and sizes of the modifiable regions and the bucket count goals to be achieved. The framework has two different generator modules: the greedy generator and the periodic pattern generator. We only discuss and work with the greedy generator, as it proved to be much more efficient in all tested scenarios, while being much simpler at the same time than its alternative.

The greedy generator algorithm selects the inserted bytes one by one trying all 256 possible values for each. The addition of each byte is considered to contribute to the bucket counts through the byte triads they complete, thus appending a byte to the end of the file increases the bucket counts by 6 in total (as it completes 6 new byte triads), while deciding the last missing byte when filling a hole in the middle of the file increases the bucket counts by 18 in total (as it completes 18 different byte triads). A possible byte value is disqualified if its contributions increase one or more bucket counts above their allowed maximum value (if there is an upper bound for the given bucket count in the bucket count goals). The remaining byte values are scored in heuristic scoring system, which rewards contributions to bucket counts that have minimum values defined in the bucket count goals, and have not yet reached that minimum value. The algorithm sorts the byte values by their score, selects the top scoring one, and moves on to the next byte to be selected. If the algorithm runs out of possible values to choose for a given byte, it does backtracking by stepping back to the previously selected byte, discarding its previously selected top scoring value, and selecting the next best option. With this backtracking implementation, the algorithm only fails, if it runs out of options for the very first byte. If the space to be written to is fixed, the algorithm stops, when it has finished generating every byte, and considered to succeed or fail depending on whether it succeeded to achieve the bucket count goals or not. If the space is unlimited and the algorithm can write any amount of data, it stops when all bucket counts having minimum target values have reached those minimum target values.

3 Generating machine code

As an extension of the greedy algorithm, we have added a feature that allows it to take additional rules into account during the generation process. These rules are checked by the algorithm for each value for each byte, and the algorithm disqualifies the values that break any of them, just like it disqualified values with undesired contributions.

In our modified attack, the algorithm must only generate valid 32-bit ARM instructions. The 32-bit ARM instructions are always four bytes in length and are split into fields that can denote registers, conditions, offsets and flags, among a few others. These fields and the rules associated with them are all determined by that instruction's type (Data Processing, Multiply, Single Data Transfer, etc.). A common example is that a four bit field denoting a register cannot be the value of 15.

We used the disassemblers of radare2[2] and pwntools[3] to verify that our program was correctly identifying the generated bytes. Throughout this process we found that these two disassemblers disagree on the validity of certain instructions, that is, radare2 considers some of them valid while pwntools considers them invalid, and vice versa. Our main goal is making sure that most disassemblers such as the aforementioned two see all of our generated four byte sequences as valid instructions, so we only enable the generation of instructions that were considered valid by both tools. In the end, of the 2^{32} possible combinations, a little over 70% are considered valid ARM32 instructions and are allowed to be generated by the algorithm.

The algorithm of the check logic receives the chosen value for the current byte, its position in the file, and it can also see which of the other bytes have been decided (i.e. kept from the original file or already decided by the generator), and what their decided values are. The rule for valid ARM instructions returns true (meaning no contradiction), if the four byte instruction the checked byte belongs to (determined using the position of the byte in the file) can still be a valid instruction, with some of the bytes potentially not yet decided, and returns false if all the bytes of the instruction are decided, and form an invalid instruction, or the decided bytes already contradict (i.e. cannot be completed to form a valid instruction with any choice of values for the undecided bytes).

While generating the bytes, the algorithm keeps track of which instruction types are compatible with the already decided parts and which ones are not, so only the rules of those remaining types have to be checked. The instruction is invalid if at least one rule of each type is violated. The algorithm is still able to backtrack if required.

Since each instruction is generated individually, it is worth noting that while the instructions will make sense on their own, they will almost never result in a cohesive program when combined.

4 Obtaining executable space

To disguise our data inserted into the file as machine code, we need to add it to a region in the file that is loaded to the memory as an executable segment. This way, the generated bytes can no longer be removed by an aggressive stripping approach because they are now referenced in the program header table. We have two strategies to achieve this: extending an existing executable segment, or adding a new one.

4.1 Extending an existing segment

Most executable ELF files contain a single executable `LOAD` segment, so it is ideal to insert our generated code into the existing one if possible, which can be achieved by increasing its size and appending to its original content.

In the case that the executable `LOAD` segment ends before the next segment begins, that is,

$$\texttt{offset}_1 + \texttt{filesz}_1 < \texttt{offset}_2$$

the space between the two called *slack space* usually contains nothing but zeros in the file. This slack space can be overwritten and sometimes is actually enough to achieve significant changes in the bucket count values. This way the file does not need to be made any longer and only the size (`filesz` and `memsz`) of the executable `LOAD` entry needs to be changed in the program header table.

If required, the segment can be further expanded by inserting additional bytes and shifting the rest of the file. Along with the size of the executable `LOAD` entry, in this case, the offsets of the subsequent `LOAD` entries need to be adjusted as well. We need to abide by the rules of alignment. `LOAD` entries in ARM binaries tend to have a few different alignment values, the most common and smallest of which is 0x1000. While there are some larger page sizes in use, to support them the alignment would need to be at least 0x10000, which is uncommon among the binaries, so we decided to set the alignment of all `LOAD` entries to 0x1000. With 0x1000 as the alignment, we can extend the segment by multiples of 4096 bytes. While not as wasteful as the higher values, this restriction still means that we will likely end up inserting more space than actually required. For example, if our attack requires 9000 bytes and there is no gap between the entries inside the file then we need to insert three 4096 byte pages to reach our goal, 3288 bytes more than what is necessary.

The total amount we can insert is also limited. The fact that enables such an insertion in the first place is that the two `LOAD` segments tend to be quite far away from each other inside the memory – usually the distance between their virtual addresses is far greater than the distance of their offsets within the file. Formally, the required gap is present if

$$\texttt{vaddr}_1 + \texttt{memsz}_1 < \texttt{vaddr}_2.$$

The size of this gap can vary and, while sometimes it is not even existent, it is usually quite significant: most commonly larger than the gap in the file by 0x8000 in our samples, which is more than enough for the majority of modifications required by the attacks.

4.2 Adding a new segment

It is also possible to add a new segment to the file if the existing executable segment cannot be extended by the required amount. A new `LOAD` entry needs to be associated with this segment, and since we do not want to increase the size of the program header table (such a modification would be hard without breaking the alignments and relative references), an existing entry should be repurposed.

As mentioned in Section 2.2, there are some types of program header entries that can sometimes be removed without affecting the behavior of the program. The two entries we are looking to replace are the `GNU_STACK` and `GNU_RELRO`. These two types of entries tend to be towards the end of the program header table, while the `LOAD` entries need to be right after each other in the order of their `vaddr`. For this reason, after removing the entry to be replaced from its original position, we simply add our new entry to its correct position in the table, thus shifting the entries in between the two positions. This rearrangement does not affect the functionality of the program.

The new segment is written to the end of the file, and it can be as long as it needs to be. Within the memory it can be placed to any unused region of the virtual address space with the correct alignment. As we are using the acquired space for ARM instructions, practically we will set a multiple of four bytes as its size.

[2] https://github.com/radareorg/radare2 Last accessed: August 12, 2024
[3] https://github.com/Gallopsled/pwntools Last accessed: August 12, 2024

5 Evaluation

We tested our two approaches, both extending an existing segment and adding a new one, on a collection of 2000 randomly selected ARM malware samples from the CUBE-MALIOT-2021[4] data set, with both the anti-blacklisting and anti-whitelisting attacks. For the anti-blacklisting attack we chose the target score of 86, a similarity detection threshold proposed by the authors of TLSH specifically for executable ELF files in [8]. For the anti-whitelisting attack, we used the TLSH of the GCC binary as target hash, without the optional feature of dealing with the `checksum` and `lvalue` fields. So we conducted four tests on these samples in total. Along with the performance of these entire attack configurations, we separately evaluate the overhead in required space caused by generating valid ARM instructions instead of arbitrary bytes.

5.1 Attacks with the segment extending method

Let's discuss the results of the anti-blacklisting, segment extending attack first. Below, we refer to a block of 0x1000 (4096) bytes as a *page*. Out of the 2000 total tested binaries, the attack succeeded to surpass the target TLSH difference score of 86 in 1679 (84.0%) cases. The maximum space available is the combined size of the gap found between the two segments inside the file and the amount we inserted. From the 1679 samples we were able to patch, 516 needed no insertion of additional pages whatsoever, 1157 needed one page, 5 needed three pages, and 1 needed four. The remaining 321 samples could not be modified due to overlapping segments (150), unconventional headers (91), or insufficient available space (80).

The anti-whitelisting, segment extending attack was successful in 1451 cases (72.6%). This version of the attack requires significantly more space since it is creating a near complete hash collision. This time all files required some insertion of anywhere from one to eight additional pages, but 1327 (over 90%) of them required two (585), three (585) or four (157) pages only. The remaining 549 samples either had insufficient available space (308) or, as previously mentioned, had overlapping segments (150) or unconventional headers (91).

The data points of Figure 1 consist of the successfully patched samples of these two attacks, where the X-axis shows the size of the truncated files, and the Y-axis shows the amount of bytes used for the modification. Due to the size of inserted pages being fixed at 0x1000 bytes, dense clusters of samples are located just above multiples of 0x1000 on the Y-axis. Most of the samples in the anti-blacklisting attack required one page to be inserted, so only one such dense cluster exists. The samples below the 4 KiB line needed no insertion and were modified solely by overwriting the gap already present between the two `LOAD` segments in the file. On the anti-whitelisting graph, the clusters caused by the page insertions are more clearly visible. One can also observe that the attack required more and more pages to be inserted as the sizes of the files increased.

5.2 Attacks with the segment adding method

The approach of adding a new segment to the file was not as successful among these 2000 samples. Of the 1919 files containing a `GNU_STACK` entry 1889 of them had the non-default executable flag, meaning they cannot be overwritten if we want to preserve the functionality of the program. The `GNU_RELRO` entry was much rarer with only 28 samples having one. These two groups did not overlap and one program had a different problem within its headers, so in the end, only 57 files could be modified using this method.

In the case of these 57 samples, the anti-blacklisting version required an approximately 0.5% increase from the truncated file on average, the minimum and maximum being 0.46% and 0.63% respectively. The anti-whitelisting attack necessitated almost 20 times that amount, with the average at 9.6%, the minimum at 8.4% and the maximum at 11.3%. Since we do not have to add entire pages at a time, this approach tends to require less space than its segment extending counterpart.

5.3 The overhead of generating valid instructions

Finally, let's inspect the overhead in required space caused by adding the rule of generating only valid 32-bit ARM instructions to the greedy algorithm. This rule is not a negligible restriction, as previously discussed, only around 70% of the possible four byte combinations are accepted. This means that our modifications are expected to require more space, since the optimal choices for byte values for manipulating TLSH may violate the rule. We used the approach of adding a new segment to evaluate the difference in spacial efficiency, since in the results of the segment extending approach it would be hard to observe the expected small difference due to the fact that entire pages are allocated.

As expected, the anti-blacklisting attack required only slightly more space, with a minimum increase of 1.0%, a maximum of 3.8%, and an average of 2.4% among all applicable samples. The anti-whitelisting attack required significantly more space in general, and with these longer modifications, the difference caused by the rule seems to further diminish. In this case, the minimum increase was 0.4%, and the maximum 1.8%, with an overall average increase of 1.3%.

6 Conclusion

In this paper we presented some extensions to an attack framework against the TLSH similarity digest scheme and tested the framework with our extensions on 2000 ARM malware samples. The framework, originally described in [4], contained existing attacks against TLSH that could modify executable ELF binaries to manipulate their TLSH hashes in different desired ways. This entailed introducing specially crafted data into the files, with the weakness that users of TLSH could rather easily identify and remove these added pieces of data, and thus eliminate the attack. To address this weakness, our extensions made it possible to disguise the added data as legitimate machine code.

First, we adjusted the algorithm generating the added data to take and abide by optional rules like to generate valid machine code instructions of a certain architecture instead of arbitrary sequences of bytes, while still making the data manipulate TLSH in the ways required by the attacks. With the rule of generating valid ARM instructions, we observed that the generator needed only a few

[4] https://github.com/CrySyS/cube-maliot-2021 Last accessed: August 12, 2024

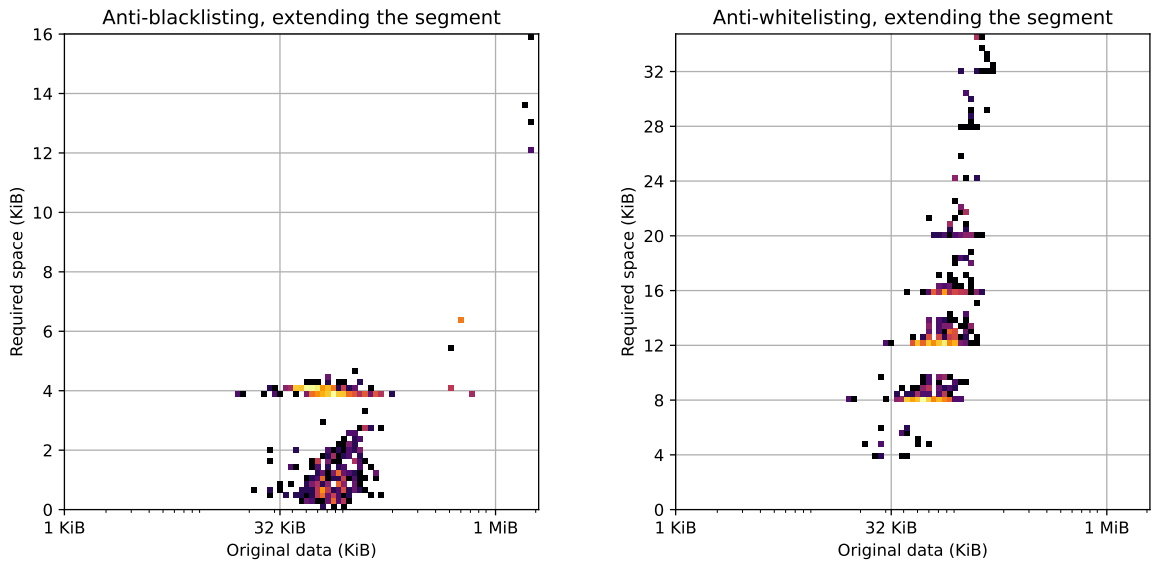


Figure 1: Required space of the successful segment extending attacks in relation to the size of the truncated file. The required space consists of the bytes overwritten between the two LOAD segments plus the size of the inserted pages. The graphs are heat maps visualizing distribution: brighter colors represent more samples, each color represents an interval on a logarithmic scale.

percents more space than without the rule (2.4% and 1.3% more on average in two different attack scenarios).

Second, we proposed two strategies to obtain space in the ELF binaries that is handled as the parts of the binary containing its original machine code: being loaded to the memory as an executable segment. The first strategy tries to extend an existing executable segment, while the second tries to create a new segment in the existing binary, both without affecting the functionality of the binary. The strategy of extending an existing segment was applicable for 1759 samples, being able to provide enough space for an attack guaranteeing a TLSH difference score of 86 from the original version of the binary in 1679 (84.0%) of the cases, and for an attack modifying the binary to have nearly the same TLSH hash as the binary of a version of the GCC C compiler in 1451 (72.6%) of the cases, while generating valid ARM instructions. The strategy of creating a new segment required an entry in the program header table of the ELF file that could be removed without affecting functionality. We could identify such entries in only 57 (2.9%) of the samples, and as this method can provide unlimited space, both described attacks succeeded in all of the 57 cases.

In our disguised attacks, even though the inserted data (while consists of valid instructions) is not meaningful code, and the program can never reach it during execution, neither of these facts make the attacks easy to detect. It is not trivial how to detect a sequence of instructions as “not meaningful” and calculating whether certain instructions are reachable or not during execution is a known hard problem (CFG generation).

We note that it is quite easy to create new rules to the generation algorithm, for example to only generate printable ASCII characters. Judging by the minimal overhead caused by the rule of generating only valid ARM machine code instructions, usage of other such rules might be feasible, and could allow the methods to be used in different file formats.

We conclude that our extensions enable the attacks proposed in [4] against all applications of TLSH; both to be much harder to detect or defend against, when the scheme is used on (ELF) software binaries; as well as, to be easily adjusted to support additional file formats. This implies that decisions based on the TLSH scheme or TLSH hashes alone cannot be trusted if an attacker can modify any parts of the input, even if such parts have certain syntactic constraints.

Acknowledgments

The research presented in this paper was carried out in the context of the European Union project no. RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory and in the project no. 2023-1.1.1-PIACI_FÓKUSZ-2024-00030 funded by the National Research, Development and Innovation Office of Hungary.

References

[1] Donghoon Chang, Mohona Ghosh, Somitra Sanadhya, Monika Singh, and Douglas White. 2019. FbHash: A New Similarity Hashing Scheme for Digital Forensics. *Digital Investigation* 29 (07 2019), S113–S123. doi:10.1016/j.diin.2019.04.006

[2] Jianxing Chen, Romain Fontugne, Akira Kato, and Kensuke Fukuda. 2014. Clustering Spam Campaigns with Fuzzy Hashing. In *Proceedings of the Asian Internet Engineering Conference (AINTEC)*. 66–73. doi:10.1145/2684793.2684803

[3] Gábor Fuchs, Roland Nagy, and Levente Buttyán. 2023. A Practical Attack on the TLSH Similarity Digest Scheme. In *Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES '23)*. Association for Computing Machinery, New York, NY, USA, Article 13, 10 pages. doi:10.1145/3600160.3600173

[4] Gábor Fuchs, Roland Nagy, and Levente Buttyán. 2024. Targeted Attacks against the TLSH Similarity Digest Scheme. Under submission.

[5] J. Kornblum. 2006. Identifying Almost Identical Files Using Context Triggered Piecewise Hashing. In *Proceedings of the 6th Annual DFRWS Confenerce*.

[6] Yuping Li, Sathya Chandran Sundaramurthy, Alexandru G. Bardas, Xinming Ou, Doina Caragea, Xin Hu, and Jiyong Jang. 2015. Experimental Study of Fuzzy Hashing in Malware Clustering Analysis. In *Proceedings of the 8th USENIX Conference on Cyber Security Experimentation and Test (CSET)*. USENIX Association.

[7] Jonathan Oliver, Chun Cheng, and Yanggui Chen. 2013. TLSH – A Locality Sensitive Hash. In *2013 Fourth Cybercrime and Trustworthy Computing Workshop*. IEEE, Sydney NSW, Australia, 7–13. doi:10.1109/CTC.2013.9

[8] Jonathan Oliver, Scott Forman, and Chun Cheng. 2014. Using randomization to attack similarity digests. In *International Conference on Applications and Techniques in Information Security*. Springer, 199–210.

[9] Fabio Pagani, Matteo Dell'Amico, and Davide Balzarotti. 2018. Beyond Precision and Recall: Understanding Uses (and Misuses) of Similarity Hashes in Binary Analysis. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy* (Tempe, AZ, USA) *(CODASPY '18)*. Association for Computing Machinery, New York, NY, USA, 354–365. doi:10.1145/3176258.3176306

[10] Peter K Pearson. 1990. Fast hashing of variable-length text strings. *Commun. ACM* 33, 6 (1990), 677–680.

[11] V. Roussev. 2010. Data Fingerprinting with Similarity Digests. In *Research Advances in Digital Forensics VI*. 207–226.

[12] Nikolaos Sarantinos, Chafika Benzaïd, Omar Arabiat, and A. Al-Nemrat. 2016. Forensic Malware Analysis: The Value of Fuzzy Hashing Algorithms in Identifying Similarities. In *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 1782–1787. doi:10.1109/TrustCom.2016.0274

[13] Csongor Tamás, Dorottya Papp, and Levente Buttyán. 2021. SIMBIoTA: Similarity-based Malware Detection on IoT Devices. In *Proceedings of the 6th International Conference on Internet of Things, Big Data and Security (IoTBDS)*. INSTICC, SciTePress, 58–69. doi:10.5220/0010441500580069

Towards Interpretable Topic Modelling as a Tool for Hypothesis-Driven Forensic Communication Analysis

Jenny Felser
University of Applied Sciences Mittweida
Mittweida, Saxony, Germany
felser@hs-mittweida.de

Michael Spranger*
University of Applied Sciences Mittweida
Mittweida, Saxony, Germany
spranger@hs-mittweida.de

Abstract

Instant messages stored on mobile devices have become a crucial source of evidence in forensic investigations. However, the immense amount of messages proves to be a challenge for investigators, especially since only a fraction of the messages are relevant to the case. Usually, the analysis of this communication data is guided by a specific forensic hypothesis. Consequently, a method that dynamically reveals evidence for these forensic hypotheses and summarises the communication parts relevant to these hypotheses is urgently needed to provide the best possible support for the investigators. Therefore, this paper presents an approach that finds evidence for case-relevant topics associated with a current hypothesis by interactively integrating case knowledge in topic modelling. However, a problem of all popular topic modelling algorithms lies in interpreting the resulting (case-relevant) topics, as ranking words usually represent these without further contextual information. Therefore, the first experiments presented in this paper focused on improving the extracted topics' interpretability, whereby five different forms of representation were analysed based on a forensic state-of-the-art data set. A qualitative evaluation suggested that generating a topical summary using a large language model (LLM) facilitates topic interpretation.

CCS Concepts

• **Applied computing → Evidence collection, storage and analysis**.

Keywords

Forensic text analysis, hypothesis-driven, topic modelling, semi-supervised

ACM Reference Format:
Jenny Felser and Michael Spranger. 2025. Towards Interpretable Topic Modelling as a Tool for Hypothesis-Driven Forensic Communication Analysis. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3712716.3712721

DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712721

1 Introduction

In today's digital age, communication via messenger services is becoming increasingly popular. Most of the population uses this channel to communicate with friends, colleagues or family about everyday or personal topics. In addition to these harmless conversations, however, mobile communication is also used by criminals to plan, commit or carry out criminal offences [45]. Accordingly, messages stored on mobile devices are becoming an essential evidence source in forensic investigations.

The immense amount of mobile communication data that investigators regularly confront proves to be an increasing challenge, especially since experience shows that the forensically relevant content constitutes only a tiny fraction of the overall data and is overlaid by irrelevant small talk [45]. Manual analysis of this communication data seems inefficient since it cannot even be assumed with certainty that any case-relevant content was discussed.

The relevance of communication varies significantly for each case. It is determined by one or more hypotheses pursued by investigators, such as the existence of a particular criminal offence [53]. Investigators formulate these hypotheses based on the results of criminal investigations and their professional expertise about case-specific peculiarities, such as the milieu involved. Therefore, the primary purpose of automated procedures should be to support investigators in dynamically identifying evidence for such a hypothesis and summarising the relevant parts of the conversation regarding this hypothesis in an interpretative manner.

Popular methods such as pre-trained Large Language Models (LLMs) [39] do not do justice to the strong individuality of each case. By contrast, topic modelling is promising, which makes it possible to automatically extract the main ideas discussed in the communication data and provide them to investigators to verify suspicions. Traditional, unsupervised approaches, however, tend to extract the dominant topics and may miss rare, case-relevant topics. Instead, the topic modelling process should be guided by the case-specific prior knowledge of the investigators. Therefore, an important research goal is to use semi-supervised topic modelling to verify whether specific topics relevant to a hypothesis have been discussed in the communication data. In the remainder of this paper, these expected topics associated with a current hypothesis are called "case-relevant topics".

One problem, however, is that the extracted, possibly case-relevant topics are often difficult to interpret because traditional topic modelling algorithms represent them as a ranking of topic-related words [e.g. 6, 18], thus losing context information. A further fundamental research objective is increasing the interpretability of the (case-relevant) topics by identifying a suitable alternative representation.

This paper presents a semi-supervised, interpretable topic modelling pipeline to support hypothesis-driven forensic communication analysis. We also present our initial experimental results on the second research objective, topic interpretability. Five representations of an online grooming topic were compared qualitatively using the data set provided by [29].

The rest of the paper is structured as follows: After discussing the current state of the art in section 2, the proposed pipeline is presented in section 3. Subsequently, the experiments to increase topic interpretability and our results are described and discussed in section 4 and in section 5, respectively. Finally, we conclude in section 6.

2 Related Work

The most straightforward approach to finding case-relevant content is a simple keyword search [e.g. 4]. However, the success of this approach depends heavily on the quality of the search terms. Too general search terms return many irrelevant messages, which cannot be efficiently analysed manually [4]. More interesting are a few previous works that aimed to find evidence for a forensic hypothesis [e.g. 39] or a case-relevant topic based on communication data [e.g. 41], e.g. by separating the relevant from the irrelevant content using a classifier [e.g. 46] including fine-tuned LLMs [e.g. 39], or a clustering algorithm [41]. However, using pre-trained models raises concerns, as they do not adequately reflect the high heterogeneity of forensic communication data and the individual circumstances of each offence [44]. Moreover, although these methods enable prioritisation, a manual analysis is still required to obtain an overview of the content of the texts classified as relevant and to assess whether a forensic hypothesis is fulfilled.

For these reasons, methods of automated content analysis, such as topic modelling, for reducing communication to (relevant) main ideas, seem more desirable. Previous work has shown the potential of topic models for the exploratory analysis of forensic text data [e.g. 10], including communication data [e.g. 11, 32]. However, these approaches focused on using widespread unsupervised, probabilistic algorithms such as Latent Dirichlet Allocation [6] to analyse communication for which it could be assumed that a large proportion of it is relevant, e.g. chats from hacker forums [e.g. 47, 51] or tweets immediately after an event like a terrorist attack [42]. The problem of extracting case-relevant topics in communication data that are overlaid with small talk has been addressed occasionally by pre-filtering the data set based on simple keyword matching [e.g. 32], but that carries the risk of discarding case-relevant texts.

In contrast, only a few previous works in the forensic domain integrated case-specific prior knowledge into semi-supervised topic modelling algorithms to find evidence for a case-relevant topic that supports a specific forensic hypothesis [e.g. 21, 48, 56]. Specifically, so-called Seed-Guided Topic Models (SGTM) were used, which require only minimal user input, such as a few characteristic words for each expected case-relevant topic, denoted as seed terms [e.g. 18, 24]. The experiments highlighted that these algorithms successfully provide evidence for case-relevant topics such as drug trafficking [56] and fraud [48] or support for a terrorist organisation [20, 21].

Although these approaches are promising, the problem has been overlooked that the identified case-relevant topics described by the words with the highest probability [e.g. 18] are hard to interpret, as also emphasised by [34]. This issue arises, in addition to the lack of contextual information, from the fact that the presence of irrelevant, high-frequency terms among the most likely words for a topic is unavoidable, especially in noisy communication data, as emphasised by [17]. The most obvious strategy to improve interpretability may be to use documents tokenised into larger N-grams as input for the SGTM. However, doing so increases the vocabulary size, which, according to [12], can negatively impact the quality of the topics.

An alternative representation of topics that guarantees even more contextual information is to provide characteristic short texts, e.g. the most characteristic message of the data set for each topic, as proposed by [17]. As experiments demonstrated, this representation can increase the topic interpretability [28]. However, it should be noted that [17] did not extend an existing probabilistic topic model but developed a model known as exemplar-based topic modelling, which determines the characteristic short text based on the cosine similarity of the texts. This approach is unsuitable for hypothesis-driven topic modelling since it is entirely unsupervised. In contrast, the approach suggested by [9], which uses an SGTM and increases the interpretability of the extracted topics through explanation generated by LLM, seems more relevant for this work. Nevertheless, their pipeline for identifying epidemic-related topics in newspaper articles was developed in a setting that differs significantly from forensic instant messages in language and length. In particular, finding suitable words to describe the case-relevant topic in the context of a hypothesis is often challenging in forensic applications.

Including feedback for the iterative refinement of these topic-characteristic words can prove beneficial. This human-in-the-loop approach to iteratively finding evidence for case-relevant topics was taken up in the context of visual topic analysis systems [e.g. 23, 30]. In particular, the framework described by [16] is relevant to our work because it combines feedback and an improved representation of topics based on sentences with high topic coverage. However, this approach, developed for extracting domestic political topics from presidential debates, is unsuitable for forensic contexts. Although feedback is included, the framework does not allow topic modelling to be guided in a desired direction from the beginning. However, this is essential if the aim is to verify whether one particular case-relevant topic has been discussed.

In conclusion, no previous work sought to find evidence for case-relevant topics and simultaneously increase their interpretability. This paper proposes a hypothesis-driven topic modelling approach that interactively finds evidence for case-relevant topics with the investigator's involvement and can present them interpretably to close this research gap. A particular contribution of the presented approach is that, while the previous work has only qualitatively evaluated whether a specific topic was found and a hypothesis could be confirmed [e.g. 48, 56], here a way is proposed to capture the evidence for a hypothesis quantitatively. In addition, we compare several, including more advanced topic representations, rather than taking the most straightforward approach of selecting documents characteristic of the topic based on topic coverage [e.g. 3, 16].

3 Interpretable, hypothesis-driven topic modelling

The task of interpretable, hypothesis-driven topic modelling based on in the context of forensic communication data can be formally formulated as follows:

Definition 3.1 (Problem definition). Let M be the message set to be analysed in a forensic investigation, consisting of individual messages m_i with $i = 1...|M|$. The investigator pursues n hypotheses $\mathscr{H}_1^{(x)}, ..., \mathscr{H}_n^{(x)}$ based on the current state of the investigation x. Hypothesis-driven topic modelling aims to extract topics to be interpreted as evidence for or against a hypothesis. That comprises the two following subgoals:

(1) Firstly, the evidence $\mathscr{E}(\theta_j^{(x)}, \mathscr{H}_i^{(x)})$ for each hypothesis $\mathscr{H}_i^{(x)}$ by a topic $\theta_j^{(x)}$ is determined in the form of a quantitative score.
(2) Subsequently, characteristic short texts $K_j = \{k_{j_1}, ..., k_{j_n}\}$ are determined for each extracted topic $\theta_j^{(x)}$.

Therefore, an iterative process is proposed that integrates a representation of expected case-relevant topics associated with the hypothesis into the topic model. Evidence for a hypothesis derives from the agreement between the expected case-relevant topics and the topics extracted. In each iteration, the representation of the expected topic is improved by integrating the feedback from the investigator, to whom the topics are presented as characteristic short texts. The entire workflow is outlined in Figure 1 using the hypothesis that online grooming occurs in chats (see section 4).

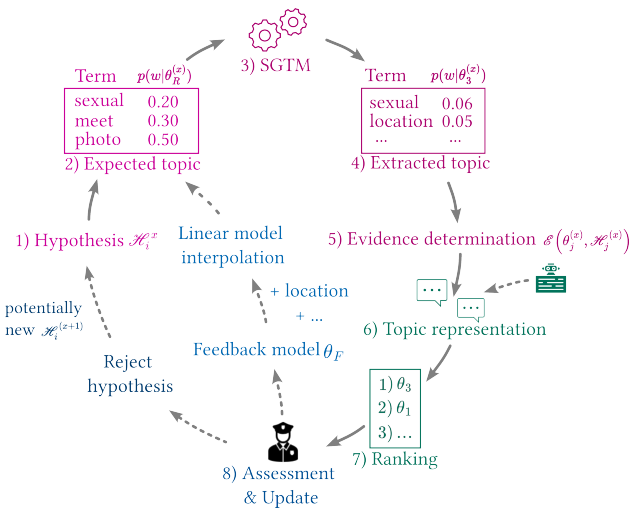


Figure 1: Procedure for interpretable, hypothesis-driven topic modelling based on forensic communication data.

Extraction of case-relevant topics: The basic idea is that a hypothesis (see Step 1 in Figure 1) is considered an expected case-relevant topic $\theta_{R_i}^{(x)}$ that the investigator assumes was discussed in the messages M. In the example shown in Figure 1, the topic "online grooming" would thus be expected. A probabilistic or neural SGTM [e.g. 18, 33] is used to verify whether this case-relevant topic is part of the communication, thus supporting the hypothesis. These SGTMs are guided by an appropriate representation of the case-relevant topic, which can also be an a priori distribution instead of a few characteristic words, as described in section 2. This a priori distribution is a probability distribution over relevant words defined by the investigator encouraging the SGTM to extract the case-relevant topics $\theta_{R_i}^{(x)}$, even if they have only been discussed to a small extent [e.g. 35] (Figure 1, Step 2). Semantically coherent conversations serve as input documents for the SGTM, for which the messages are clustered as described by [45], for example. This way, the short length of individual messages, a known problem for topic modelling, is addressed [e.g. 27].

Subsequently, the SGTM (step 3) extracts a user-defined number l of topics $\Theta^{(x)} = \{\theta_1^{(x)}, ..., \theta_l^{(x)}\}$ (Step 4). The extracted topics are then compared with the expected case-relevant topics, both represented as word probability distributions [e.g. 18, 33] (step 5). Specifically, the evidence $\mathscr{E}(\theta_j^{(x)}, \mathscr{H}_i^{(x)})$ is defined as the Kullback-Leibler divergence between the extracted topic $\theta_j^{(x)}$ and the case-relevant topic $\theta_{R_i}^{(x)}$ representing the hypothesis $\mathscr{H}_i^{(x)}$ (see Equation 1).

$$\mathscr{E}\left(\theta_j^{(x)}, \mathscr{H}_i^{(x)}\right) = D\left(\theta_j^{(x)} || \theta_{R_i}^{(x)}\right) \tag{1}$$

The topics can then be sorted according to their evidence for the hypothesis $\mathscr{H}_i^{(x)}$ (step 7).

Improvement of topic representation: The extracted topics $\Theta^{(x)}$ must be sufficiently interpretable to allow the investigators to assess whether the highly ranked topics reflect the case-relevant topic $\theta_{R_i}^{(x)}$, they expect. For this purpose, the interpretability of the topics is improved by presenting them as characteristic short texts $K_j = \{k_{j_1}, ..., k_{j_n}\}$ (step 6).

Analogous to text summarisation methods, the approaches for determining the topic-characteristic short texts can be divided into extractive and generative [25]: An extractive method is understood here to mean that the messages of the forensic communication data set are ranked according to their relevance to the topic and the n most characteristic messages are presented. It should be noted that since conversations are used as input documents, the SGTM does not directly provide the topic coverage of individual messages. Instead, the importance of a topic for a message can be inferred from the topic probability of its words using the "summation over words" procedure [e.g. 31], or the similarity between the probable words of the topic and each message can be calculated [50].

Concerning generative approaches, i.e. the generation of new, topic-characteristic short texts that are not present in the communication data set [25], LLMs, in particular, appear promising. Those have already been successfully used to summarise content predicted as forensically relevant by supervised classifiers [39, 49]. In the context of hypothesis-driven topic modelling, in addition to generating a thematic summary, generating an artificial characteristic message, for example, can also be considered.

In any case, these improved representations of the topic confirm or refute the investigator's expectations regarding the case-relevant topic and enable him to support, discard or refine his hypothesis.

Incorporation of feedback: An essential prerequisite for the SGTM to be able to extract the expected, case-relevant topics is that the a priori distribution is of sufficient quality [e.g. 18, 54]. The words in the a priori distribution should sufficiently represent the topic for the hypothesis and be present in the examined communication data set with a suitably high frequency. Since these conditions are not guaranteed per se, the a priori distribution is improved iteratively in a human-in-the-loop process, as proposed by [30] for document retrieval, for example (step 8).

Specifically, the assessment of the ranking of topics, represented by short texts, by the investigators can result in two possible scenarios: On the one hand, the investigator can discard the hypothesis $\mathscr{H}_i^{(x)}$ and possibly replace it with a new hypothesis $\mathscr{H}_i^{(x+1)}$ if none of the highly ranked topics reflects his expected case-relevant topic. On the other hand, he could see one of the highly ranked topics as a confirmation of his expected topic and, by describing this topic using the characteristic short texts K, find new terms that represent the current hypothesis even better. In the second case, more meaningful terms can be used as a feedback model θ_F to modify the original hypothesis representation $\theta_{R_i}^{(x)}$. The update of the hypothesis representation is done using a linear model interpolation, as initially proposed for the information retrieval by [1] (see Equation 2). Here, α represents the interpolation parameter that controls the adoption of the feedback model θ_F into the new hypothesis representation $\hat{\theta}_{R_i}^{(x)}$.

$$\hat{\theta}_{R_i}^{(x)} = (1-\alpha)\theta_{R_i}^{(x)} + \alpha\theta_F \tag{2}$$

With the updated hypothesis representation $\hat{\theta}_{R_i}^{(x)}$, topics are extracted again so that the investigator finds more evidence for or against the hypothesis in each iteration.

4 Preliminary experiments

In the following, initial experiments focus on increasing the interpretability of topics. The online grooming scenario illustrated in Figure 1 was chosen as the use case. Accordingly, the hypothesis was examined that sex offenders were trying to make contact with minors in the form of online grooming in the analysed chats [52]. Five different forms of representation of a grooming topic extracted with an SGTM were qualitatively evaluated: traditional descriptions based on the most probable words or bigrams and one extractive and two generative approaches for topic-characteristic short texts.

4.1 Data

Due to a lack of publicly available forensic data from mobile communications, posts from social media, which resemble the linguistic structure of instant messages, were used as the data basis. Specifically, the training data set provided by the "International Predator Identification" competition organisers was selected [29]. This data set contains chats between convicted sex offenders and pseudo-victims, i.e. adults posing as minors and two types of harmless chat protocols: Internet Reality Chats (IRC) on various topics and Omegle chats between adults on sexual topics.

The competition organisers [29] have already grouped the 903,607 messages in the training data set into 66.927 coherent conversations based on the time intervals between messages in a chat. Since fewer than 4% of these conversations involve a sex offender, and since sex offenders may also talk to pseudo-victims about harmless matters, it can be assumed that the grooming topic is only represented to a small extent in the data set, which is a realistic scenario for hypothesis-driven topic modelling.

All unique conversation messages were considered one topic-modelling document (see section 3). Some basic preprocessing steps were performed, such as normalising XML characters, splitting hashtags to treat them as ordinary tokens, and removing all tokens irrelevant to the topic of grooming, such as mentions and stop words. Subsequently, lemmatised and lowercase bag-of-words and bag-of-bigrams representations of the conversations served as input for training a unigram and bigram SGTM, respectively.

4.2 Implementation of Seed-Guided Topic Modelling

The SGTM algorithm selected was the keyword-assisted topic model (keyATM) developed by [18], as it had already shown promising results in previous work [e.g. 20]. For this, it was necessary to represent the hypothesis with topic-characteristic seed terms instead of the a priori distribution described in section 3.

The investigator who usually determines these seed terms was replaced by the LLM Mistral Large 2 [2] for test purposes. This model was also used for all further experiments in a zero-shot scenario. First, a new role was assigned to the LLM via a system prompt, that of a forensic linguist specialising in online grooming. It was then prompted to generate typical bigrams for online grooming conversations. The seed bigrams obtained were used directly as input for the bigram SGTM and split into individual words for the unigram SGTM, with keyATM requiring the seed terms to be reduced to those present in the vocabulary of the training data [18]. Some seed unigrams and bigrams are listed as examples in Table 1.

Table 1: Examples of seed unigrams and bigrams used to extract an online grooming topic.

Seed Unigrams	"secretly", "meet", "personal", "question", "inappropriate", "photo", "intimate", "sexual"
Seed Bigrams	"personal question", "private chat", "explicit content", "personal secret"

Using the generated seed term sets, keyATM was trained with the bag-of-words and subsequently with the bag-of-bigram representations of the conversation documents over 2000 iterations. Training the topic model was the most time-consuming step of the experiments, taking about 25 minutes using an AMD Ryzen 7 PRO 8840Uw 8-core processor equipped with 32 GB RAM. The hyperparameters α as a prior for the document-topic distribution and β as a prior for the topic-word distribution were set to 0.1 and 0.01. By choosing small values for these parameters, more focused

distributions can be obtained, which, according to [57], is more suitable for short texts. Specific hyperparameters of keyATM are set to the default values proposed by [18]. In addition to the topic encoded by seed words, keyATM can extract a predefined number of common, unseeded topics. These can absorb the dominant small talk topics, which are thus not mixed with the case-relevant topic. keyATM is known to be relatively insensitive to the number of unseeded topics [18]. Nevertheless, it was separately optimised for the documents, tokenised into unigrams and bigrams, according to semantic coherence [37], to find an appropriate number of topics.

Since the goal was only to extract a single case-relevant topic, the topic ranking by quantitative evidence described in section 3 was omitted in these first experiments.

4.3 Topic representations

The following representations of the extracted grooming topic were examined:

Most probable unigrams: The ten words with the highest probability in the grooming topic were used as baseline representation, with the topic being extracted using the unigram SGTM.

Most probable bigrams: The most straightforward alternative was to extract the topics with the bigram SGTM and to describe the grooming topic with the ten most probable bigrams, reported sometimes as better interpretable [e.g. 12, 40].

Most characteristic, extracted messages: The extractive approach to determining characteristic short texts combined information on topic coverage and semantic similarity between messages and the grooming topic. In the first step, all messages from the five conversations in which the grooming topic occurred with the highest topic coverage were selected as candidates.

Subsequently, the approach proposed by [50] was used, which deems the messages as characteristic whose embedding is most similar to an embedding representation of the grooming topic. The approach in this work differed from that of [50] in that not word2vec [36] was used. Instead, a fastText skip-gram model [8] was trained on the preprocessed training data with default values for all parameters [19] to represent the words of the vocabulary as 100-dimensional vectors. fastText was preferred over word2vec because of its robustness against spelling errors common in social media data [15]. Then, a vector representation of the candidate messages and the grooming topic was obtained by averaging the embeddings of the unique words in the messages, respectively, the ten most probable words in the topic [50]. The three messages with the highest cosine similarity of their embedding representation were determined to be the most characteristic.

Most characteristic, generated messages: The extractive method was compared with a generative approach to obtain characteristic messages using the LLM Mistral Large 2 in a zero-shot scenario. First, the system parameter was set to assign the LLM the personality of an author so that it is possible to overcome possible guardrails that might prevent the LLM from writing harmful grooming messages [e.g. 13], and also so that the LLM might write more imaginative messages. The LLM was then instructed to write three conversational messages reflecting a topic described by the ten most probable words of the grooming topic presented. It was explicitly stated to the LLM that not all top words have to appear in the message to avoid messages that seem too affected.

Summary of the conversations covered by the grooming topic: Finally, Mistral 2.0 Large was assigned the role of a text summarisation system to summarise the five conversations with the highest coverage of the grooming topic. It was to place particular emphasis on making the main topic of the chats clear.

5 Results

This section qualitatively compares the five topic representations, whereas a quantitative evaluation is beyond the scope of this paper.

Table 2: Representation of the online grooming topic by the most probable unigrams and bigrams, respectively.

	Most probable unigrams	Most probable bigrams
1	asl	bestthe bestthe
2	stranger	fuck face
3	omegle	waffle sniff
4	ur	sniff fuck
5	girl	face fagblue
6	**chat**	fagblue waffle
7	send	work fine
8	male	window xp
9	cool	dispatch error
10	horny	hard drive

Most probable unigrams: In the first column of Table 2, the grooming topic is represented by the ten most likely unigrams, with the seed words highlighted in bold. It is noticeable that it is impossible for most of the words to assess whether they are related to online grooming without further context. At most, it can be speculated that words such as "asl" as an abbreviation for age, sex and location, and "girl" or "male" are used in the initial phase of grooming, when the offender is trying to get to know the minor better [38]. Similarly, words such as "send" can be completely harmless but can also be used in a request for (sexual) images. The only word related to sexuality would be "horny". However, since words such as "omegle" or "stranger" also have a high probability, it cannot be ruled out that this topic refers to general sexual communication between adults, for example, on Omegle, where strangers are paired randomly in sessions. Online grooming and 'legal' sex talk on Omegle are difficult to separate as they use similar vocabulary [7].

Most probable bigrams: The bigram representation in the second column of Table 2 did not lead to better interpretability. Instead, unusual bigrams such as "bestthe" and everyday phrases such as "work fine" were highly probable in the topic. Furthermore, some multiword terms such as "window xp", "dispatch error", and "hard drive" do not reflect the expected topic of cyber grooming but can instead be associated with computer science. One possible cause of the low quality of the topics is the problem already mentioned in section 2, that the vocabulary is increased by the tokenisation into bigrams and thus the issue of high sparseness, which is problematic for topic

Table 3: Representation of the grooming topic by characteristic messages extracted from the data set, as well as messages and a summary generated by an LLM.

Method	Topic representation
Characteristic extracted messages	1. You're now chatting with a random stranger. Say hi! 2. female 3. ficial messages from Omegle will not be sent with the label 'Stranger:'. Strangers claiming to represent Omegle are lying.You're now chatting with a random stranger. Say hi!
Characteristic generated messages	1. "Just had the coolest convo with a random girl on Omegle! Never know who you'll meet next. #Omegle #RandomChat" 2. "Asl? Really? Come on, let's be more original than that! #OmegleStruggles #ChatLife" 3. "Met a dude on Omegle who was actually respectful and not all 'horny' upfront, faith in humanity restored! #NotAllGuys #CoolConvos"
Generative topic summary	The conversations involve individuals discussing and arranging potential meetings, often with a romantic or sexual intent. They exchange details about their ages, locations, and interests and discuss the logistics of meeting up, including sharing phone numbers, addresses, and setting times. Some conversations also involve flirtation and discussions about personal preferences and past relationships.

modelling [12]. The vocabulary size of the conversations tokenised into bigrams was over five times higher than those tokenised into unigrams. This problem could be addressed by reducing the vocabulary to meaningful bigrams, such as noun phrases, as proposed by [55] for unsupervised topic modelling. Nevertheless, it is questionable whether bigrams alone allow a sufficient interpretation since, for example, "fagblue waffle" could refer to an invented venereal disease circulating on social media [5], where it is not clear without further context whether it is related to online grooming.

Most characteristic, extracted messages: The three messages directly extracted from the data set are shown in the first row of Table 3, with the third message truncated because it contained the listed text over 900 times. It is noticeable that the first and third messages are more likely to be associated with adult sex talk. That can be attributed to the fact that the messages were determined based on their similarity to the top ten words of the grooming topic, which included words such as "stranger" or "omegle" (see Table 2). Another problem is that the second message consists of only a single word, so without further context, it is impossible to tell whether it relates to a grooming topic. One solution would be to display the previous and following messages for each characteristic message. However, this would make the display less compact, as the data set contains longer messages with more than 66.000 words.

Most characteristic, generated messages: The messages generated by the LLM based on the ten most probable words of the grooming topic are illustrated in the second row of Table 3. Once again, the problem arises that the messages tend to describe chatting with strangers via Omegle. This topic representation could, therefore, falsely lead to the conclusion that only sex conversations between adults (on Omegle) can be found in the chats examined, but that there is no evidence for the case-relevant topic of grooming. Accordingly, it is essential to improve the extraction of the grooming topic by the SGTM, for example, by choosing seed words that are more characteristic of grooming since these essentially influence the subsequent topic representations.

Summary of the conversations covered by the grooming topic: The summary generated by the LLM, displayed in the last row of Table 3, does not explicitly mention cyber grooming. However, in contrast to all the other topic representations, it is clear that the chats involve a contact initiation with sexual intentions, which even included arranging (physical) meetings. The exchange of contact information mentioned in the summary and the slow development of a relationship, including the exchange of interests, are typical elements of the grooming process [26, 38]. This information can thus be seen as evidence that the topic of 'grooming' is present in the chats and that the hypothesis that grooming occurred is fulfilled. Investigators can then prioritise the conversations summarised by the LLM that have high coverage of the grooming topic in the manual evaluation to check whether minors were involved in the alleged appointments for meetings.

6 Conclusion

When conducting forensic analysis of large amounts of communication data, investigators are often interested in whether an expected case-relevant topic has been discussed, providing evidence for a specific forensic hypothesis. To this end, this paper presents an interactive approach that integrates a representation of the expected topic into topic modelling and provides evidence for the hypothesis based on the agreement between expected and extracted topics. To enable the investigator to evaluate the evidentiary topics and provide feedback, they must be interpretable, so various topic representations were examined. The initial experiments based on an online grooming data set showed that high interpretability can be achieved by summarising thematically important conversations.

However, a comprehensive user study is needed to evaluate interpretability quantitatively, for example, based on topic labelling [14]. Further research should also investigate the suitability of neural SGTM algorithms and use an a priori distribution as input to quantify the evidence. Finally, the question remains to what extent the integration of milieu-specific dictionaries and word recommendation systems [e.g. 22] into the pipeline can overcome the particular

challenge of coded conversations. Here, it should also be investigated whether pre-processing steps such as lemmatisation have led to a loss of intentional spelling mistakes used to encode hidden meanings [e.g. 43].

References

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC)*. Gaithersburg, Maryland, 1–12.

[2] Mistral AI. 2024. Large Enough. https://mistral.ai/news/mistral-large-2407/ Section: news.

[3] Fuad Alattar and Khaled Shaalan. 2021. Using Artificial Intelligence to Understand What Causes Sentiment Changes on Social Media. *IEEE Access* 9 (2021), 61756–61767. doi:10.1109/ACCESS.2021.3073657

[4] Humaira Arshad, Aman Jantan, and Esther Omolara. 2019. Evidence collection and forensics on social networks: Research challenges and directions. *Digital Investigation* 28 (March 2019), 126–138. doi:10.1016/j.diin.2019.02.001

[5] Ann D. Bagchi, Adam Thompson, Kasny Damas, and Elise Corasim. 2022. Step UP! To Stamp Out Stigma: adapting and testing a bystander intervention to reduce HIV-related stigma. *Journal of Public Health* 30, 1 (Jan. 2022), 185–193. doi:10.1007/s10389-020-01284-1

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *The Journal of Machine Learning Research* 3 (May 2003), 993–1022. https://dl.acm.org/doi/10.5555/944919.944937

[7] Dasha Bogdanova, Paolo Rosso, and Thamar Solorio. 2014. Exploring high-level features for detecting cyberpedophilia. *Computer Speech and Language* 28, 1 (Jan. 2014), 108–120. doi:10.1016/j.csl.2013.04.007

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (Dec. 2017), 135–146. doi:10.1162/tacl_a_00051

[9] Federico Borazio, Danilo Croce, Giorgio Gambosi, Roberto Basili, Daniele Margiotta, Antonio Scaiella, Martina Del Manso, Daniele Petrone, Andrea Cannone, Alberto M. Urdiales, Chiara Sacco, Patrizio Pezzotti, Flavia Riccardo, Daniele Mipatrini, Federica Ferraro, and Sobha Pilati. 2024. Semi-automatic topic discovery and classification for epidemic intelligence via large language models. In *Proceedings of the second workshop on natural language processing for political sciences @ LREC-COLING 2024*, Haithem Afli, Houda Bouamor, Cristina Blasi Casagran, and Sahar Ghannay (Eds.). ELRA and ICCL, Torino, Italia, 68–84. https://aclanthology.org/2024.politicalnlp-1.8

[10] Lucia Busso, Marton Petyko, Sarah Atkins, and Tim Grant. 2022. Operation Heron: Latent Topic Changes in an Abusive Letter Series. *Corpora* 17, 2 (Aug. 2022), 225–258. doi:10.3366/cor.2022.0255

[11] Maxime Bérubé, Thuc-Uyên Tang, Francis Fortin, Sefa Ozalp, Matthew L. Williams, and Pete Burnap. 2020. Social Media Forensics Applied to Assessment of Post–Critical Incident Social Reaction: The Case of the 2017 Manchester Arena Terrorist Attack. *Forensic Science International* 313 (Aug. 2020), 1–37. doi:10.1016/j.forsciint.2020.110364

[12] Rob Churchill and Lisa Singh. 2021. textPrep: A Text Preprocessing Toolkit for Topic Modeling on Social Media Data. In *Proceedings of the 10th International Conference on Data Science, Technology and Applications (DATA)*. Science and Technology Publications (SCITEPRESS), Online, 60–70. doi:10.5220/0010559000600070

[13] Ameet Deshpande, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, and Karthik Narasimhan. 2023. Toxicity in chatgpt: Analyzing persona-assigned language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, Singapore, 1236–1270. doi:10.18653/v1/2023.findings-emnlp.88

[14] Caitlin Doogan and Wray Buntine. 2021. Topic Model or Topic Twaddle? Re-evaluating Semantic Interpretability Measures. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 3824–3848. doi:10.18653/v1/2021.naacl-main.300

[15] Ashwin Geet D'Sa, Irina Illina, and Dominique Fohr. 2020. BERT and fastText Embeddings for Automatic Detection of Toxic Speech. In *Proceedings of the International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)*. IEEE, Tunis, Tunisia, 1–5. doi:10.1109/OCTA49274.2020.9151853

[16] Mennatallah El-Assady, Rita Sevastjanova, Fabian Sperrle, Daniel Keim, and Christopher Collins. 2018. Progressive Learning of Topic Modeling Parameters: A Visual Analytics Framework. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 382–391. doi:10.1109/TVCG.2017.2745080 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[17] Ahmed Elbagoury, Rania Ibrahim, Ahmed Farahat, Mohamed Kamel, and Fakhri Karray. 2021. Exemplar-Based Topic Detection in Twitter Streams. In *Proceedings of the Ninth International AAAI Conference on Web and Social Media*. AAAI Press, Oxford, Großbritannien, 610–613. doi:10.1609/icwsm.v9i1.14651

[18] Shusei Eshima, Kosuke Imai, and Tomoya Sasaki. 2023. Keyword-Assisted Topic Models. *American Journal of Political Science* 0, 0 (April 2023), 1–21. doi:10.1111/ajps.12779

[19] Facebook Open Source. 2023. List of options - fastText. https://fasttext.cc/index.html

[20] Jenny Felser, Dirk Labudde, and Michael Spranger. 2023. Towards Hypothesis-driven Forensic Text Exploration System. In *Proceedings of the 2023 IARIA Annual Congress on Frontiers in Science, Technology, Services, and Applications*. Valencia, 42–47.

[21] Jenny Felser and Michael Spranger. 2024. Semi-supervised topic modelling as a tool for hypothesis-driven forensic communication analysis. In *Proceedings of the 4. International Workshop on Digital Forensics (IWDF4)*, Vol. Informatik 2024. Gesellschaft für Informatik e.V., Bonn, 321–330. doi:10.18420/inf2024_22 tex.pissn: 1617-5468.

[22] Jenny Felser, Jian Xi, Christoph Demus, Dirk Labudde, and Michael Spranger. 2022. Recommendation of Query Terms for Colloquial Texts in Forensic Text Analysis. In *Proceedings of the International Workshop On Digital Forensics (IWDF)*. Gesellschaft für Informatik (GI), Hamburg, Deutschland, 35–47. doi:10.18420/inf2022_02

[23] Jielin Feng, Kehao Wu, and Siming Chen. 2023. TopicBubbler: An interactive visual analytics system for cross-level fine-grained exploration of social media data. *Visual Informatics* 7, 4 (Dec. 2023), 41–56. doi:10.1016/j.visinf.2023.08.002

[24] Ryan J. Gallagher, Kyle Reing, David Kale, and Greg Ver Steeg. 2017. Anchored Correlation Explanation: Topic Modeling with Minimal Domain Knowledge. *Transactions of the Association for Computational Linguistics* 5 (Dec. 2017), 529–542. doi:10.1162/tacl_a_00078

[25] Nikolaos Giarelis, Charalampos Mastrokostas, and Nikos Karacapilidis. 2023. Abstractive vs. Extractive Summarization: An Experimental Review. *Applied Sciences* 13, 13 (June 2023), 7620. doi:10.3390/app13137620

[26] Aditi Gupta, Ponnurangam Kumaraguru, and Ashish Sureka. 2012. Characterizing Pedophile Conversations on the Internet using Online Grooming. doi:10.48550/arXiv.1208.4324 arXiv:1208.4324 [cs].

[27] Liangjie Hong and Brian D. Davison. 2010. Empirical study of topic modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*. Association for Computing Machinery, Washington D.C., USA, 80–88. doi:10.1145/1964858.1964870

[28] Rania Ibrahim, Ahmed Elbagoury, Mohamed S. Kamel, and Fakhri Karray. 2018. Tools and approaches for topic detection from Twitter streams: survey. *Knowledge and Information Systems* 54, 3 (March 2018), 511–539. doi:10.1007/s10115-017-1081-x

[29] Giacomo Inches and Fabio Crestani. 2012. Overview of the International Sexual Predator Identification Competition at PAN-2012. In *CLEF (Online Working Notes/Labs/Workshop)*. CEUR-WS, Rome, 1–12. https://pan.webis.de/downloads/publications/papers/inches_2012.pdf

[30] Hannah Kim, Dongjin Choi, Barry Drake, Alex Endert, and Haesun Park. 2019. TopicSifter: Interactive Search Space Reduction through Targeted Topic Modeling. In *Proceedings of the Conference on Visual Analytics Science and Technology*. IEEE, Vancouver, Kanada, 35–45. doi:10.1109/VAST47406.2019.8986922

[31] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2016. Topic Modeling for Short Texts with Auxiliary Word Embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16)*. Association for Computing Machinery, New York City, New York, USA, 165–174. doi:10.1145/2911451.2911499 Number of pages: 10 Place: Pisa, Italy.

[32] Jiawei Li, Wen-Hao Chen, Qing Xu, Neal Shah, Jillian C. Kohler, and Tim K. Mackey. 2020. Detection of self-reported experiences with corruption on twitter using unsupervised machine learning. *Social Sciences & Humanities Open* 2, 1 (Jan. 2020), 100060. doi:10.1016/j.ssaho.2020.100060

[33] Yang Lin, Xin Gao, Xu Chu, Yasha Wang, Junfeng Zhao, and Chao Chen. 2023. Enhancing Neural Topic Model with Multi-Level Supervisions from Seed Words. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 13361–13377. doi:10.18653/v1/2023.findings-acl.845

[34] Robert Lindsey, William Headden, and Michael Stipicevic. 2012. A Phrase-Discovering Topic Model Using Hierarchical Pitman-Yor Processes. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, Jun'ichi Tsujii, James Henderson, and Marius Paşca (Eds.). Association for Computational Linguistics, Jeju Island, Korea, 214–222. https://aclanthology.org/D12-1020

[35] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of the 18th international conference on World wide web*. ACM, Madrid Spain, 131–140. doi:10.1145/1526709.1526728

[36] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Vol. 2. Curran Associates, Inc., Lake Tahoe, Nevada, USA, 3111–3119. https://dl.acm.org/doi/10.5555/2999792.2999959

[37] David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing Semantic Coherence in Topic Models. In *Proceedings of the 2011 conference on empirical methods in natural language processing*. Association for Computational Linguistics, Edinburgh, Schottland, Großbritannien, 262–272. https://aclanthology.org/D11-1024

[38] Rachel O'Connell. 2003. *A typology of child cybersexploitation and online grooming practices*. Technical Report. Cyberspace Research Unit, University of Central Lancashire, Preston, UK. 1–19 pages. http://image.guardian.co.uk/sys-files/Society/documents/2003/07/17/Groomingreport.pdf

[39] Sumit Pai, Sounak Lahiri, Ujjwal Kumar, Krishanu Baksi, Elijah Soba, Michael Suesserman, Nirmala Pudota, Jon Foster, Edward Bowen, and Sanmitra Bhattacharya. 2023. Exploration of Open Large Language Models for eDiscovery. In *Proceedings of the Natural Legal Language Processing Workshop 2023*. Association for Computational Linguistics, Singapore, 166–177. doi:10.18653/v1/2023.nllp-1.17

[40] Youngsun Park, Md. Hijbul Alam, Woo-Jong Ryu, and Sangkeun Lee. 2015. BL-LDA: Bringing Bigram to Supervised Topic Model. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, Las Vegas, NV, USA, 83–88. doi:10.1109/CSCI.2015.146

[41] Liwen Peng, Xiaolin Zhu, and Peng Zhang. 2020. An Efficient Model for Smartphone Forensics Using SMS Spam Filtering. In *Proceedings of the 3rd International Conference on Hot Information-Centric Networking (HotICN)*. IEEE, Hefei, China, 166–169. doi:10.1109/HotICN50779.2020.9350843

[42] William Rule, Wanyi Duan, Nitesh Prakash, Ni Zhuang, Rafael C. Alvarado, and Donald E. Brown. 2018. Social Pressure Analysis of Local Events Using Social Media Data. In *Proceedings of the Conference on Systems and Information Engineering Design Symposium*. IEEE, Charlottesville, Virginia, USA, 277–281. doi:10.1109/SIEDS.2018.8374751

[43] Michael Spranger, Florian Heinke, Luisa Appelt, Marcus Puder, and Dirk Labudde. 2016. MoNA: Automated Identification of Evidence in Forensic Short Messages. *International Journal on Advances in Security* 9, 1 & 2 (Aug. 2016), 14–24. http://www.iariajournals.org/security/

[44] Michael Spranger and Dirk Labudde. 2013. Semantic Tools for Forensics: Approaches in Forensic Text Analysis. In *Proceedings of the Third International Conference on Advances in Information Mining and Management (IMMM)*. IARIA Press, Lissabon, Portugal, 97–100. doi:10.13140/RG.2.1.2342.7685

[45] Michael Spranger, Jian Xi, Lukas Jaeckel, Jenny Felser, and Dirk Labudde. 2022. MoNA: A Forensic Analysis Platform for Mobile Communication. *Künstliche Intelligenz* 36, 2 (May 2022), 163–169. doi:10.1007/s13218-022-00762-w

[46] Chris Stahlhut. 2019. Interactive Evidence Detection: train state-of-the-art model out-of-domain or simple model interactively?. In *Proceedings of the second workshop on fact extraction and verification (FEVER)*, James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal (Eds.). Association for Computational Linguistics, Hong Kong, China, 79–89. doi:10.18653/v1/D19-6613

[47] Hatma Suryotrisongko, Hari Ginardi, Henning Titi Ciptaningtyas, Saeed Dehqan, and Yasuo Musashi. 2022. Topic Modeling for Cyber Threat Intelligence (CTI). In *Proceedings of the Seventh International Conference on Informatics and Computing*. IEEE, Denpasar, Bali, Indonesien, 1–7. doi:10.1109/ICIC56845.2022.10006988

[48] Marco Sánchez and Luis Urquiza. 2024. Improving fraud detection with semi-supervised topic modeling and keyword integration. *PeerJ Computer Science* 10 (Jan. 2024), e1733. doi:10.7717/peerj-cs.1733

[49] Rika Tanaka and Yusuke Fukazawa. 2024. Integrating Supervised Extractive and Generative Language Models for Suicide Risk Evidence Summarization. In *Proceedings of the 9th Workshop on Computational Linguistics and Clinical Psychology (CLPsych 2024)*, Andrew Yates, Bart Desmet, Emily Prud'hommeaux, Ayah Zirikly, Steven Bedrick, Sean MacAvaney, Kfir Bar, Molly Ireland, and Yaakov Ophir (Eds.). Association for Computational Linguistics, St. Julians, Malta, 270–277. https://aclanthology.org/2024.clpsych-1.27

[50] Gulmira Tolegen, Alymzhan Toleu, Rustam Mussabayev, and Alexander Krassovitskiy. 2022. A Clustering-based Approach for Topic Modeling via Word Network Analysis. In *Proceedings of the 7th International Conference on Computer Science and Engineering*. IEEE, Diyarbakir, Türkei, 192–197. doi:10.1109/UBMK55850.2022.9919530 ISSN: 2521-1641.

[51] Tala Vahedi, Benjamin Ampel, Sagar Samtani, and Hsinchun Chen. 2021. Identifying and Categorizing Malicious Content on Paste Sites: A Neural Topic Modeling Approach. In *Proceedings of the International Conference on Intelligence and Security Informatics*. IEEE, San Antonio, Texas, USA, 1–6. doi:10.1109/ISI53945.2021.9624765

[52] Sebastian Wachs, Karsten D. Wolf, and Ching-Ching Pan. 2012. Cybergrooming: risk factors, coping strategies and associations with cyberbullying. *Psicothema* 24, Número 4 (2012), 628–633. https://reunido.uniovi.es/index.php/PST/article/view/9714

[53] Hans Walder, Thomas Hansjakob, Thomas E. Gundlach, and Peter Straub. 2023. *Kriminalistisches Denken* (12 ed.). Number 41 in Grundlagen der Kriminalistik. Kriminalistik-Verlag, Heidelberg, Deutschland.

[54] Kohei Watanabe and Alexander Baturo. 2023. Seeded Sequential LDA: A Semi-Supervised Algorithm for Topic-Specific Analysis of Sentences. *Social Science Computer Review* 0, 0 (May 2023), 1–25. doi:10.1177/08944393231178605

[55] Mingyang Xu, Ruixin Yang, Stephen Ranshous, Shijie Li, and Nagiza F. Samatova. 2017. Leveraging External Knowledge for Phrase-Based Topic Modeling. In *Proceedings of the Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, Taipei, Taiwan, 29–32. doi:10.1109/TAAI.2017.25 ISSN: 2376-6824.

[56] Riheng Yao, Qiudan Li, Wei-Hsuan Lo-Ciganic, and Daniel Dajun Zeng. 2019. A Prior Knowledge Based Neural Attention Model for Opioid Topic Identification. In *Proceedings of the International Conference on Intelligence and Security Informatics (ISI)*. IEEE, Shenzhen, 215–217. https://ieeexplore.ieee.org/document/8823280/

[57] Yuan Zuo, Junjie Wu, Hui Zhang, Hao Lin, Fei Wang, Ke Xu, and Hui Xiong. 2016. Topic Modeling of Short Texts: A Pseudo-Document View. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, San Francisco, Kalifornien, USA, 2105–2114. doi:10.1145/2939672.2939880

Low-overhead and Non-invasive Electromagnetic Side-Channel Monitoring for Forensic-ready Industrial Control Systems

Buddhima Weerasinghe
University of Colombo School of Computing
Colombo, Sri Lanka
ima@ucsc.cmb.ac.lk

Asanka Sayakkara
University of Colombo School of Computing
Colombo, Sri Lanka
asa@ucsc.cmb.ac.lk

Kasun De Zoysa
University of Colombo School of Computing
Colombo, Sri Lanka
kasun@ucsc.cmb.ac.lk

Mark Scanlon
School of Computer Science
University College Dublin
Dublin, Ireland
mark.scanlon@ucd.ie

Abstract

Industrial control systems (ICS) are the backbone of modern manufacturing facilities. Due to the distributed nature of ICS hardware in their deployment environment, they are often networked through Ethernet, opening up a window for network-based attacks. Preventive security measures, such as constant packet capture and inspection, are impractical due to the computational overhead required. Therefore, computationally feasible trigger mechanisms are needed that can activate security, as well as on-demand forensic readiness features, in the infrastructure. This work proposes an approach to monitor ICS network infrastructure using unintentional electromagnetic (EM) radiation emitted by Ethernet network cables during their regular operation. An empirical evaluation highlights that it is possible to detect various types of denial of service (DoS) attacks through EM emission patterns of Ethernet cables with considerable accuracy (HTTP Flood = 99.70%, TCP Flood = 73.22%, UDP Flood = 69.95%). Based on the experimental findings, this work introduces an architecture for the ICS infrastructure to be forensic-ready with minimal computational resources while being independent and non-invasive to the infrastructure itself.

CCS Concepts

• **Applied computing** → **Surveillance mechanisms**; **Network forensics**; • **Security and privacy** → *Side-channel analysis and countermeasures.*

Keywords

Industrial control systems, electromagnetic side channel analysis, network security, forensic readiness

ACM Reference Format:
Buddhima Weerasinghe, Asanka Sayakkara, Kasun De Zoysa, and Mark Scanlon. 2025. Low-overhead and Non-invasive Electromagnetic Side-Channel Monitoring for Forensic-ready Industrial Control Systems. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712716.3712722

DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712722

1 Introduction

Modern industrial manufacturing facilities consist of highly complicated machinery that is required to operate around the clock to meet production targets. Industrial control systems (ICS) are the backbone of these highly demanding environments, which monitor and control factory equipment to keep them in order [10]. ICSs are typically networked together over Ethernet [5]. The significant role ICS plays attracts a host of security threats. Being network devices with time-sensitive functionalities, most of such threats are delivered through the network [3]. Network-based attacks to ICS includes, denial of service (DoS) attacks, remote malware infections, Man-in-the-Middle attacks (MitM), Spoofing, etc. These attacks can originate from both external and internal sources. For example, a DoS attack may originate from malware-infected network devices from an industrial facility that targets a critical component of their own ICS [11].

When security incidents related to ICS occur, they are subject to forensic investigations [23]. The success of such a forensic analysis depends on the availability of useful evidence retained in the ICS and other network infrastructure. This had led to the need to have a comprehensive network forensic readiness strategy in place to ensure that pertinent evidence is available when needed, but the balance of how much network traffic to store comes with considerations on performance impacts and potentially excessive data collection [17]. Various network and embedded system security mechanisms can be employed in ICS to ensure their security. Furthermore, their forensic readiness can be enabled through the continuous capture and saving of network packets, the regular preservation of the internal states of the ICS devices, and various other methods. Enabling measures for ICS security and forensic readiness have been shown to incur significant computational overhead in terms of real-time processing of network traffic and storage capacity [2]. Meanwhile, not having such measures in place can derail an investigation due to the loss of valuable evidence during the post-incident stage [1].

An ideal mechanism to ensure security and forensic readiness of ICS infrastructure should consist of a variety of measures that

should be enabled on demand whenever an indication of a threat/ incident is looming. This could be detected and triggered by an independent monitoring mechanism that has minimal computational overheads. The necessity for such trigger mechanisms to be independent from the ICS infrastructure itself is due to the possibility that whatever threatening the ICS infrastructure can pose the same threat to the security and forensic readiness triggering mechanisms. In these circumstances, research is needed to discover novel non-invasive, low-overhead, network-based threat detection mechanisms.

This work explores the potential of using electromagnetic (EM) radiation emitted by the ICS network infrastructure as a window to detect network-based threats and act as a trigger mechanism to activate the forensic readiness features of the ICS infrastructure. Toward this goal, through empirical experimentation, appropriate algorithms, tools, and techniques are developed and tested to evaluate the effectiveness of such an approach.

This paper makes the following contributions:

- Introduces EM side-channel analysis (EM-SCA) as a non-intrusive technique to detect network-based threats to ICS infrastructure.
- Experimentally evaluates three light-weight machine learning algorithms to process EM radiation patterns caused by malicious traffic.
- Presents a methodology to trigger security and forensic-readiness features of ICS infrastructure with minimal overhead.

The rest of this paper is organised as follows. Section 2 provides a brief overview of the state-of-the-art in this problem domain. Section 3 introduces the tools and techniques used to capture and analyse radiation data originating from Ethernet network infrastructure. Based on these techniques, Section 4 experimentally evaluates multiple machine learning algorithms to distinguish EM radiation patterns caused by malicious network traffic. Section 5 proposes a novel ICS security architecture based on the experimental findings. Finally, Section 6 discusses the conclusion and future directions of this work.

2 Related Work

Detecting network-based attacks through traffic pattern analysis is a widely studied area [16]. Specific packet types like TCP or UDP and the rate at which they flow can signal potential attacks. Neto et al. [15] created a comprehensive dataset of network attacks comprising 33 types grouped into seven categories, including DDoS, DoS, reconnaissance, web-based, brute force, spoofing, and Mirai attacks. Their study shows that machine learning algorithms can effectively distinguish between these attacks with high accuracy, underscoring the importance of network traffic data in attack classification. Similarly, Dhanya et al. [4] designed machine learning and deep learning models to detect intrusions and classify attacks, using the UNSW-NB15 dataset [14], which features nine types of attacks and 49 attributes derived from contemporary network traffic patterns.

For Industrial Control Systems (ICS), detecting anomalous behaviour often relies on data from various sensors. Tang et al. [22] used neural graph networks for anomaly detection in ICS environments, while Kim et al. [12] explored several machine learning-based techniques for the same purpose. By aggregating data from multiple sensors, such as temperature or flow sensors, these approaches produce accurate and reliable results. However, centralising and processing large volumes of sensor data poses computational challenges, especially with the increased complexity of modern machine learning models that require significant computational resources.

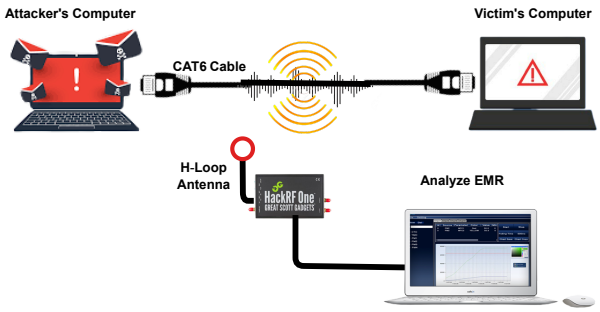


Figure 1: Overview of the experimental hardware setup.

In recent years, information leakage via EM radiation from Ethernet cables has become a significant concern in security and digital forensics. Schulz et al. [21] explored the vulnerabilities of Ethernet networks to -destructive wiretap attacks, demonstrating that attackers using a USRP X300 device can intercept and decode sensitive information, such as the preamble, start of frame delimiter (SFD) and MAC address. Further expanding on this threat, Guri introduced LANTENNA [8], an EM attack that enables data leakage from air-gapped networks by turning Ethernet cables into unintended transmitting antennas. This approach uses malware to manipulate the EM emissions of a compromised workstation's Ethernet cable, allowing covert data transmission. Similarly, Sachintha et al. [18] revealed a related EM-based attack targeting Industrial Control Systems (ICS), where compromised firmware in network controllers can encode sensitive information within network packet patterns. An attacker can capture the EM radiation from a few metres away to extract the transmitted data.

These studies highlight the evolving risk landscape, where EM radiation from wired connections becomes a potential conduit for covert data exfiltration.

3 Radiation from Network Infrastructure

Although a wide array of frequency channels have the potential to convey information about network traffic, typically only a small subset of them prove to be truly valuable. Some channels may contain redundant information, while others might not disclose any information at all. Hence, the identification of these informative frequency channels from the numerous available channels plays a pivotal role in enhancing the efficiency of EM-SCA for digital forensics [19, 20].

3.1 Experimental Hardware Setup

The experimental hardware setup involves a computer, representing the attacker, connected to another computer, representing the victim, via a Cat 6 unshielded twisted pair (UTP) Ethernet cable, simulating the ICS wired network. EM radiation signals from this setup are captured using a HackRF One [6] SDR device, paired with a magnetic H-loop antenna, and connected to the investigator's computer. The collected EM radiation data are stored on the investigator's computer for subsequent analysis. Figure 1 depicts the arrangement of the target network and the attacker's equipment in the experimental scenario. For the experimental evaluation of this phase, where the radiation emission frequency of the Ethernet cable is identified, both the attacker's computer connected to the Cat 6 cable and the investigator's computer connected to the SDR hardware are set to be the same machine.

3.2 Experimental Software Setup

In order to conduct experiments, a program with 3 parallel threads was executed on the attacker's computer for data collection in an annotated manner. The first thread is tasked with the transmission of network traffic on-demand over the Ethernet cable. Simultaneously, the second and third threads undertake the responsibility of capturing EM radiation and monitoring of network interfaces to identify outbound packets, respectively. Using this software setup, it is possible to transmit a specific network traffic pattern on the cable while capturing the same network packets as PCAP files, as well as emitted radiation data files, in a precisely timed manner.

3.3 Collection of Data

In accordance with insights from Guri's work [8], it has been established that Ethernet cables emit EM waves primarily in the frequency bands of 125 MHz and its harmonics, with 250 MHz being the most prominent among these harmonics. Consequently, in this work, the experiments were tailored to scan the frequency range spanning from 30 MHz to 260 MHz to pinpoint an information leakage channel. It is important to note that the lower operating frequency limit of HackRF is at 20 MHz, and approaching this limit may introduce interference from internal circuitry. Therefore, it was pragmatically decided to set 30 MHz as the lower limit of the frequency range for this investigation to ensure a reliable data collection.

The experimental software setup running on the hardware setup operates seamlessly to autonomously gather samples across the 30–260 MHz frequency range. The packet sender produces heavy TCP traffic using the Python `Scapy` library across the cable at each frequency, while the setup records the EM data in the in-phase and quadrature (IQ) data format and a corresponding PCAP file for each packet pattern. For benign traffic at each frequency, a separate IQ file and a PCAP file are recorded during the normal operation of the devices.

3.4 Dissimilarity Analysis Algorithm

Analysing the dissimilarity between two traffic patterns is a crucial component of the methodology, tasked with comparing each EM trace of heavy traffic for a given signal frequency with benign traffic EM trace of the same frequency. This comparison aims to identify the frequency at which the two network traffic patterns produce the most distinct EM radiation patterns. The procedure to achieve this task, as shown in Algorithm 1, begins by computing the Fast Fourier Transform (FFT) of EM trace files and subsequently applying the resulting FFT vectors to different similarity measurement functions, namely, cross-correlation (xCor) and normalised cross-correlation (nXCor). The calculated correlation values are then recorded in a file for subsequent analysis. The algorithm considers the shortest file length as the FFT window size, accounting for discrepancies in array sizes between the two traces.

Algorithm 1 Dissimilarity Analysis Algorithm

Require: $Data_1$: Data set containing malicious pattern traces.
$Data_2$: Data set containing benign pattern traces.
Ensure: Similarity measurement of patterns.

```
for freq ← 30 to 260 MHz do
    for pattern_i from Data_1[freq] and pattern_j from Data_2[freq] do
        windowSize ← minLength(pattern_i, pattern_j)
        fft_i ← getFFT(pattern_i, windowSize)
        fft_j ← getFFT(pattern_j, windowSize)
        xCor ← crossCorrelate(fft_i, fft_j)
        nXCor ← NormalizedCrossCorrelate(fft_i, fft_j)
        results[ ] ← (xCor, nXCor)
    end for
end for
output ← minimumSimilarity(results[ ])
```

The results of the analysis were saved to a CSV file and later plotted to visualise the variation of correlation values across different suspicious emission frequencies. As is evident in Figure 2, the xCor parameter exhibits a lower correlation than nXCor, and the most dissimilarity of two traffic patterns occurs at 240 MHz. After manually validating these results, the emitting frequency for the Cat 6 UTP cable was determined as 240 MHz. In Figure 3, the radiation pattern for attack traffic is depicted in red, while benign traffic is represented in green. A distinct contrast is evident between the two packet patterns within these PSD graphs.

4 Detection of Network Attacks

Once the emission frequency of the EM radiation was identified for the target Ethernet cable, the same hardware and software setup was used to emulate realistic network-based attack scenarios. In order to produce realistic attack traffic patterns on the Ethernet cable, the CICIoT2023 dataset [15] network attack dataset was used. The CICIoT2023 dataset is available in two different file formats: PCAP [9] and CSV. The PCAP files comprise the original data generated and collected in the CIC IoT network, which is the IoT infrastructure that consists of 105 IoT devices. The PCAP files were replayed by the software setup to recreate the exact attack scenarios on the experimental hardware platform. The selected attacks include DoS HTTP Flood, DoS TCP Flood, and DoS UDP Flood.

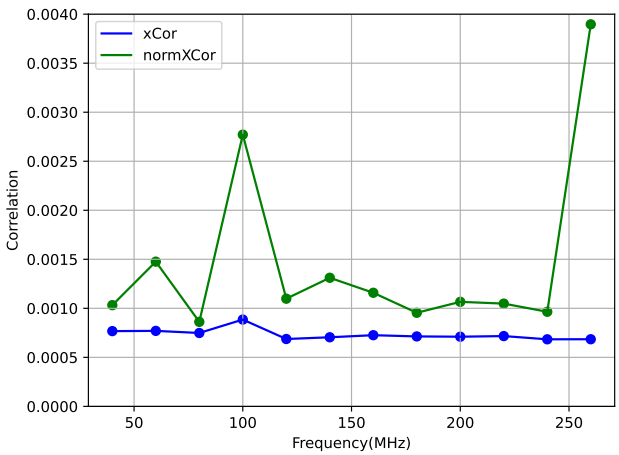


Figure 2: Variation of correlation measurements across various suspected emission frequencies.

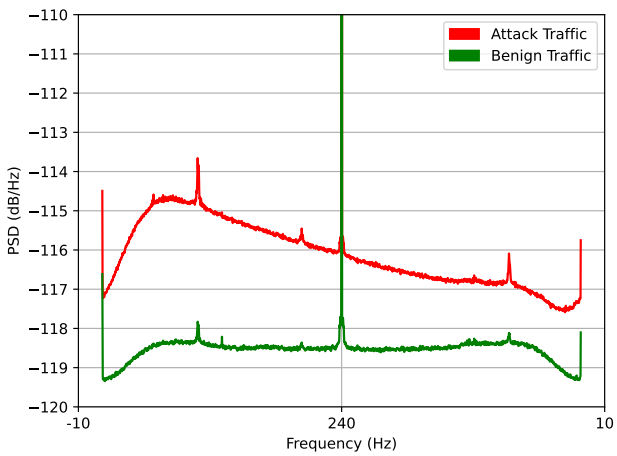


Figure 3: The power spectral density (PSD) variation of two traffic patterns at 240 MHz emission frequency.

4.1 Data Preprocessing for Machine Learning

EM radiation samples were collected at a sampling rate of 20 MHz with a target centre frequency of 240 MHz. Each trace file, representing a time-domain signal, underwent Short-time Fourier Transformation (STFT) processing to generate frequency-domain windows. In the case of MLP, RFC with AdaBoost, and SVM models, these windows were utilised as training instances, with labels corresponding to the respective network traffic.

Subsequently, individual ML models were constructed to identify malicious network activity using the resulting EM datasets for each network attack. For this purpose, 10,000 samples were extracted from each EM trace file representing a specific network attack, the relevant network attack serving as the label. Certain hyperparameters were determined on the basis of the dimensions of the EM datasets. During hyperparameter tuning, specific settings for the STFT operation, such as the FFT window size and overlapping samples, were adjusted accordingly.

4.2 Experiment 1: Impact of Probe Location

This experiment was conducted with the aim of discovering the optimal probe placement for maximising signal reception and detection. The entire data collection process was repeated 9 times, covering 3 attacks at 3 different locations, to explore various probe positions relative to the Ethernet cable. These locations were empirically selected to gauge the sensitivity of the ML models to probe placement.

Initially, the probe was positioned directly on top of the cable, in direct contact with it, marking the initial phase of analysis. Given its proximity to the primary emission source, it was expected to generate the strongest EM field, facilitating optimal signal detection. Subsequently, the probe was elevated 1cm and 10cm above the cable to simulate scenarios where cables are installed under enclosures. The results of these experiments are presented in table 1. For the initial phase, where the H-Loop antenna (EM probe) was placed directly in contact with the cable, the attacks are distinguishable from normal traffic, with DoS HTTP Flood exhibiting the highest detectability. Across all models, the highest accuracy for DoS HTTP Flood is at 99.70%. Following this, DoS TCP Flood demonstrates the highest accuracy at 73.22%, while DoS UDP Flood ranks last with the highest accuracy at 69.95%.

When the EM probe is located 1cm away from the cable, the overall accuracy in all traffic patterns falls below 60%, suggesting a detectable difference between normal and malicious operations, although less pronounced than before. Notably, the distinct gap observed in accuracy between HTTP Flood and TCP and UDP Floods diminishes. All accuracy values are within the same range, with RFC consistently demonstrating the highest accuracy. In addition, the precision and recall values align closely with the accuracy. Compared to the experiment with the probe directly atop the cable, a decrease is observed in all results, consistent with expectations that the initial location would yield higher accuracy. Meanwhile, when the EM probe is located 10cm away from the cable, the overall accuracy is recorded below 59%. However, the effectiveness of detection between normal and malicious operations remains consistent despite the change in probe location.

4.3 Experiment 2: Impact of Observation Time

When detecting network-based attacks, it is necessary to perform the detection in a minimal amount of time. The longer it takes for detection, the greater the possibility of an attack causing damage to the ICS infrastructure. With this objective, this experiment aims to determine the point in time during the attack at which it is most detectable, whether it occurs at the beginning, middle, or end of the observation time period. For this purpose, the dataset is divided into segments that correspond to the start, middle, and end of the attack sequence. Each segment is then individually analysed to assess the ability to detect it using ML algorithms. By comparing

Table 1: Performance comparison of DoS attack detection over probe placement.

Probe placement	DoS HTTP flood			DoS UDP flood			DoS TCP flood		
	RFC	MLP	SVM	RFC	MLP	SVM	RFC	MLP	SVM
On the cable	99.70	99.55	99.68	69.95	57.70	66.25	73.22	67.25	66.42
1cm away	60.42	51.90	53.99	59.38	55.15	54.26	59.50	52.00	55.57
10cm away	59.92	52.30	54.70	60.05	54.80	54.51	58.83	51.40	54.21

Table 2: Performance comparison of DoS attack detection over time splits.

Time Split	DoS HTTP flood			DoS UDP flood			DoS TCP flood		
	RFC	MLP	SVM	RFC	MLP	SVM	RFC	MLP	SVM
First third	99.58	99.40	99.73	70.43	54.20	58.95	67.70	51.65	61.60
Middle third	99.70	99.60	99.63	71.17	56.65	68.94	63.25	52.20	56.72
Last third	99.50	99.50	99.58	67.92	53.70	56.58	70.62	54.60	57.77

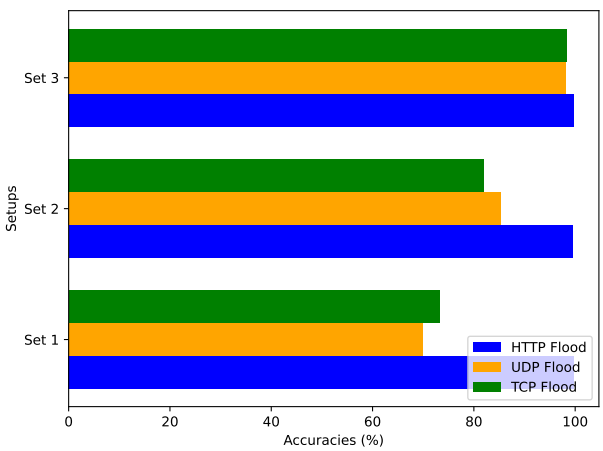


Figure 4: DoS Attack Accuracies Across Different Setups

the accuracy of detection across these segments, insights can be gained into the optimal timing to detect malicious network activity.

The analysis was performed using the dataset collected with the probe in contact with the cable. The dataset was divided into three subsets and the ML classification results were calculated for each subset. New models were trained for each subset and the results are illustrated in Table 2. In both HTTP and UDP DoS attacks, a slight increase in accuracy could be observed in the middle third time point compared to the other two time points. In contrast, TCP DoS attack exhibits a slight decrease in accuracy at the middle third time point compared to the other two time points.

4.4 Experiment 3: Impact of Sampling Rate

Capturing EM data with SDR devices requires extremely fast sample rates to capture a significant amount of information. Reducing the sample rate below a certain threshold can adversely impact the detection process. It is crucial to identify the minimum sample rate that does not compromise the effectiveness of ML-based classification for detecting malicious network activity. In this experiment, due to the lower accuracy observed (>75%) with the other two attacks, only the DoS HTTP Flood dataset collected with the probe in contact with the cable was considered. The original trace file was downsampled to 10 MHz and 4 MHz, resulting in two new trace files. These downsampled files were then used as input data for the ML models to perform the classification task.

The RFC with Adaboost and SVM models maintained close to 100% accuracy consistently at all sample rates, indicating strong performance even at lower sample rates. However, the MLP model experiences a noticeable drop in accuracy at 4 MHz, only reaching around 80%, before achieving high accuracy at higher sample rates. This suggests that while RFC and SVM are robust to lower sample rates, MLP requires a higher sample rate for optimal accuracy.

4.5 Experiment 4: Impact of the Environment

This experiment was conducted to explore the effect of ambient EM radiation in the environment on the accuracy of the classification. For this purpose, EM traces were captured under three distinct environmental conditions, i.e., Setup 1, 2 and 3, for each attack, and the accuracy of classification was assessed. The probe remained in contact with the cable during these conditions. Setup 1 corresponds to the traces used in previous experimental efforts. Setup 2 comprises data collected using the same hardware setup in a different environment, while Setup 3 encompasses traces collected in an alternate hardware configuration where the victim device was altered. RFC with AdaBoost was used for this analysis, given its superior accuracy in prior investigations. As illustrated in Figure 4, it can be seen that the DoS HTTP flood maintains a consistent accuracy across all setups. However, there is notable variability in the accuracies of DoS UDP and DoS TCP flood across different setups.

5 Monitoring Industrial Control Systems

The empirical findings in Section 4 point to the possibility of using EM radiation patterns emerging from network infrastructure to look out for network-based attacks. This section introduces a potential design blueprint for an EM-SCA-based, low-overhead, and non-intrusive ICS monitoring mechanism.

5.1 Implementation Considerations

A network-based threat detection mechanism of this nature has to include an EM radiation capturing and processing capability in real time. Although the experiments presented relied on SDR hardware to discover and capture EM emissions, it is not necessary to use them in a real-world deployment. Once the emission frequency of the infrastructure is identified, a purpose-built fixed radio receiver can be used to capture EM emission. Furthermore, the processing of captured EM data in real time can be performed onboard the signal capturing hardware using a dedicated embedded processor or a field-programmable gate array (FPGA) built into the signal capturing hardware itself [7]. Self-contained EM radiation capture and processing equipment (called a *monitor node* hereafter) can be powered using the same power supply facility in the ICS infrastructure. However, networking them with each other needs to be achieved using a communication infrastructure independent of the ICS network.

There are multiple potential approaches to connect the monitor nodes together. The obvious solution is to have a separate internal network — wired or wireless — to which the monitor nodes are connected. However, because monitor nodes do not require a high-bandwidth communication channel, having a dedicated network only to serve them is an unnecessary overhead. Alternatively, it is possible to use the existing power supply infrastructure for transferring network packets in a reliable manner, i.e., powerline communication [13]. In that approach, the monitor nodes can deliver their detection alerts and other telemetry through their power supply wiring, which is highly reliable and difficult to disrupt by an attacker.

5.2 High-level Design

Figure 5 illustrates the high-level view of an ICS infrastructure where monitor nodes are deployed in multiple locations on the network. At the highest level, i.e., Level 3, of the ICS infrastructure, there are engineering workstations that run specialised software to govern the entire manufacturing process. The level below that, i.e., Level 2, has a human-machine interface (HMI) that facilitates monitoring and controlling specific functionalities of the ICS infrastructure by allowing human technicians to interact with the devices. At Level 1, all automated devices are placed to operate the ICS infrastructure, such as programmable logic controllers (PLC), intelligent electronic devices (IED), and remote terminal units (RTU). Finally, at Level 0, the sensors and actuators that perform the manufacturing tasks are available.

5.3 Backbone-level Monitoring

The monitor nodes can be placed at different locations throughout the ICS infrastructure. Among them, an important and most obvious location is at the backbone level of the network, which connects general-purpose computers at Level 3 to other levels in the infrastructure. If packet sniffing and other security mechanisms were always active in the network at this level, the processing and storage overhead would be significantly higher. In contrast, the monitor node that works in this network segment will be processing EM emission in real time with a fixed processing overhead and no storage requirement.

5.4 Device-level Monitoring

Although the monitoring at the backbone-level of the network allows the observation of the full picture of network behaviour from outside world, it does not enable detecting subtle traffic patterns at the close proximity to different individual ICS devices. Therefore, it is important to deploy and monitor nodes at branches in the ICS network, closer to individual devices of interest. An important advantage the monitor nodes have is that regardless of where exactly they are deployed in the network, i.e., at a busy network backbone or low-traffic branch, they have the same amount of processing and other computational resource usage. Hence, the monitor nodes distributed across the ICS infrastructure will be identical in all aspects.

6 Conclusion and Future Direction

This work demonstrates the potential of using unintentional EM radiation emitted from Ethernet cables as a non-invasive tool for enhancing security and forensic readiness in ICS. By employing EM side-channel monitoring, we can detect network-based attacks without directly interfacing with the ICS infrastructure, making this approach particularly valuable for independent forensic audits. Through targeted frequency identification and EM trace analysis using efficient machine learning models, such as Random Forest classifiers with AdaBoost, our method achieved a high detection accuracy of 99.70%, supporting the feasibility of resource-conscious real-time attack detection in ICS environments.

Machine learning models such as RFC with AdaBoost, MLP, and SVM ensure low processing overhead, making them suitable for resource-constrained environments. Future developments include creating a compact self-contained hardware unit that integrates EM signal capture, embedded processing, and real-time analysis. This scalable and efficient design would enhance ICS security, providing responsive and autonomous intrusion detection capabilities.

Acknowledgments

We acknowledge the funding received from the University of Colombo School of Computing through the Research Allocation for Research and Development, under Grant No: UCSC/RQ/2024/Sec&For/01. This financial support contributed significantly to the success and publication of this research work.

Appendix A

Figure 5 illustrates the high-level view of the ICS infrastructure with the placement of the sniffer and monitor nodes. In this ICS infrastructure, the typical network security and forensic-readiness features are available, such as packet sniffers in strategic locations of the network. However, they are deactivated by default and only enabled on demand whenever a suspicious network-based threat is noticed. The network of the monitor nodes that captures EM radiation data and processes in real time for suspicious activity is placed independently of the ICS infrastructure. They communicate with each other through their own independent network, such as powerline communication, and are capable of directly communicating with security components, such as packet sniffers. These packet sniffers and any other security mechanisms are activated directly by a monitor node upon the detection of suspicious network activity.

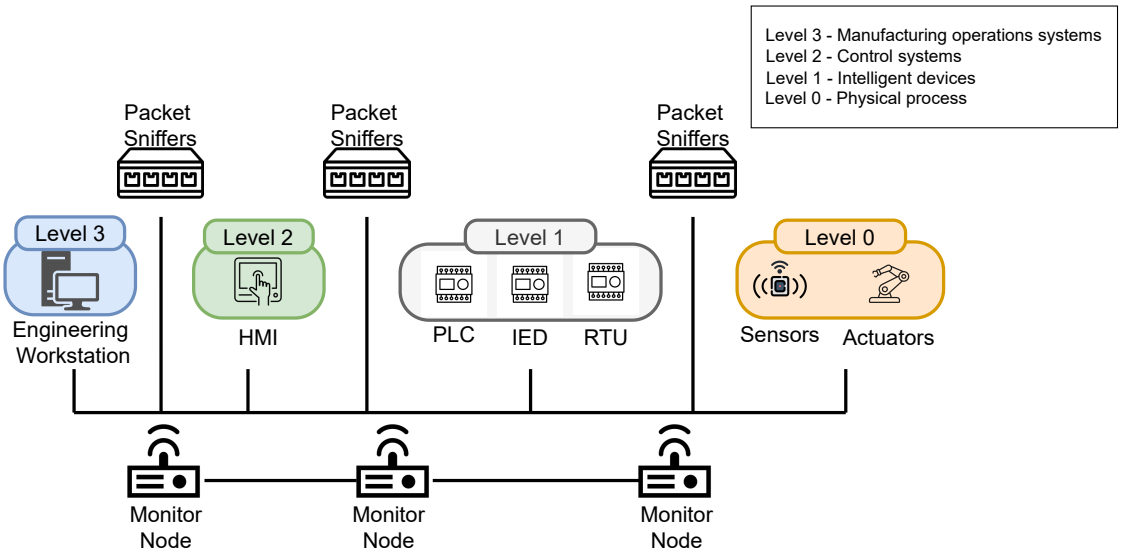


Figure 5: A high-level view of an ICS infrastructure with EM-SCA monitoring mechanism in place.

References

[1] Mohammed Asiri, Neetesh Saxena, Rigel Gjomemo, and Pete Burnap. 2023. Understanding indicators of compromise against cyber-attacks in industrial control systems: a security perspective. *ACM transactions on cyber-physical systems* 7, 2 (2023), 1–33.

[2] Mazen Azzam, Liliana Pasquale, Gregory Provan, and Bashar Nuseibeh. 2023. Forensic readiness of industrial control systems under stealthy attacks. *Computers & Security* 125 (2023), 103010.

[3] Mauro Conti, Denis Donadel, and Federico Turrin. 2021. A survey on industrial control system testbeds and datasets for security research. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2248–2294.

[4] K.A. Dhanya, Sulakshan Vajipayajula, Kartik Srinivasan, Anjali Tibrewal, T. Senthil Kumar, and T. Gireesh Kumar. 2023. Detection of Network Attacks using Machine Learning and Deep Learning Models. *Procedia Computer Science* 218 (2023), 57–66. https://doi.org/10.1016/j.procs.2022.12.401 International Conference on Machine Learning and Data Engineering.

[5] Zaloa Fernandez, Oscar Seijo, Mikel Mendicute, and Inaki Val. 2019. Analysis and Evaluation of a Wired/Wireless Hybrid Architecture for Distributed Control Systems With Mobility Requirements. *IEEE Access* 7 (2019), 95915–95931. https://doi.org/10.1109/ACCESS.2019.2927298

[6] Great Scott Gadgets. 2024. *HackRF One.* https://greatscottgadgets.com/hackrf/one/ Accessed: April 2024.

[7] Davide Giri, Kuan-Lin Chiu, Giuseppe Di Guglielmo, Paolo Mantovani, and Luca P Carloni. 2020. ESP4ML: Platform-based design of systems-on-chip for embedded machine learning. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1049–1054.

[8] Mordechai Guri. 2021. LANTENNA: Exfiltrating Data from Air-Gapped Networks via Ethernet Cables. *CoRR* abs/2110.00104 (2021). arXiv:2110.00104 https://arxiv.org/abs/2110.00104

[9] G. Harris. 2022. PCAP-NG Summary Block Compatibility Specification. https://datatracker.ietf.org/doc/id/draft-gharris-opsawg-pcap-00.html.

[10] Peng Jie and Liu Li. 2011. Industrial control system security. In *2011 third international conference on intelligent human-machine systems and cybernetics*, Vol. 2. IEEE, 156–158.

[11] Christopher Kelly, Nikolaos Pitropakis, Sean McKeown, and Costas Lambrinoudakis. 2020. Testing and hardening IoT devices against the Mirai botnet. In *2020 International conference on cyber security and protection of digital services (cyber security)*. IEEE, 1–8.

[12] Bedeuro Kim, Mohsen Ali Alawami, Eunsoo Kim, Sanghak Oh, Jeongyong Park, and Hyoungshick Kim. 2023. A comparative study of time series anomaly detection models for industrial control systems. *Sensors* 23, 3 (2023), 1310.

[13] Anindya Majumder et al. 2004. Power line communications. *IEEE potentials* 23, 4 (2004), 4–8.

[14] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). https://doi.org/10.1109/MilCIS.2015.7348942

[15] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani. 2023. CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors* 23, 13 (2023). https://doi.org/10.3390/s23135941

[16] Syed Rizvi, Mark Scanlon, Jimmy McGibney, and John Sheppard. 2022. Application of Artificial Intelligence to Network Forensics: Survey, Challenges and Future Directions. *IEEE Access* 10 (10 2022).

[17] Syed Rizvi, Mark Scanlon, Jimmy McGibney, and John Sheppard. 2024. Pushing Network Forensic Readiness to the Edge: A Resource Constrained Artificial Intelligence Based Methodology. In *2024 Cyber Research Conference - Ireland (Cyber-RCI)*. IEEE.

[18] Shakthi Sachintha, Nhien-An Le-Khac, Mark Scanlon, and Asanka P. Sayakkara. 2023. Data Exfiltration through Electromagnetic Covert Channel of Wired Industrial Control Systems. *Applied Sciences* 13, 5 (2023). https://doi.org/10.3390/app13052928

[19] Asanka Sayakkara, Nhien-An Le-Khac, and Mark Scanlon. 2018. Electromagnetic side-channel attacks: potential for progressing hindered digital forensic analysis. In *Companion Proceedings for the ISSTA/ECOOP 2018 Workshops* (Amsterdam, Netherlands) *(ISSTA '18)*. Association for Computing Machinery, New York, NY, USA, 138–143. https://doi.org/10.1145/3236454.3236512

[20] Asanka Sayakkara, Nhien-An Le-Khac, and Mark Scanlon. 2020. EMvidence: A Framework for Digital Evidence Acquisition from IoT Devices through Electromagnetic Side-Channel Analysis. *Forensic Science International: Digital Investigation* 32 (04 2020), 300907. https://doi.org/10.1016/j.fsidi.2020.300907

[21] Matthias Schulz, Patrick Klapper, Matthias Hollick, Erik Tews, and Stefan Katzenbeisser. 2016. Trust The Wire, They Always Told Me! On Practical Non-Destructive Wire-Tap Attacks Against Ethernet. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 43–48.

[22] Chaofan Tang, Lijuan Xu, Bo Yang, Yongwei Tang, and Dawei Zhao. 2023. GRU-based interpretable multivariate time series anomaly detection in industrial control system. *Computers & Security* 127 (2023), 103094.

[23] Ken Yau, Kam-Pui Chow, and Siu-Ming Yiu. 2019. An Incident Response Model for Industrial Control System Forensics Based on Historical Events. In *13th International Conference on Critical Infrastructure Protection (ICCIP) (Critical Infrastructure Protection XIII, Vol. AICT-570)*, Jason Staggs and Sujeet Shenoi (Eds.). Springer International Publishing, Arlington, VA, United States, 311–328. https://doi.org/10.1007/978-3-030-34647-8_16 Part 6: Industrial Control Systems Security.

Advancing Event Reconstruction in Network Forensics: Extending and Evaluating SMB Command Fingerprinting

Jan-Niclas Hilgert
Fraunhofer FKIE
Bonn, Germany
jan-niclas.hilgert@fkie.fraunhofer.de

Martin Lambertz
Fraunhofer FKIE
Bonn, Germany
martin.lambertz@fkie.fraunhofer.de

Abstract

The Server Message Block (SMB) protocol is a vital component of modern internet infrastructure, facilitating file sharing across multiple clients. Due to its widespread use, SMB frequently appears in forensic investigations, particularly in corporate environments. While tools like Wireshark and Zeek enable the extraction of files and creation of logs from SMB traffic, a comprehensive forensic analysis of SMB network traffic, including advanced event reconstruction, remains underexplored. SMB Command Fingerprinting (SCF) is an innovative method that reconstructs file operations and user interactions directly from SMB traffic using meticulously crafted SCF rules. However, prior research demonstrated the feasibility of this method only through a limited case study focused on Windows command-line utilities.

In this work, we enhance the underlying logic of SMB Command Fingerprinting and expand its ruleset to encompass a broader range of actions and applications. To evaluate our approach, we develop a large-scale evaluation framework that automates file and directory operations on an SMB file share while generating a reliable ground truth. Our comprehensive evaluation demonstrates that SCF is able to reconstruct more than 97% of performed operations, even in the worst-case scenario. Additionally, we evaluate SCF's ability to precisely identify the specific application responsible for an action. While minor ambiguities were observed—particularly between closely related operations in `cmd.exe` and PowerShell—these occurred in only a small fraction of operations and did not affect the accurate reconstruction of the underlying event type. Our findings demonstrate the applicability and scalability of the SCF approach for event reconstruction of SMB network traffic, underscoring its potential for broader adoption in network forensics.

CCS Concepts

• **Applied computing** → **Network forensics**.

Keywords

Server Message Block, Network forensics, Event reconstruction

ACM Reference Format:
Jan-Niclas Hilgert and Martin Lambertz. 2025. Advancing Event Reconstruction in Network Forensics: Extending and Evaluating SMB Command Fingerprinting. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3712716.3712723

DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712723

1 Introduction

In digital forensics, the process of *event reconstruction* involves transforming artifacts into specific, high-level events [2]. Basically, its goal is to recreate the sequence of *actions* performed that resulted in the current state of the system. From a formal perspective, this topic has been extensively studied, with research spanning methodologies and frameworks for event reconstruction [2, 3, 6, 18], approaches to timeline reconstruction [4, 7, 13], and methods for event reconstruction in cloud environments [12]. Practical approaches often rely on specific signatures or fingerprints derived from file system timestamps and other metadata [1, 10, 11, 19]. Beyond file system metadata, the use of additional artifacts, such as shellbags, log files, and system calls, has been studied along with their limitations [14, 15, 22].

When evaluating the feasibility of using network traffic for event reconstruction, a clear research gap emerges, especially in the context of tracing specific user actions that led to the creation of the observed traffic. While tools like Wireshark and Zeek enable file extraction and provide a structured overview of network traffic events, they lack the ability to reliably correlate these events with the originating actions on the source system. A notable exception is ClickMiner, a method capable of reconstructing user-browser interactions from network traces [16]. Additionally, Hilgert et al. proposed a method to reconstruct file system operations performed on an SMB share leveraging specifically crafted *SMB Command Fingerprinting rules*. Beyond these, much of the existing research focuses on identifying and fingerprinting applications within network traffic rather than reconstructing high-level events[17, 20, 21].

Consequently, this work aims to further explore the potential of network traffic for event reconstruction, with a particular emphasis on the SMB protocol. The SMB protocol remains widely used providing clients access to a file share, however, has received limited attention from a forensic perspective. Building on the work of Hilgert et al., whose initial results in reconstructing events from SMB network traffic were promising, this study addresses two key gaps in their work: the lack of a comprehensive evaluation of their approach and the unexplored potential for using their method to fingerprint the specific application involved in file system operations. In summary, our contributions include the following:

- **Extension** of the SMB Command Fingerprinting approach and the development of SCF rulesets for `cmd.exe`, PowerShell, and `smbclient`, enabling broader applicability across different applications.

- Development of a **scalable framework** for automating the evaluation of reconstructed events from network traffic, generating reliable ground truth datasets.
- **Comprehensive evaluation** of SCF, including its effectiveness for **cross-application event reconstruction**.

The remainder of the paper is structured as follows: Section 2 provides a detailed overview of the SCF approach introduced by Hilgert et al. and outlines our extensions to enable the creation of more complex rules. Section 3 introduces our evaluation framework and presents the results of a comprehensive evaluation of the extended `cmd.exe` SCF ruleset. In Section 4, we discuss the specific challenges of rule creation for additional applications and evaluate SCF's applicability to these applications, including its use for cross-application event reconstruction. Finally, Section 5 concludes the paper and outlines future research directions.

2 SMB Command Fingerprinting

To interact with an SMB share, an SMB client must send a request specifying the intended operation. The SMB protocol defines a variety of *command types* for this purpose. Examples include `CREATE`, which establishes access to a file or directory; `READ` and `WRITE`, which handle data reading and writing, respectively; and `QUERY_INFO`, which retrieves various types of metadata about a file or directory. Each SMB request is followed by a corresponding SMB response of the same command type. Depending on the command, SMB requests can include various parameters such as access masks, file attributes, or the type of information being requested. Comprehensive details on all supported command types and their parameters are available in Microsoft's official SMB specification [5].

The concept of SMB Command Fingerprinting, introduced in 2024 [9], leverages the dynamic attributes within SMB commands to identify their origin. The authors demonstrated that different API calls in Windows set specific attributes within certain SMB commands, such as the initial `CREATE` request. These attributes allow for inferences about the origin of observed SMB commands in network traffic. Thus, the SCF approach requires unencrypted or decrypted SMB network traffic to access these attributes and command types. To facilitate the identification of potentially unique commands, the authors proposed the concept of *SCF hashes*. An SCF hash is computed for each SMB command, incorporating its command type and any dynamic attributes that can uniquely characterize it, such as access masks in `CREATE` commands. Dynamic values that are not generalizable, such as file paths, are intentionally excluded from the SCF hash computation.

However, performing a file operation on an SMB share does not necessarily result in a single SMB command that can be directly identified using an SCF hash. Instead, a file operation typically generates multiple request-and-response pairs. The authors argue that the sequence of these SMB commands can further be leveraged to identify the originating file operation. To achieve this, they propose combining multiple SCF hashes into an *SCF rule*, which represents not only individual SMB commands but also the specific sequence of commands associated with a particular operation.

To evaluate this approach, the authors examined the `cmd.exe` utility in Windows. They executed various file operations on an SMB share, captured the resulting network traffic, and applied their previously crafted SCF ruleset to the captured data. This method successfully reconstructed the majority of the file operations performed, demonstrating the potential of SCF. However, a comprehensive evaluation of SCF's capabilities and limitations remains an open challenge. For this reason, our first goal is to provide a more in-depth and extensive evaluation of the SCF approach also utilizing `cmd.exe` as an example.

2.1 Extensions to SCF

Before conducting the evaluation, we propose several extensions to SCF that enhance its ability to create more precise and effective rules. These extensions have also been implemented into the existing framework:

- **Skipping SCF Hashes:** In the original implementation, an SCF rule required an exact match of the SMB command sequence. However, we observed instances where additional commands were sent within the sequence. To address this, we introduce a *skipping option* for SCF rules. This parameter specifies the maximum number of unrelated hashes that can occur within the detected sequence of SMB commands without invalidating the rule.
- **Excluded Hashes:** Conversely, we implement an *excluded list*, which defines hashes that must not appear in a matching sequence. This feature helps refine SCF rules by differentiating between sequences that are otherwise similar but diverge based on specific SCF hashes.
- **Extended Command Support:** While the original implementation focused solely on basic SMB commands, we extended SCF to include support for additional commands, such as specific response types. This enhancement enables the creation of more precise and comprehensive rules.
- **Enhanced Matching Logic:** While the specifics of the matching algorithm are beyond the scope of this work, we improved its efficiency and extended its logic to support advanced features, such as the simultaneous matching of multiple rules.

Listing 1: JSON rule for file creation in cmd.

```
{
    "application": "cmd",
    "description": "Create a file with echo",
    "command": "echo >",
    "max_skip": 0,
    "excluded": [],
    "signature": [
        "ebe5eb76fabfe0e41eb63d4fbd06bcd1",
        "CREATE_STATUS_OKAY_HASH",
        "WRITE_HASH",
        "WRITE_STATUS_OKAY_HASH",
        "80c2cc1529acacebb810ec4014119967",
        "QUERY_INFO_STATUS_OKAY_HASH"
    ],
    "parsing": "signature_parsing_cmd_echo_file"
}
```

Listing 1 provides an example of an SCF rule consisting of six SCF hashes for the creation of a file using `cmd.exe`. The parsing method

is used to accurately extract and interpret all relevant information from the corresponding SMB commands.

3 Evaluation

To conduct a comprehensive evaluation of the extended SCF approach, it is essential to perform a large number of operations on the SMB file share. To facilitate this, we first define a set of fundamental file system operations that can be performed using `cmd.exe`:

- **Create and Remove Directories:** These operations can be performed using the `mkdir` and `rmdir` command, respectively.
- **Create and Remove Files:** To create a file, we focus on the usage of the `echo` command in combination with > to store its output in a file. Conversely, files can be removed using the `rm` command.
- **List Directory:** In `cmd.exe`, listing the contents of a directory in is performed using the `dir` command.
- **Upload and Download Files:** For uploading data on the share as well as downloading data from it, the `copy` command can be used.
- **File Content Modification:** For the modification of file content, we will focus on the possibility to append content to it using the `echo` command together with ».
- **Move and Rename File:** On Windows, unlike in default shells in Linux for example, renaming and moving files require two distinct commands in `cmd.exe`. The `ren` command can rename a file or directory but cannot change its location within the file system hierarchy. To move a file to a different location, the `move` command must be used.
- **Move and Rename Directory:** This operation is analogous to moving and renaming a file and also employs the `ren` and `move` commands.
- **Read Content:** To read the content of a file, we utilize the `more` command, which outputs its content to `cmd.exe`.

3.1 Extended SCF Ruleset

Building upon the capabilities of the extended SCF approach, we developed an enhanced set of SCF rules for `cmd.exe`, based on the previous work of the authors, covering the previously described file system operations. To facilitate this process, we utilized the tool by Hilgert et al., which analyzes SMB network captures, computes SCF hashes, and outputs them along with additional contextual information.

During the ruleset creation, we observed that our rules for matching *rename* and *move* operations were identical. This finding suggests that, although implemented as separate commands in `cmd.exe`, the resulting commands transmitted in SMB are the same for both `ren` and `move`. Consequently, we chose to categorize both events, rename and move, as a single reconstructed *move event*. Importantly, this approach does not result in information loss, as the corresponding source and target paths inherently distinguish whether the original operation was a rename or a move.

3.2 Framework

In order to provide a comprehensive evaluation of our extended rule set and the underlying SCF approach in general, we have designed and implemented a scalable framework meeting the following requirements:

- **File Operations:** The framework is capable of performing all previously defined file system operations on a given SMB share using a specified application, for example `cmd.exe`.
- **Consistency:** To facilitate the logical execution of these operations, the framework keeps track of the file system hierarchy it creates.
- **Scalability:** It should be easily possible to create large and random amounts of file system operations.
- **Ground Truth:** Most importantly, the framework provides a ground truth of all its actions, including timestamps.

For the evaluation, we used our framework to execute operations in five distinct phases. Within each phase, the operations were selected with the same probability and each operation was executed approximately 1,000 times[1] to ensure a thorough and comprehensive analysis:

(1) **Initialization:** This phase involves creating files and directories, as well as uploading files to the share.
(2) **Exploration:** In this phase, the share remains unchanged. Operations include viewing files, copying files from the share, and performing directory listings.
(3) **Modification:** During this phase, the share is modified by renaming and moving files or directories, as well as altering file contents by appending data.
(4) **Deletion:** This phase focuses on removing files and directories from the share.
(5) **Combination:** In the final phase, all available operations are executed to simulate diverse activity on the share.

3.3 Results

Table 1 presents the results of our evaluation, detailing the total number of times each operation was performed, the instances where a corresponding rule matched and reconstructed an event, and the cases where the match accurately represented the originating event shown in brackets. Recall is calculated as the proportion of correctly reconstructed events for a specific operation type relative to the total number of operations performed for that type. In general, our results demonstrate that the extended SCF approach and ruleset for `cmd.exe` successfully reconstructed approximately 99.40% of all performed operations.

The lowest reconstruction rate was observed for the *create file* operation, with 29 events not being reconstructed. Upon further inspection, we identified that this was caused by the use of compound requests for create operations. Compound requests combine multiple SMB commands into a single request, resulting in their own unique SCF hash. By updating the SCF ruleset to account for compound requests, it is possible to reconstruct these missing events.

For the *remove file* and *remove directory* operations, the number of reconstructed events exceeded the actual number of files or directories removed. This discrepancy arises from the behavior of `cmd.exe` when deleting directories with content. Although the user's originating action may be a single `rmdir` command, `cmd.exe`

[1] Destructive commands were executed only 500 times to prevent complete destruction of the share.

Table 1: SCF evaluation results for `cmd.exe`.

Operation	Performed	# Matches	Recall (%)
Create Directory	1,061	1,049 (1,049)	99.81
Create File	1,084	1,055 (1,055)	97.32
Upload File	1,107	1,092 (1,092)	98.64
Download File	1,096	1,093 (1,093)	99.73
View File	1,063	1,062 (1,062)	99.90
List Directory	1,059	1,059 (1,059)	100.00
Rename Directory	1,020	-	-
Move Directory	1,057	2,073 (2,073)	99.81
Rename File	1,043	-	-
Move File	1,012	2,045 (2,045)	99.51
Append to File	1,090	1,088 (1,088)	99.82
Remove Directory	442	1,057 (441)	99.78
Remove File	575	1,846 (571)	99.30
Total / Average	**12,709**	**14,519 (12,628)**	**99.36**

executes separate remove operations for each subdirectory and file within the target directory before finally deleting the directory itself. Consequently, our SCF rules reconstruct a remove event for every removed subdirectory and file, leading to a higher count of removal events than the actual user-initiated actions. As a result, the current SCF ruleset cannot distinguish between a user manually deleting the contents of a directory sequentially and the deletion of the entire directory in one operation. However, reconstructing all deletions provides more detailed information about the data that was actually removed from the share.

We also observed that, despite our framework processing operations sequentially, some SMB requests and responses for multiple operations became interleaved within our network capture. This interleaving caused the sequence of SMB commands to deviate from the expected order defined in the SCF ruleset, resulting in mismatches. To address this, we repeated the evaluation with a slightly increased delay between operations, which eliminated this source of mismatches. It is important to emphasize that this behavior does not affect the applicability of SCF rules in scenarios involving multi-client SMB setups or multiple applications accessing the same share. In such cases, network data such as IP addresses and ports can still be used to differentiate between clients and applications, enabling the application of SCF rules separately to each client's TCP stream.

Finally, not all operations involving read access, such as *download file* or *view file*, were reconstructed. A manual inspection of the network traffic revealed that this was due to caching mechanisms. For example, when a file is accessed multiple times within a certain time frame, it is not retransmitted over the network. Consequently, these operations cannot be reconstructed from network traffic, as no corresponding commands exist in it.

3.4 Time Skew

Beyond reconstructing the originating operation from network traffic, a comprehensive event reconstruction should also include the corresponding timestamp of the event. This facilitates the creation of timelines and supports further analysis. When an SCF rule matches an SMB command sequence, it assigns a timestamp to the reconstructed event based on the timestamps of the captured network packets.

Table 4 presents the average offset between the timestamps of reconstructed and their originating events for each operation observed in our evaluation. The average time skew across all operations is 0.551 seconds, indicating that the rules, despite slight delays, provide a reasonable estimate of the actual event times. However, it is important to note that this accuracy is influenced by the characteristics of the network in which the traffic is captured. For example, a congested network can naturally lead to more significant delays in the reconstructed timestamps.

4 Cross-Application Fingerprinting

This section extends the evaluation of SCF to other applications, addressing two key questions: First, we assess whether SCF can reconstruct file operations from other applications by creating and testing new SCF rulesets. Second, we explore whether SCF can distinguish between file operations across different applications by applying multiple SCF rulesets simultaneously.

To explore these aspects, we evaluated two additional applications. First, we selected *PowerShell* due to its similarity to `cmd.exe`, enabling us to assess whether their respective rulesets produce false positives for each other. Second, we chose the `smbclient` utility from Ubuntu 24.04, representing a fundamentally different application running on an entirely different operating system. By examining these examples, we aim to provide a broader and more comprehensive perspective on the applicability of SCF.

4.1 File Operations

For PowerShell, we used the same commands as for `cmd.exe`, but replaced the `rmdir` command with `rm -r -fo`, as `rmdir` is not available in PowerShell.

For `smbclient`, we employed commands corresponding to the file operations being evaluated. However, unlike `cmd.exe` or PowerShell, `smbclient` does not provide separate commands for *rename* and *move* operations. Instead, it offers the `rename` command, which is used for both operations. As a result, moving and renaming a file or directory is treated as a single event in `smbclient`. Furthermore, `smbclient` does not support the use of the `echo` command, which led us to omit these commands for this application.

4.2 Rule Creation

Similar to the extended rule set we developed for `cmd.exe`, we also manually created SCF rules for PowerShell and `smbclient`. Although this work does not provide a detailed comparison of the created rule sets, we highlight several notable peculiarities observed during the rule creation process:

- **Create Directory:** For this operation, PowerShell generates the same SMB commands and hashes as `cmd.exe`. However, unlike `cmd.exe`, it performs a preceding `CREATE` request to check if the specified directory already exists. If a `OBJECT_NAME_NOT_FOUND` error status is received, it sends the corresponding commands to create the directory. This behavior allows for the creation of a unique rule for directory creation in PowerShell. In `smbclient`,

this operation produces unique hashes that can be used directly for the creation of a specific rule.

- **Upload File:** At the beginning, the SMB sequence for uploading a file to an SMB share is identical for `cmd.exe` and PowerShell. However, the ruleset for `cmd.exe` includes additional commands at the end that are absent from the corresponding ruleset for PowerShell.
- **View File:** Unlike `cmd.exe` and PowerShell, which primarily query various types of information for files, `smbclient` queries the `FileAllInformation` class. Since the information class also contributes to an SCF hash, this behavior results in unique SCF hashes that can be utilized for rule creation.
- **Rename and Move Directory:** The rules for this operation overlap between `cmd.exe` and PowerShell. Although for PowerShell our ruleset includes rules with preceding `CREATE` requests that are not present for `cmd.exe`, other rules are identical.
- **Rename and Move File:** A similar behavior involving overlapping rulesets can be observed for these operations in `cmd.exe` and PowerShell.

During the creation of rules for `smbclient`, we observed that the rules for the `rename` operation, which is used for both renaming and moving, are identical for directories and files. Similarly, the rules for viewing a file and downloading it from the share are also the identical. To address this, we created two generalized rules: one for *Move File*, which encompasses events for changing the name or path of both files and directories, and another for *Read File*, which covers operations for viewing and downloading files. To maintain consistency, we refer to these rules as *Move File* and *View File*, respectively, in the evaluation tables.

4.3 Evaluation

For the evaluation, we extended our framework to support the two additional applications, enabling the generation of SMB network captures with corresponding ground truth data. Subsequently, we conducted two evaluations: In the first evaluation, we applied the generated ruleset for each application individually to assess the applicability of SCF and the effectiveness of the ruleset for that specific application. In the second evaluation, we enabled all three rulesets simultaneously to determine whether they interfere with each other and potentially cause false positives.

4.3.1 Single Application. Table 2 provides an overview of the created datasets for the three applications, highlighting key characteristics such as the total number of SMB2 packets and the overall recall of the evaluation. As shown, SCF performed slightly less effectively for PowerShell, with 330 operations not being reconstructed. Table 5 provides a more detailed breakdown of these results. In particular, rename and move directory events, both reconstructed as move events, account for the majority of missing operations, suggesting that the rules for these operations require refinement.

Beyond this, we observed behaviors consistent with the evaluation of `cmd.exe`, including intertwined commands, mismatches caused by caching mechanisms, and the generation of multiple remove events for subdirectories and files within a deleted directory. Moreover, we identified several false positives for the *list directory* event. This issue stems from the nature of the corresponding SCF rule, which only matches a `QUERY_DIRECTORY` command and response that PowerShell sends when listing a directory. However, the same command is also sent in conjunction with other operations, leading to multiple matches. A potential solution to mitigate these false positives is to incorporate this sequence into the corresponding rule for each of the other operations, thereby eliminating the unintended matches.

Table 2: Characteristics of generated and evaluated datasets.

Dataset	Operations	Recall (%)	Time	Packets
cmd.exe	12,709	99.36	01:57h	321,767
PowerShell	12,758	97.41	04:08h	311,556
smbclient	10,417	99.99	05:18h	75,764

For `smbclient`, the results demonstrate that almost all the operations performed were successfully reconstructed by our ruleset. The only exception was a *move file* operation that attempted to move a file to the same directory. This did not result in an actual move operation and, consequently, did not generate the corresponding commands in the network traffic.

Furthermore, our evaluation reveals that the delete operations in `smbclient` differ from those in `cmd.exe` and PowerShell. While the latter applications generate additional remove operations for subdirectories and files within a deleted directory, `smbclient` only sends a single delete command for the corresponding directory.

Although the results are highly accurate, certain limitations remain. As noted earlier, it is not possible to distinguish between merely viewing a file in `smbclient` and downloading it from the server. Similarly, it is not possible to differentiate between a rename and a move operation, as both are executed using the same command in `smbclient`. Additionally, `smbclient` does not provide explicit information to determine whether the operation was performed on a file or a directory. Nevertheless, this information can often be inferred from the file path, as it typically includes the file name and extension.

4.3.2 Cross-Application. Our previous evaluation demonstrated the effectiveness of SCF and our rulesets for two additional applications. To assess its general applicability, we now focus on its performance when applying all three rulesets simultaneously. Table 3 presents the results of SCF when applied to the `cmd.exe` dataset using all rulesets. For each operation, the table shows the number of times a rule corresponding to an application was matched, with the number in parentheses indicating how many matches actually correctly reconstructed an originating event, regardless of the application. Additionally, the column *Multiple* indicates instances, in which both the `cmd.exe` and PowerShell rules matched the same reconstructed event.

For clarity, we calculated two recall values: a *App Recall* reflecting the correct identification of both the event and its originating application and a *Event Recall* focused solely on reconstructing the event type, regardless of the application. *App Recall* measures the proportion of events SCF reconstructed correctly while uniquely identifying the responsible application. The second metric assesses SCF's ability to reconstruct the event type alone. As shown in

Table 3: Evaluation results for `cmd.exe` when applying all created rulesets.

Operation	Performed	# Matches (Correct)				Not Reconstructed	App Recall (%)	Event Recall (%)
		`cmd.exe`	PowerShell	`smbclient`	Multiple			
Create Directory	1,061	1,049 (1,049)	0 (0)	0 (0)	0 (0)	12	99.81	99.81
Create File	1,084	1,055 (1,055)	0 (0)	-	0 (0)	29	97.32	97.32
Upload File	1,107	1,087 (1,087)	10 (10)	0 (0)	0 (0)	10	98.20	99.10
Download File	1,096	1,093 (1,093)	2 (2)	-	0 (0)	1	99.73	99.90
View File	1,063	1,062 (1,062)	0 (0)	0 (0)	0 (0)	1	99.90	99.90
List Directory	1,059	1,059 (1,059)	5,115 (0)	0 (0)	0 (0)	0	100.00	100.00
Rename Directory	1,020	-	-	-	-	0	-	-
Move Directory	1,057	715 (715)	90 (85)	-	1,274 (1,274)	3	34.42	99.86
Rename File	1,043	-	-	-	-	1	-	-
Move File	1,012	32 (32)	118 (118)	-	1,895 (1,895)	9	3.16	99.51
Append to File	1,090	1,088 (1,088)	0 (0)	-	0 (0)	2	99.82	99.82
Remove Directory	442	1,057 (441)	0 (0)	0 (0)	0 (0)	1	99.78	99.78
Remove File	575	1,846 (571)	0 (0)	0 (0)	0 (0)	4	99.30	99.30
Total / Average	**12,709**	**11,143**	**5,335**	**0**	**3,169**	**73**	**72.80**	**99.43**

Table 3 in green, these recall values were identical for most operations, highlighting SCF's ability to reliably identify both operations and their originating applications, even with multiple rulesets enabled.

However, for *move directory* and *move file* operations, these values differ significantly, as shown in red. This is because most of these events were matched by `cmd.exe` and PowerShell rules. Since the rulesets for these operations partially overlap, this behavior is expected. Furthermore, some move operations were incorrectly classified as having been performed by PowerShell including 5 false positives, which were actually misclassified *move file* operations. Moreover, the results show that PowerShell rules matched some of the upload and download file operations missed by the `cmd.exe` ruleset, indicating a need to refine the `cmd.exe` ruleset to address these cases, which currently result in a lower App Recall, shown in orange.

Although these ambiguous and incorrect application classifications limit the precise identification of the application responsible for an event and result in a corresponding lower recall value, the underlying operation itself was still accurately reconstructed with recall values above 99%. Moreover, our evaluation indicates that such incorrect classifications occurred in only a small fraction of the total operations. To provide a comprehensive analysis,

Lastly, a high number of false positives was also observed for the *list directory* operation of PowerShell. As discussed previously, the rule for this operation is relatively short, and the corresponding SMB command sequence is also sent alongside other PowerShell commands. Apparently, this behavior also occurs for `cmd.exe` commands. One potential solution to mitigate these false positives is to incorporate the command sequence into the corresponding `cmd.exe` rules. Alternatively, the rule for this specific operation could be omitted if its forensic value is deemed less critical during an analysis.

Table 7 presents the evaluation of the PowerShell dataset. Similar to the `cmd.exe` dataset, move operations exhibited comparable behavior, but with significantly higher App Recall values. Event Recall values also remained consistently high. Some *create directory* operations were misclassified as `cmd.exe`, which manual inspection attributed to the absence of the initial `CREATE` command PowerShell usually sends to check if a directory exists. This omission was likely caused by caching mechanisms. Nevertheless, all other PowerShell operations were accurately reconstructed.

For `smbclient`, no interference with other SCF rules was observed, as shown in Table 8.

5 Conclusion

Reconstructing events into timelines is crucial in digital forensics. While widely studied in other areas, event reconstruction in network forensics has not been explored as deeply. SMB Command Fingerprinting (SCF), introduced by Hilgert et al., presents a novel method for reconstructing operations on SMB file shares. However, its initial evaluation was limited.

Our work expands SCF with more advanced rules, including a ruleset for the Windows command-line utility `cmd.exe` to cover key file system operations. To assess its scalability, we built a framework that executes operations on an SMB share while generating a ground truth dataset. Our evaluation demonstrated that SCF reconstructed over 99% of operations accurately, confirming its reliability.

To explore the broader applicability of SCF, we extended the evaluation to include PowerShell and `smbclient` by crafting additional rulesets for these applications. Our results were similarly robust, with reconstruction rates exceeding 97% in the worst-case scenario. Furthermore, we evaluated SCF's ability to handle cross-application event reconstruction by applying multiple rulesets simultaneously. While some ambiguity in application identification was observed—primarily between closely related `cmd.exe` and PowerShell operations—the actual events were consistently reconstructed with high accuracy.

Our results demonstrate that SCF is a practical and effective approach for event reconstruction from SMB network traffic. This

is particularly significant given the vast volume of packets generated by SMB, as shown in Table 2, which would otherwise require extensive manual analysis. We contribute our extended rulesets, the framework for dataset generation as well as the dataset itself to the community [8], providing a robust foundation for further rule development, refinement, and research. Future work should explore automated methods for rule generation as well as the possibilities to apply SCF to encrypted SMB traffic. Additionally, it is important to investigate the applicability of the SCF approach to other applications, e.g. the Windows Explorer, as well as entirely different network protocols.

References

[1] Jelle Bouma, Hugo Jonker, Vincent van der Meer, and Eddy Van Den Aker. 2023. Reconstructing Timelines: From NTFS Timestamps to File Histories. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*. 1–9.

[2] Brian Carrier and Eugene Spafford. 2004. An event-based digital forensic investigation framework. *Digital Investigation* (2004).

[3] Brian D Carrier and Eugene H Spafford. 2004. Defining event reconstruction of digital crime scenes. *Journal of Forensic and Sciences* 49, 6 (2004), JFS2004127–8.

[4] Yoan Chabot, Aurélie Bertaux, Christophe Nicolle, and M-Tahar Kechadi. 2014. A complete formalized knowledge representation model for advanced digital forensics timeline analysis. *Digital Investigation* 11 (2014), S95–S105.

[5] Microsoft Corporation. 2024. [MS-SMB2]: Server Message Block (SMB) Protocol Versions 2 and 3. https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/f210069c-7086-4dc2-885e-861d837df688.

[6] Pavel Gladyshev and Ahmed Patel. 2004. Finite state machine approach to digital event reconstruction. *Digital Investigation* 1, 2 (2004), 130–149.

[7] Christopher Hargreaves and Jonathan Patterson. 2012. An automated timeline reconstruction approach for digital forensic investigations. *Digital Investigation* 9 (2012), S69–S79.

[8] Jan-Niclas Hilgert. [n. d.]. Repository for SCF rulesets and dataset framework. https://github.com/fkie-cad/SCF.

[9] Jan-Niclas Hilgert, Axel Mahr, and Martin Lambertz. 2024. Mount SMB. pcap: Reconstructing file systems and file operations from network traffic. *Forensic Science International: Digital Investigation* 50 (2024), 301807.

[10] Joshua Isaac James, Pavel Gladyshev, and Yuandong Zhu. 2011. Signature based detection of user events for post-mortem forensic analysis. In *Digital Forensics and Cyber Crime: Second International ICST Conference, ICDF2C 2010, Abu Dhabi, United Arab Emirates, October 4-6, 2010, Revised Selected Papers 2*. Springer, 96–109.

[11] Sven Kälber, Andreas Dewald, and Felix C Freiling. 2013. Forensic application-fingerprinting based on file system metadata. In *2013 seventh international conference on it security incident management and it forensics*. IEEE, 98–112.

[12] Victor R Kebande and Hein S Venter. 2015. Adding event reconstruction to a Cloud Forensic Readiness model. In *2015 Information Security for South Africa (ISSA)*. IEEE, 1–9.

[13] MNA Khan, Chris R Chatwin, and Rupert CD Young. 2007. A framework for post-event timeline reconstruction using neural networks. *digital investigation* 4, 3-4 (2007), 146–157.

[14] Tobias Latzo. 2021. Efficient fingerprint matching for forensic event reconstruction. In *Digital Forensics and Cyber Crime: 11th EAI International Conference, ICDF2C 2020, Boston, MA, USA, October 15-16, 2020, Proceedings 11*. Springer, 98–120.

[15] Tobias Latzo and Felix Freiling. 2019. Characterizing the limitations of forensic event reconstruction based on log files. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 466–475.

[16] Christopher Neasbitt, Roberto Perdisci, Kang Li, and Terry Nelms. 2014. Clickminer: Towards forensic reconstruction of user-browser interactions from network traces. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 1244–1255.

[17] Shahbaz Rezaei and Xin Liu. 2019. Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine* 57, 5 (2019), 76–81.

[18] Somayeh Soltani and Seyed Amin Hosseini Seno. 2019. A formal model for event reconstruction in digital forensic investigation. *Digital Investigation* 30 (2019), 148–160.

[19] Somayeh Soltani, Seyed Amin Hosseini Seno, and Hadi Sadoghi Yazdi. 2019. Event reconstruction using temporal pattern of file system modification. *IET Information Security* 13, 3 (2019), 201–212.

[20] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2016. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 439–454.

[21] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2017. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security* 13, 1 (2017), 63–78.

[22] Yuandong Zhu, Pavel Gladyshev, and Joshua James. 2009. Using shellbag information to reconstruct user activities. *digital investigation* 6 (2009), S69–S77.

A Appendices

This section presents an evaluation of the average time skew between the originating and the reconstructed event.

A.1 Time Skew

Table 4: Average Time Skew for reconstructed events in `cmd.exe`.

Operation	Average Time Skew (s)
Creation of Directory	0.541
Upload File	0.528
Creation of File	0.530
Download File	0.544
View File	0.553
List Directory	0.560
Move Directory	0.562
Move File	0.563
Appending to File	0.534
Remove Directory	0.576
Remove File	0.574
Average	**0.551**

A.2 Single Application Evaluation

This section presents the SCF evaluation results for PowerShell and `smbclient`, when only the corresponding ruleset is applied.

Table 5: SCF evaluation results for PowerShell.

Operation	Performed	# Matches	Recall (%)
Create Directory	1,061	1,045 (1,045)	98.50
Create File	1,100	1,090 (1,090)	99.10
Upload File	1,090	1,089 (1,089)	99.90
Download File	1,097	1,097 (1,097)	100.00
View File	1,052	1,052 (1,052)	100.00
List Directory	1,059	2,998 (1,044)	98.58
Rename Directory	1,034	-	-
Move Directory	1,057	1,827 (1,824)	87.23
Rename File	1,054	-	-
Move File	1,040	2,080 (2,080)	99.33
Append to File	1,088	1,088 (1,088)	100.00
Remove Directory	447	1,055 (441)	98.66
Remove File	579	1,885 (578)	99.83
Total / Average	**12,758**	**16,306 (12,428)**	**97.41**

A.3 Cross-Application Evaluation

This section presents the SCF evaluation results for PowerShell and `smbclient`, when all created rulesets are applied.

Table 6: SCF evaluation results for `smbclient`.

Operation	Performed	# Matches	Recall (%)
Create Directory	1,108	1,108 (1,108)	100.00
Upload File	1,061	1,061 (1,061)	100.00
Download File	1,060	0 (0)	0.0
View File	1,034	2,094 (2,094)	100.0
List Directory	1,078	1,078 (1,078)	100.00
Rename Directory	1,049	-	-
Move Directory	1,037	-	-
Rename File	1,047	-	-
Moved File	1,034	4,167 (4,166)	99.97
Remove Directory	469	469 (469)	100.00
Remove File	440	440 (440)	100.00
Total / Average	**10,417**	**10,417 (10,416)**	**99.99**

Table 7: Evaluation results for Powershell when applying all created rulesets.

Operation	Performed	# Matches (Correct)				Not Reconstructed	App Recall (%)	Event Recall (%)
		`cmd.exe`	PowerShell	`smbclient`	Multiple			
Create Directory	1,061	15 (15)	1,045 (1,045)	0 (0)	0 (0)	1	98.50	99.90
Create File	1,100	0 (0)	1,090 (1,090)	-	0 (0)	10	99.09	99.09
Upload File	1,090	0 (0)	1,089 (1,089)	0 (0)	0 (0)	1	99.90	99.90
Download File	1,097	0 (0)	1,097 (1,097)	-	0 (0)	0	100.00	100.00
View File	1,052	0 (0)	1,052 (1,052)	0 (0)	0 (0)	0	100.00	100.00
List Directory	1,059	0 (0)	2,998 (1,044)	0 (0)	0 (0)	15	98.59	98.59
Rename Directory	1,034	-	-	-	-	1	-	-
Move Directory	1,057	673 (673)	596 (593)	-	814 (814)	10	28.36	99.47
Rename File	1,054	-	-	-	-	1	-	
Move File	1,040	22 (22)	1,896 (1,896)	0 (0)	162 (162)	13	90.54	99.33
Append to File	1,088	0 (0)	1,088 (1,088)	-	0 (0)	0	100.00	100.00
Remove Directory	447	0 (0)	1,055 (441)	0 (0)	0 (0)	6	98.66	98.66
Remove File	579	0 (0)	1,885 (578)	0 (0)	0 (0)	1	98.66	99.83
Total / Average	**12,758**	**710**	**16,575**	**0**	**976**	**59**	**-**	**-**

Table 8: Evaluation results for `smbclient` when applying all created rulesets.

Operation	Performed	# Matches (Correct)			Not Reconstructed	App Recall (%)	Event Recall (%)
		`cmd.exe`	PowerShell	`smbclient`			
Create Directory	1,108	0 (0)	0 (0)	1,108 (1,108)	0	100.00	100.00
Upload File	1,061	0 (0)	0 (0)	1,061 (1,061)	0	100.00	100.00
Download File	1,060	0 (0)	0 (0)	-	0	-	-
View File	1,034	0 (0)	0 (0)	2,094 (2,094)	0	100.00	100.00
List Directory	1,078	0 (0)	0 (0)	1,078 (1,078)	0	100.00	100.00
Rename Directory	1,049	0 (0)	-	-	0	-	-
Move Directory	1,037	0 (0)	0 (0)	-	0	-	-
Rename File	1,047	-	-	-	-	-	-
Move File	1,034	0 (0)	0 (0)	4,166 (4,166)	1	99.98	99.98
Remove Directory	469	0 (0)	0 (0)	469 (469)	0	100.00	100.00
Remove File	440	0 (0)	0 (0)	440 (440)	0	100.00	100.00
Total / Average	**10,417**	**0**	**0**	**10,416**	**1**	**99.99**	**99.99**

Exploring Dataset Diversity for GenAI Image Inpainting Localisation in Digital Forensics

Matthew Thomson
Edinburgh Napier University
Edinburgh, Scotland, United Kingdom
matthew.thomson@napier.ac.uk

Sean McKeown
Edinburgh Napier University
Edinburgh, Scotland, United Kingdom
s.mckeown@napier.ac.uk

Richard Macfarlane
Edinburgh Napier University
Edinburgh, Scotland, United Kingdom
r.macfarlane@napier.ac.uk

Petra Leimich
Edinburgh Napier University
Edinburgh, Scotland, United Kingdom
p.leimich@napier.ac.uk

Abstract

Generative Artificial Intelligence (GenAI) has significantly increased the sophistication and ease of image tampering techniques, posing challenges for digital forensics in identifying manipulated images. A lack of dataset standardisation hinders the ability to effectively benchmark and compare GenAI inpainting localisation techniques, reducing their reliability in digital forensic applications. This paper aims to address this gap by exploring the need for standardised criteria for datasets in digital forensics for benchmarking detection techniques through preliminary experiments.

To address the limited diversity in existing datasets, a small-scale dataset was developed, consisting of 240 tampered images, 20 masks and 20 authentic images. This dataset includes four subject image classes (animals, objects, persons, scenery) and three inpainting tools (GLIDE, GalaxyAI, Photoshop). The dataset was evaluated against 13 localisation algorithms from the Image Forensics MATLAB Toolbox to determine key components that should be considered in the standardisation of testing environments.

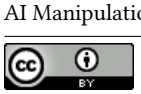
The results show that the images in the animals and persons categories achieved the highest F1-Scores and accuracy over the other classes. Among tools, GLIDE inpainted images were consistently shown to be the most challenging to detect, underscoring the importance of further investigating these images. These findings provide foundational insights for identifying a set of criteria to establish robust testing environments, enabling the development of reliable and accurate GenAI inpainting localisation techniques.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; • **Applied computing** → *Computer forensics.*

Keywords

Digital Forensics, Artificial Intelligence (AI), Generative AI (GenAI), AI Manipulation, Inpainting, Image Forgery Localisation



DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712724

ACM Reference Format:
Matthew Thomson, Sean McKeown, Richard Macfarlane, and Petra Leimich. 2025. Exploring Dataset Diversity for GenAI Image Inpainting Localisation in Digital Forensics. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712716.3712724

1 Introduction

Image tampering has become increasingly sophisticated, posing significant challenges to digital forensics, especially with the introduction of generative artificial Intelligence (GenAI). GenAI is now capable of creating media that appear highly realistic: impossible for the human eye to distinguish tampered from authentic images, and difficult to detect automatically. If undetected, such media could lead to misinformation impacting the investigation process, leading to wrongful convictions or releasing a guilty suspect. These challenges highlight the need for work towards robust and reliable tampering detection and localisation methods [17, 22].

Realistic testing environments are essential to accurately evaluate detection and localisation methods. Without diverse and realistic datasets, the accuracy and reliability of detection and localisation methods cannot be effectively evaluated. To overcome these challenges, this study aims to identify the similarities and patterns within subject image classes and tools that contribute to the performance of detection and localisation techniques through a quantitative evaluation. The experimentation considered the subject class of an image, as the variation in image complexities and textures can significantly affect performance. Additionally, the inpainting tool utilised introduces differing image attributes due to its divergent image manipulation processes. Through preliminary experimentation, this work demonstrates how these components impact localisation results, emphasising the importance of standardised and realistic testing environments.

To address the challenges, we explore the following research questions:

- **RQ1:** How do different image classes, such as animals, objects, persons, and scenery, affect the performance of tampering localisation algorithms?
- **RQ2:** How do different inpainting tools, such as GLIDE, GalaxyAI, and Photoshop, impact tampering localisation performance?

- **RQ3:** What additional considerations should be addressed to establish a comprehensive set of criteria for the standardisation of a realistic testing environment for digital forensics purposes?

Our evaluation lays the foundation towards establishing standardised criteria for testing environment datasets within digital forensics. Developing a standardised dataset will enable consistent testing, improving the reliability and applicability of GenAI image detection and localisation techniques in forensic investigations.

2 Background and Related Work

In digital forensics, image tampering or forgery refers to the intentional modification of the content to mislead the viewer. Common image tampering techniques include copy-move and splicing manipulations [24]. However, the integration of GenAI into these processes has significantly increased their sophistication, making them particularly dangerous. Alongside this, image tampering is now more accessible and requires no prior expertise, widening its potential for use. GenAI for image manipulation can typically be divided into two main categories: fully generated and partially generated content. Partially AI-generated or tampered images involve the use of GenAI models to enhance traditional manipulation techniques [23]. One widely accessible and commercialised image tampering technique is inpainting, where elements of interest, such as people or weapons, can be removed from the scene to alter the image's context [23]. Malicious image tampering such as this can be used to obscure digital evidence, posing significant challenges for forensic investigators [25]. The ability of GenAI models to generate highly realistic and coherent background textures from object removal makes detecting tampered regions particularly difficult.

While new detection and localisation techniques are being developed, their evaluation is often limited by the lack of standardised criteria for testing environments. For instance, Li et al. [12] found a substantial drop in performance when testing their transformer-based detection technique against unseen datasets. Similarly, Patel et al. [19] explored the use of machine-based techniques, specifically Dense CNNs, and evaluated their approach using the Deepfake Images Detection and Reconstruction Challenge dataset. Patel et al.'s study reported the accuracy metric ranging from 94.67% to 99.33%. However, when testing the generalisation capabilities of the proposed model, the accuracy decreased significantly to only 77%. This highlights the importance of standardised datasets and testing environments in determining a technique's applicability to digital forensics. Some techniques may perform better against certain manipulation types or GenAI models than others and this is hard to determine without a wide scale dataset for benchmarking. This lack of standardisation limits the ability to assess the accuracy and reliability of detection and localisation techniques across diverse scenarios. Forensic analysts often encounter highly specific manipulation types that correspond to different image classes, such as face swaps in social media or object removal in CCTV [16]. A testing environment that allows for divided classes, such as persons, objects, and scenery, would enable forensics analysts to evaluate whether the nature of the replacement influences localisation performance. Identifying these impacts allows examiners to tailor their analysis strategies to specific content types.

Despite the number of datasets available in the field, their relevance becomes outdated very quickly, with new models constantly being developed. Furthermore, several of the datasets tend to be for very specific problems or areas rather than providing a diverse variety of options. Datasets such as *Artifact* [20] include images from a variety of models and classes, but their exclusive focus on art styles and the absence of options to isolate and test specific classes limits their applicability for forensic scenarios. The Hierarchical Fine-grained (*HiFi-IFDL*) dataset [8] addresses additional manipulation types beyond just fully AI-generated or inpainting, but omits commonly used commercial models such as Adobe Firefly, Midjourney, DALL-E, or newer versions of Stable Diffusion. Conversely, datasets such as *CIFAKE* [5] have a large variety of classes, but are limited by very small image resolutions of 32x32 or 64x64 compared to the typical Stable Diffusion output of 512x512.

This work aims to address such gaps by exploring the influence of dataset components, such as a variety of image classes and tools, on localisation performance. By identifying foundational components that should be dictated in standardised criteria for datasets, our work contributes to developing robust and standardised testing environments for digital forensics applications.

3 Methodology

To evaluate the requirements for a standardised testing environment for GenAI localisation techniques, we assess the impact of dataset components through two primary experiments. The first experiment focuses on the influence that the image subject classes, animals, objects, persons, and scenery, can have on the localisation performance. The second experiment evaluates the impact of the inpainting tool used for tampering, where three tools, GLIDE, GalaxyAI, and Adobe Photoshop, are chosen for their distinct approaches to inpainting. The focus on localisation is due to the availability of the Image Forensics MATLAB Toolbox [28], which contains a range of localisation algorithms. This approach allows for the evaluation to prioritise the impact of dataset components, rather than specific detection or localisation techniques. To conduct these experiments, a comprehensive dataset consisting of authentic images, tampered images, and masks, must be created, alongside adaptations to the Image Forensics MATLAB Toolbox.

3.1 Dataset Creation

A small-scale dataset consisting of 20 authentic images with 20 corresponding edit localisation masks, each used to generate 12 tampered images (totalling 240), with the tampered content being created using the GenAI technique of inpainting. This tampering method alters an image by reconstructing segments to conceal elements, often deceiving viewers [23]. The inpainting process is displayed in Figure 1, where the cow is removed from the image. This technique was selected due to its availability and ease of use, alongside similarities to the traditional technique of splicing.

Figure 1: Example of inpainting using GalaxyAI on an image of a cow.

The authentic images were sourced from the Microsoft Common Objects in Context (*MS COCO*) dataset [13], which contains 90 categories of images. Four categories, animals, objects, persons and scenery, were selected to include a variety of image options and complexities. Within each subject class, five images were selected from categories of similar classes. For example, dog, cat, bear, chicken and cow were combined to form the "animals" class. The four selected image subject classes represent a variety of the 90 *MS COCO* categories, enabling the combination of multiple subcategories to provide a more diverse evaluation.

To manipulate the images with the chosen inpainting techniques, each authentic image was tampered with four times using the three tools. The first method involved Adobe Firefly, which was accessed through the desktop Adobe Photoshop application [1]. Subject masks in the images were selected using the semi-automatic Magic Selection Tool, and the prompt "remove" was used to guide the inpainting process. Photoshop's GenAI tool generates three variations per run, so the process was repeated twice to produce sufficient outputs, with the first four results selected. The second method used GLIDE, which was run locally using the code available from the glide-text2im project [18]. To allow for automation, the Jupyter notebooks contained within the project's source code were converted into a Python script. The script requires the input of the authentic image, a reference mask, and the "remove" prompt. Masks initially created from the Photoshop process were reused to ensure consistency across tools. The third method used GalaxyAI, an object editor and removal feature on Galaxy phones [21]. The masks were manually drawn over the images to replicate those used in Photoshop and GLIDE. This tool uses an erase button to remove the selected masks, no prompt is required.

For each category, five masks corresponding to the authentic images were created, resulting in twenty masks across the dataset. These masks, initially created in Adobe Photoshop, were used throughout for the tampering reference masks. The full dataset consists of 3 tools, 4 classes, and 20 tampered images each, resulting in 240 tampered images total (3x4x20).

3.2 Experiment Setup

The experiments were conducted using the Image Forensics MATLAB Toolbox [28], which was created primarily for splicing localisation. To improve upon the work presented in [28], our experimentation includes images with inpainting modification rather than splicing. The evaluation focuses on the differing performance across classes and tools rather than each algorithm's performance. Additionally, the toolbox was used to allow for multiple localisation methods to be run on the images, providing results specific to the tampered images rather than one specific method of localisation. The toolbox contains a total of 16 algorithms, of which 13 were selected for this experiment as they can be directly applied without further modifications to the code being necessary. The selected algorithms are: ADQ1 [14], ADQ2 [3], ADQ3 [2], BLK [11], CAGI [9], CFA1 [7], CFA3 [6], DCT [27], ELA [10], NADQ [4], NOI1 [15], NOI4 [26], and NOI5 [29].

To adapt the toolbox, the EvaluateAlgorithm.m script was modified to allow for multiple algorithms to be passed at once, and the ExtractMaps.m script was modified to allow for an option accommodating the specific mask and authentic image setup that this experiment consisted of. Additional MATLAB scripts were created to process the probability maps generated by the toolbox, which represent the likelihood of pixel regions being tampered with. The probability map output was then normalised to values between 0 and 1 across the dataset for each algorithm. The confusion matrix values are calculated using the normalised probability map of the corresponding images as the threshold. As shown in Table Table 1, where TR is the Tampered Result probability map value and AR is the Authentic Result probability map value, if the TR is more than or equal to the AR then it is deemed as tampered with. Then the actual mask is used to determine if this was a correct identification. Using the authentic image as the threshold in this experiment allows for the evaluation of differences across image classes and inpainting tools. However, it is important to note that this approach would not be possible in real life forensics scenarios; it is used here for the purpose of identifying key components and artefacts of images for the localisation of GenAI manipulation, particularly inpainting.

Table 1: Confusion Matrix value breakdown, where TR is the Tampered Result probability map value and AR is the Authentic Result probability map value

	Mask	Predicted
True Positive (TP)	True	TR >= AR
False Positive (FP)	False	TR >= AR
True Negative (TN)	False	TR < AR
False Negative (FN)	True	TR < AR

To evaluate the influence that the image classes and tools have on localisation methods, the data in the .mat files were organised with a MATLAB script that restructured them based on their class and tool fields. This allowed for the evaluation metrics to be calculated on the full dataset, as well as per class and per tool.

For the visual analysis, heatmaps were created using the MATLAB imagesc function, which creates a colour map from the values. This heatmap was saved to a pdf for all the images within each algorithm. The probability maps passed to this function were the binary threshold maps from the predicted function within Table 1. Full implementation details are available on GitHub[1].

3.3 Evaluation Metrics

The evaluation metrics selected for this experiment include the confusion matrix alongside the accuracy, precision, recall and F1-scores. The discussion will mainly focus on accuracy and F1-score due to their relevance in digital forensics investigations. Accuracy

[1]https://github.com/MatthewT0/GenAI-Image-Forensics-Toolbox

assesses the localisation methods' ability to correctly identify tampered and untampered areas, which is vital for ensuring reliability within digital forensics investigations. Using accuracy alone can provide misleading results, which is why the F1-score will also be used. This addresses accuracy limitations by balancing precision and recall, offering a robust metric for evaluating the performance in scenarios where both false positives and false negatives carry significant weight. This balance is crucial in digital forensics investigations, where undetected tampered images or images falsely identified to be tampered with when they are authentic could compromise the integrity of a case. By combining these metrics, we ensure a comprehensive evaluation of localisation methods.

4 Results and Discussion

The results from both experiments are displayed in three ways. A visual heatmap shows the highlighted tampered and untampered regions identified by the probability map. Secondly, a bar chart is used to represent similarities grouped by different dataset components, and finally a table presents all evaluation metrics for comparison purposes. For demonstration purposes, the heatmaps from the NOI4 algorithm were used due to having particularly strong results. These heatmaps are the binary decision values once the normalised probability maps from the tampered image and authentic reference image are compared. During the review process, a manual inspection of the output heatmaps was performed to identify images that could be detected through the human eye. These were selected based on obvious mask detection without looking at the authentic or tampered images.

4.1 Experiment 1: Image Classes

Experiment 1 explored the influence of the subject class on the performance of various localisation algorithms. The dataset was divided into four classes of animals, objects, persons, and scenery. The probability map from the animals subject class image once evaluated against the threshold probability map values can be seen in Figure 2. The figure is from the same example shown in Figure 1, where the cow is removed from the image. The heatmap highlights the detected tampered areas in yellow, where the cow is correctly identified and the sky is incorrectly identified as tampered. Through the visual heatmaps inspection there were notably more identifications in the animals image class, with 32% of all visual identifications being from this class.

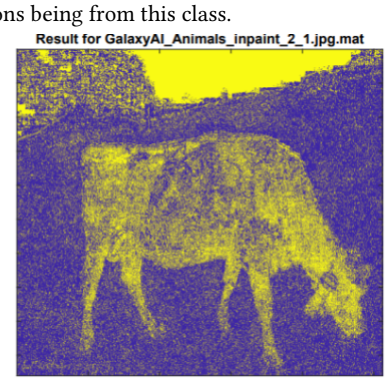


Figure 2: NOI4 localisation result, where yellow is tampered.

The F1-scores across classes are illustrated as a bar chart in Figure 3, where a consistent pattern can be identified between the animals and persons classes, achieving higher scores than objects or scenery across most algorithms. This suggests that the localisation methods are more effective in identifying the tampered regions within these two classes. In contrast, the objects and scenery classes show notably lower scores averaging at around half the other two classes' F1-scores, at 10% compared to 20%. The consistent underperformance in objects and scenery categories highlights a potential weakness in the handling of these image classes, indicating a need for future work. The F1 scores across the evaluated categories highlight the importance in having a diverse dataset of varying classes. However, to better determine which image attributes in the class cause such opposing F1-scores, a more in-depth analysis will need to be conducted.

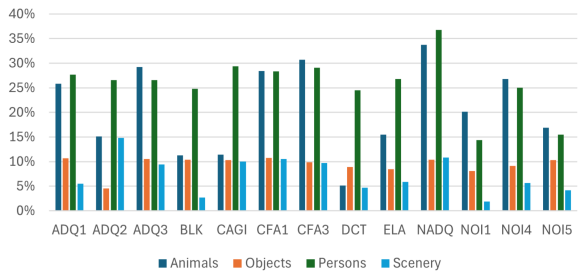


Figure 3: Average F1-score across algorithms, grouped by the class.

Table 2 presents the metrics for the full dataset as the baseline and the difference to each of the classes. The results highlight key trends in how the localisation methods perform across various image classes.

The objects class presents the highest TN and lowest FN results, demonstrating that the algorithms against this class are particularly effective in correctly identifying untampered areas within an image. In contrast, the precision and F1-score ratios decrease substantially, highlighting the difficulty of the classes in balancing the correct identifications of tampered regions with incorrect untampered regions. The results suggest that images within the objects class are more likely to be classified as untampered, regardless of whether they have been or not. Further investigation into the specific image attributes which cause the inaccuracy in image tampering detection must be conducted before a definitive analysis can be performed. However, it is important to note that other factors, such as smaller mask region, could be a result of inaccurate localisation.

On the contrary, the persons class can be seen to have the highest precision and F1-scores, but the lowest accuracy. This demonstrates the persons' class strength in correctly identifying tampered regions whilst struggling to identify the untampered areas correctly. The animals and persons classes share similar patterns in their F1-scores as previously highlighted, which is further reflected in Table 2. Both classes show confusion matrix values that deviate from the baseline in the same direction, increasing or decreasing together. This suggests that the two classes share similar underlying attributes that influence the localisation algorithms in comparable ways.

Table 2: Baseline metrics for the full dataset of tampered images compared to the per-class (60 images per class) differences from the baseline. Negative values indicate a decrease from the baseline metrics, while positive values represent an increase from the baseline metrics.

		Classes			
	Baseline	Animals	Objects	Persons	Scenery
TP	4.65	1.86	-2.67	3.75	-2.94
TN	55.13	-1.30	6.98	-8.98	3.30
FP	31.15	-5.03	1.28	-0.74	4.48
FN	9.06	4.47	-5.59	5.97	-4.84
Accuracy	59.79	0.56	4.31	-5.23	0.36
Precision	13.20	6.72	-7.16	8.36	-8.84
Recall	33.92	-1.44	2.48	1.92	-5.02
F1-Score	17.63	3.14	-8.22	8.16	-10.26

In summary, the results from the classes analysis highlight important patterns in how localisation methods perform across different image classes. This underscores the importance of a standardised and realistic testing environment through a set of clearly defined criteria. The similarities and patterns discovered between the persons and animals classes, compared to the objects and scenery classes, demonstrate the importance of rigorous testing to identify the features that influence the localisation performance. Establishing a set of criteria is essential for creating datasets that accurately reflect real-world scenarios, enabling the evaluation and creation of digital forensics techniques to overcome the rise of GenAI image manipulation.

4.2 Experiment 2: Tools

This experiment examines the impact of differing inpainting tools on the performance of localisation methods. The tools GLIDE, GalaxyAI, and Adobe Photoshop were selected for this evaluation due to their distinct approaches to tampering, which could introduce varying characteristics in tampered regions. The goal of this experiment is to determine whether the tool used affects the localisation ability, highlighting the importance of tool diversity in the creation of realistic datasets.

Across the tools utilised, a visual heatmap inspection was performed where Adobe Photoshop generally showed more obvious tampering indications, consisting of 44.62% out of identifications. However, some occasions were identifiable due to the mask edges being flagged as not tampered with and a large distribution of tampered indications falling within the mask region. Visual examples of manually identified tampering can be seen in Figure 4, where examples are shown of GLIDE, GalaxyAI and Photoshop respectively. Out of the manually identified masks, the GalaxyAI masks were more obvious to notice when they occurred due to more correct TN pixels being identified, but there were more Adobe Photoshop instances.

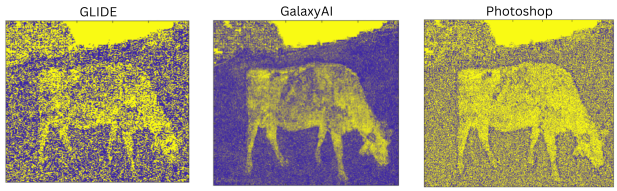


Figure 4: NOI4 visual heatmaps for the inpainting tools.

The accuracy results, as shown in Figure 5, reveal that GalaxyAI consistently demonstrates a higher performance compared to Photoshop and GLIDE's accuracy, averaging at 63%. This suggests that GalaxyAI tampered images have more detectable artefacts within the images, possibly due to their built-in watermarking. Photoshop and GLIDE's accuracy are generally lower across algorithms, demonstrating that these images are harder to detect. For GLIDE, the performance difference appears to be influenced by the lower resolution of GLIDE tampered images compared to the other tools, as the GLIDE code downscales the image resolution. The resolution disparity strongly suggests a contributing factor to the decreased localised performance. The underlying reason behind GalaxyAI's higher accuracy and GLIDE's lower accuracy would require further investigation to fully understand the contributing factors. In the case of Photoshop, the factors being its lower accuracy remain unclear and further experimentation would be beneficial in determining what components make these tampered images harder to detect.

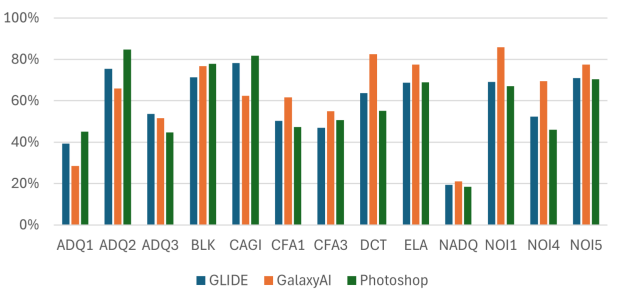


Figure 5: Average accuracy across algorithms, grouped by the tool.

The baseline metrics for the full dataset and the difference between ratios from the baseline to the classes can be seen in Table 3. GalaxyAI differentiates the most from the baseline, with the highest TN, accuracy, and precision, as well as the lowest FP. These results indicate that the localisation algorithms applied to GalaxyAI tampered images were generally more effective at correctly identifying untampered areas, averaging at 63% accuracy.
For all evaluation metrics except precision, GLIDE and Photoshop deviate from the baseline in the same direction. This consistent deviation in most metrics indicates similarities in the localisation difficulty of these tools.

Table 3: Baseline metrics for the full dataset of tampered images compared to the per-tool differences from the baseline. Negative values indicate a decrease from the baseline metrics, while positive values represent an increase from the baseline metrics.

		Tools		
	Baseline	GLIDE	GalaxyAI	Photoshop
TP	4.65	0.17	-0.26	0.10
TN	55.13	-1.57	3.19	-1.62
FP	31.15	1.56	-3.18	1.63
FN	9.06	-0.15	0.26	-0.11
Accuracy	59.79	-1.41	2.93	-1.52
Precision	13.20	-0.43	1.79	0.44
Recall	33.92	1.16	-1.91	0.75
F1-Score	17.63	-0.23	-1.04	-0.45

The conducted experiment builds upon the findings from the subject class analysis by focusing on the tools used to generate the tampered images rather than the content itself. From this, Photoshop provided the highest number of human eye detections from the output heatmaps, whilst GalaxyAI should have the most distinctive heatmaps. Additionally, GalaxyAI consistently demonstrated higher accuracy than the other two tools across algorithms, indicating more identifiable attributes in these images. Furthermore, GalaxyAI also showed the most significant deviations from the baseline with a 3.19% increase to TN and a 3.18% decrease from FP. This suggests that the localisation algorithms were typically producing more non-tampered classifications within their probability maps.

5 Conclusion

Many GenAI tampering datasets do not include a variety of manipulation types, classes or tools within them, creating a limited testing environment for digital forensics. This can result in the evaluation of detection and localisation techniques suffering from overfitting and generalisation, leading to their performance decreasing greatly when applied to real-world scenarios. An example of this is when GenAI localisation techniques are used in areas that differ from its original testing environment, such as a tampering technique that is used for detecting the alteration of people being used on identifying removed objects. This paper explored the need for standardised testing environments to enhance the performance of upcoming detection and localisation techniques being developed.

Addressing RQ1, it was found that certain classes had a higher level of influence on the performance of the localisation algorithms compared to others. Based upon the manual heatmap inspection, the animals class had more notable identifications, being 31.97% of the overall identifications. This class, along with the persons class, illustrated key patterns, consistently scoring higher F1-score by almost double the other classes analysed, at 20% compared to 10%. The identified similarities emphasise the importance of the inclusion of a variety of classes within the testing environment for digital forensics, where analysts often face a wide range of image classes from animals to objects.

The evaluation aimed at addressing RQ2 showed that Photoshop had the highest notable visual heatmaps, being 44% of the overall identifications. However, the GalaxyAI manual identifications were clearer with more correct TN values. Furthermore, GLIDE inpainted images were the most challenging for algorithms to detect, with consistently lower accuracy compared to GalaxyAI and Photoshop. This highlights the need for further attribute testing on the impact of lower resolution, which could be causing the lower accuracy for GLIDE inpainted images.

Having a realistic dataset that can cover a range of classes and tools will provide a robust testing environment for the development of techniques to counteract AI-based manipulation and cover the nature of media that forensic analysts handle. Without standardised criteria for datasets, the detection and localisation methods being developed cannot be accurately and rigorously tested to ensure their reliability in forensic investigations.

Additional components that should be further investigated for inpainting detection and localisation include the mask size, image complexity, and image resolution. These components alongside the underlying artefacts, such as watermarking in GalaxyAI, highlight the requirements for establishing a comprehensive set of criteria as raised in RQ3.

6 Future Work

The work conducted was performed at a small-scale level, and while it provided preliminary results and patterns, a larger dataset is necessary to validate these and establish the criteria for standardisation of datasets. Expanding the dataset would facilitate an extensive and comprehensive analysis of image attributes and dataset components.

Although a manual visual inspection of the heatmaps was conducted, future experimentation could include automating this process by analysing the pixel distribution of identified tampered areas, minimising any implication of human error. Furthermore, the analysis was conducted using a single threshold option, specifically the authentic image probability map. Whilst the data collected from the authentic image as the threshold provided interesting results, this is not feasible in real-world examples where the authentic image and prior knowledge are unknown. Therefore, additional threshold tests to determine if blind localisation is possible would be beneficial in improving forensic applicability. Further analysis into an error percentage margin around the threshold value should be investigated to determine if the threshold is suitable.

Lastly, additional testing on the localisation abilities of further file types, such as PNGS and WEBPs, would be invaluable in determining the various formats required for a diverse dataset. The MATLAB toolbox was specifically created for JPG images, although some algorithms allow for PNGs. However, MATLAB lacks official support for WEBP image format, so alternative solutions would be required to address this. File type testing may include various conversion approaches, including converting each of the three aforementioned file types into every other type to assess whether the direction of image conversion impacts the outcomes. Additionally, comparing double-compressed JPG images and WEBP images may provide insight to determine whether any additional components of the images are being identified from the localisation algorithms.

References

[1] Adobe. 2024. Adobe Firefly - Free Generative AI for creatives. https://www.adobe.com/uk/products/firefly.html

[2] Irene Amerini, Rudy Becarelli, Roberto Caldelli, and Andrea Del Mastio. 2014. Splicing forgeries localization through the use of first digit features. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 143–148. https://doi.org/10.1109/WIFS.2014.7084318 ISSN: 2157-4774.

[3] Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. 2011. Improved DCT coefficient analysis for forgery localization in JPEG images. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2444–2447. https://doi.org/10.1109/ICASSP.2011.5946978 ISSN: 2379-190X.

[4] Tiziano Bianchi and Alessandro Piva. 2012. Image Forgery Localization via Block-Grained Analysis of JPEG Artifacts. *IEEE Transactions on Information Forensics and Security* 7, 3 (June 2012), 1003–1017. https://doi.org/10.1109/TIFS.2012.2187516 Conference Name: IEEE Transactions on Information Forensics and Security.

[5] Jordan J. Bird and Ahmad Lotfi. 2024. CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images. *IEEE Access* 12 (2024), 15642–15650. https://doi.org/10.1109/ACCESS.2024.3356122 Conference Name: IEEE Access.

[6] Ahmet Emir Dirik and Nasir Memon. 2009. Image tamper detection based on demosaicing artifacts. In *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE, 1497–1500. https://doi.org/10.1109/ICIP.2009.5414611 ISSN: 2381-8549.

[7] Pasquale Ferrara, Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. 2012. Image Forgery Localization via Fine-Grained Analysis of CFA Artifacts. *IEEE Transactions on Information Forensics and Security* 7, 5 (Oct. 2012), 1566–1577. https://doi.org/10.1109/TIFS.2012.2202227 Conference Name: IEEE Transactions on Information Forensics and Security.

[8] Xiao Guo, Xiaohong Liu, Zhiyuan Ren, Steven Grosz, Iacopo Masi, and Xiaoming Liu. 2023. Hierarchical Fine-Grained Image Forgery Detection and Localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3155–3165. https://openaccess.thecvf.com/content/CVPR2023/html/Guo_Hierarchical_Fine-Grained_Image_Forgery_Detection_and_Localization_CVPR_2023_paper.html

[9] Chryssanthi Iakovidou, Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2018. Content-aware detection of JPEG grid inconsistencies for intuitive image forensics. *Journal of Visual Communication and Image Representation* 54 (July 2018), 155–170. https://doi.org/10.1016/j.jvcir.2018.05.011

[10] Neal Krawetz. 2007. A Picture's Worth: Digital Image Analysis and Forensics. In *A Picture's Worth: Digital Image Analysis and Forensics*. Black Hat USA, USA, 1 to 31. https://blackhat.com/presentations/bh-dc-08/Krawetz/Whitepaper/bh-dc-08-krawetz-WP.pdf

[11] Weihai Li, Yuan Yuan, and Nenghai Yu. 2009. Passive detection of doctored JPEG image via block artifact grid extraction. *Signal Processing* 89, 9 (Sept. 2009), 1821–1829. https://doi.org/10.1016/j.sigpro.2009.03.025

[12] Yuanman Li, Liangpei Hu, Li Dong, Haiwei Wu, Jinyu Tian, Jiantao Zhou, and Xia Li. 2024. Transformer-Based Image Inpainting Detection via Label Decoupling and Constrained Adversarial Training. *IEEE Transactions on Circuits and Systems for Video Technology* 34, 3 (March 2024), 1857–1872. https://doi.org/10.1109/TCSVT.2023.3299278 Conference Name: IEEE Transactions on Circuits and Systems for Video Technology.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[14] Zhouchen Lin, Junfeng He, Xiaoou Tang, and Chi-Keung Tang. 2009. Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis. *Pattern Recognition* 42, 11 (Nov. 2009), 2492–2501. https://doi.org/10.1016/j.patcog.2009.03.019

[15] Babak Mahdian and Stanislav Saic. 2009. Using noise inconsistencies for blind image forensics. *Image and Vision Computing* 27, 10 (Sept. 2009), 1497–1503. https://doi.org/10.1016/j.imavis.2009.02.001

[16] Marie-Helen Maras and Alex Alexandrou. 2019. Determining authenticity of video evidence in the age of artificial intelligence and in the wake of Deepfake videos. *The International Journal of Evidence & Proof* 23, 3 (July 2019), 255–262. https://doi.org/10.1177/1365712718807226 Publisher: SAGE Publications Ltd.

[17] Mekhail Mustak, Joni Salminen, Matti Mäntymäki, Arafat Rahman, and Yogesh K. Dwivedi. 2023. Deepfakes: Deceptions, mitigations, and opportunities. *Journal of Business Research* 154 (Jan. 2023), 113368. https://doi.org/10.1016/j.jbusres.2022.113368

[18] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2022. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. https://doi.org/10.48550/arXiv.2112.10741 arXiv:2112.10741.

[19] Yogesh Patel, Sudeep Tanwar, Pronaya Bhattacharya, Rajesh Gupta, Turki Alsuwian, Innocent Ewean Davidson, and Thokozile F. Mazibuko. 2023. An Improved Dense CNN Architecture for Deepfake Image Detection. *IEEE Access* 11 (2023), 22081–22095. https://doi.org/10.1109/ACCESS.2023.3251417 Conference Name: IEEE Access.

[20] Md Awsafur Rahman, Bishmoy Paul, Najibul Haque Sarker, Zaber Ibn Abdul Hakim, and Shaikh Anowarul Fattah. 2023. Artifact: A Large-Scale Dataset With Artificial And Factual Images For Generalizable And Robust Synthetic Image Detection. In *2023 IEEE International Conference on Image Processing (ICIP)*. 2200–2204. https://doi.org/10.1109/ICIP49359.2023.10222083

[21] Samsung. 2024. How to use Generative photo editing with Galaxy AI. https://www.samsung.com/sg/support/mobile-devices/how-to-use-generative-photo-editing-on-the-galaxy-s24/

[22] Maria-Paz Sandoval, Maria de Almeida Vau, John Solaas, and Luano Rodrigues. 2024. Threat of deepfakes to the criminal justice system: a systematic review. *Crime Science* 13, 1 (Nov. 2024), 41. https://doi.org/10.1186/s40163-024-00239-1

[23] Deependra Kumar Shukla, Abhishek Bansal, and Pawan Singh. 2024. A survey on digital image forensic methods based on blind forgery detection. *Multimedia Tools and Applications* (Jan. 2024). https://doi.org/10.1007/s11042-023-18090-y

[24] Satyendra Singh and Rajesh Kumar. 2024. Image forgery detection: comprehensive review of digital forensics approaches. *Journal of Computational Social Science* (April 2024). https://doi.org/10.1007/s42001-024-00265-8

[25] Shobhit Tyagi and Divakar Yadav. 2023. A detailed analysis of image and video forgery detection techniques. *The Visual Computer* 39, 3 (March 2023), 813–833. https://doi.org/10.1007/s00371-021-02347-4

[26] Jonas Wagner. 2015. Noise Analysis for Image Forensics. http://29a.ch/2015/08/21/noise-analysis-for-image-forensics

[27] Shuiming Ye, Qibin Sun, and Ee-Chien Chang. 2007. Detecting Digital Image Forgeries by Measuring Inconsistencies of Blocking Artifact. In *2007 IEEE International Conference on Multimedia and Expo*. IEEE, 12–15. https://doi.org/10.1109/ICME.2007.4284574 ISSN: 1945-788X.

[28] Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2017. Large-scale evaluation of splicing localization algorithms for web images. *Multimedia Tools and Applications* 76, 4 (Feb. 2017), 4801–4834. https://doi.org/10.1007/s11042-016-3795-2

[29] Hui Zeng, Yifeng Zhan, Xiangui Kang, and Xiaodan Lin. 2017. Image splicing localization using PCA-based noise level estimation. *Multimedia Tools and Applications* 76, 4 (Feb. 2017), 4783–4799. https://doi.org/10.1007/s11042-016-3712-8

Received 25th November 2024; revised 13th January 2025; accepted 14th December 2024

Automation for digital forensics: Towards a classification model for the community

Gaëtan Michelet
School of Criminal Justice
University of Lausanne
Lausanne, Switzerland
gaetan.michelet@unil.ch

Frank Breitinger
Institute of Computer Science
University of Augsburg
Augsburg, Germany
frank.breitinger@uni-a.de

Abstract

The current state of automation in digital forensics remains insufficiently defined. While the complexity of automated tools and methods has evolved significantly (e.g., from basic parsers to the integration of advanced techniques), it remains challenging to pinpoint the field's overall progress or compare methods. A first step towards a solution was the work 'Automation for digital forensics: Towards a definition for the community' which defines automation but cannot categorize various methods. This work aims to address this gap and presents a first classification model for automation for digital forensics. Therefore, we analyzed automation classification schemes from different disciplines (e.g., cars) and assessed various model possibilities as well as characteristics. We conclude that a 2-dimensional model with the axis 'decision' and 'level of automation' is most appropriate and provide an overview table with examples.

Keywords

Automation, Model, Digital Forensics investigation

ACM Reference Format:
Gaëtan Michelet and Frank Breitinger. 2025. Automation for digital forensics: Towards a classification model for the community. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3712716.3712725

1 Introduction

Definitions and classification schemes are foundational to any discipline, providing a structured view of the field's current state, identifying areas requiring advancement, and facilitating the systematic selection and comparison of methods. In 2023, Michelet et al. [14] defined automation for digital forensics as "Software or hardware that completes a task more efficiently, reliably, or transparently by reducing or removing the need for human engagement." The brought definition implies that two very different automated methods (varying in complexity, capabilities, or impact on the investigation) both qualify as automation. For example, an NTFS file system parser as provided in many digital forensics tools (FTK, Autopsy...), and a machine-learning-based tool helping identify artifacts of interest [3].

DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712725

While both approaches arguably automate steps of the process, they differ in their capabilities and the risks they entail. The first one follows the NTFS specifications and extracts information about each file and folder present in the partition. No decision is made, reducing the associated risks but limiting the tool's capabilities, e.g., a slightly altered/corrupted partition may not be parsed. In comparison, the second tool makes decisions based on a learning process, which enhances its capabilities (e.g., detection of previously unknown artifacts) but may lead to errors (i.e., the risks of potential error increases). Given current literature, there is no easy way to compare these two approaches despite their different nature. Consequently, a classification model providing a quick and easy understanding of the capabilities and risks of each method is needed. Having such a model allows us to better understand the capabilities and limits of a given method/tool, but more importantly the associated risks.

Creating models to classify automation is not new and has been addressed in various other areas by various authors such as Sheridan and Verplank [24], Endsley [4], or Parasuraman et al. [18] (more details are provided in Section 2). It even has been discussed to a certain extent for traditional forensic science, where [25] created a classification model for the use of algorithms (software) by an expert during the decision-making process.

In this article, we propose a preliminary classification model for automation in digital forensics, building on prior research and existing frameworks. We evaluate various potential classification models tailored to the unique requirements of digital forensics, drawing insights from previous findings to inform our approach. Our contributions are the following:

- Presentation of different possible approaches and considerations for the creation of a model classifying automated methods/tools in digital forensics.
- Selection and proposition of a preliminary classification model.
- Showcase the class attribution for four different methods/tools.

Terminology: The literature employs various terms, including method, approach, technique, and process, to describe aspects of automation. However, a clear and consistent definition of these terms is lacking. In this work, we primarily use the term method while occasionally employing synonyms for variety. Future research is needed to establish clear distinctions and definitions for these terms.

The rest of the paper is organized as follows: Existing ways to classify automation in general as well as in particular disciplines are explored in Section 2. The classification model for automation in digital forensics as well as the elements considered during its

creation are presented in Section 3. Section 4 provides considerations towards the proposed model and a discussion of the next steps. Finally, Section 5 concludes the paper.

2 Background and related work

The first section details some fundamental models for the classification of automation. These models had a large impact and served as input for many other models some of which are discussed in Section 2.2. The third section briefly summarizes literature in the domain of autonomous cars where well-known institutions developed and adopted models. Automation classification models proposed in less familiar or indirectly related disciplines are highlighted in Section 2.4. Lastly, articles providing discussions and surveys addressing this topic are mentioned.

2.1 Levels of automation

The possibilities to classify automated approaches or systems are frequently referred to as "levels of automation" (LoA). One of the first to introduce this concept was Sheridan and Verplank [24], who proposed a 10-LoA-model. The idea was to emphasize that a system is rarely completely manual or completely automated and to provide a way to better understand the degree to which a system is automated. Sheridan [23] made some modifications to the 10 levels resulting in a slightly more general description listed in Table 1.

1	The computer offers no assistance, human must take all decisions and actions
2	The computer offers a complete set of decisions/action alternatives, or
3	narrows the selection down to a few, or
4	suggests one alternative, and
5	executes that suggestion if the human approves, or
6	allows the human a restricted veto time before automatic execution
7	executes automatically, then necessarily informs the human, and
8	informs the human only if asked, or
9	informs the human only if it, the computer, decides to.
10	The computer decides everything, acts autonomously, ignores the human

Table 1: Levels of Automation from low (1) to high (10) proposed by [23, p.358]

This model often serves as a basis for the development of newer models.

Endsley [4] proposed a similar concept for the development of advanced cockpits. The authors called it the "allocation of roles between the expert system and the pilot". It consists of four levels with a progression that is similar to Sheridan's: The first level provides recommendations, the second level applies these recommendations if the operator accepts, the third level provides and applies the recommendation unless the operator restrains it, and the fourth level represents an autonomous system.

Another important concept was introduced by Parasuraman et al. [18] who divided the process into four distinct stages (information acquisition, information analysis, decision selection, and action implementation). By proposing to attribute a LoA to each of these stages, they created a two-dimensional level, allowing to precise the automation of each stage. The LoA model used is the same among the four stages.

2.2 Adaptations and new models

In addition to the fundamental work summarized in the previous section, many other models were created with variations in the name of the model, the number and name of levels, and the description of these levels[1].

For example, Draper [2] presented a 'level of control' model, that could be assimilated to LoAs. In their five levels model the control is incrementally reduced for the human and increased for the machine (levels: manual control, manual control with intelligent assistance, shared control, traded control, and supervisory control).

Another approach was explored by Endsley and Kaber [5], Kaber [10], Kaber and Endsley [11, 12], who listed four tasks that could be attributed to the human or to the computer (monitoring/generating (options or strategies)/selecting/implementing). They then introduced novel LoAs, specifying for each level which task is attributed to the human or the computer.

Proud et al. [19] revisited Parasuraman's proposition with four different tasks (observe, orient, decide, act) in a human spaceflight vehicle. They suggest eight levels of automation tailored for each task, with a similar difference in magnitude between each level. A similar approach was followed by Save et al. [22] who presented a LoA model for Air traffic management. They build on Parasuraman's idea and propose four different sets of levels (varying between five and eight levels), one for each stage of automation.

Some other work, such as the one presented by Fereidunian et al. [7] or Endsley and Kiris [6], adapted existing models. In the former, the authors required a new level for their application and added a 'data acquisition' level between L1 and L2 of Sheridan's model. The latter presents a situation awareness study, with the use of the control operator level, a modified version of Endsley's first model completed with a fifth 'manual level'

Among the different propositions, some of them stood out by bringing new dimensions and possibilities. For example, Frohm et al. [8] analyzed LoAs in manufacturing and proposed their own model making a distinction between mechanization and computerization. Each element has its own 7-level model. Milgram et al. [15] created a 3-dimensional model to classify 'human-mediated control of remote manipulation systems'. This was done based on three elements: the machine autonomy, the structure of the remote environment, and the knowledge of that remote environment. They then represent systems based on boxes with each edge of the box representing the lowest/highest levels of a system for a given element. Riley [20] presented a model composed of the levels of intelligence and the levels of autonomy. The level of automation is obtained by combining them. The level of intelligence relates to the type of information it can process, and the possibility to adapt based on the environment, operator's preferences, predicted operator's intent, or predicted operator's actions. The level of autonomy relates to the independence it has. In the beginning, it only provides information to the operator, then it can act with more and more autonomy.

2.3 Autonomous cars

Automation, and particularly levels of automation, has been widely discussed in the domain of autonomous cars. Different institutions such as the National Highway Traffic Safety Administration

[1]This section was inspired by Vagia et al. [26] and has therefore similarities with this work, e.g., many articles referenced here were also described in more depth in their survey; summaries of the referenced articles are similar.

(NHTSA), or the Society of Automotive Engineers (SAE) proposed models with various levels of car automation/autonomy.

The different levels provided by the NHTSA [1] and the SAE [17] are similar but have different names. In level 0, the driving is only operated by the driver, and in level 5, it is only operated by the car (under all conditions). Level 1 provides assistance with the steering or the acceleration/braking while level 2 provides assistance for both of them. Note that the driver is still responsible for monitoring the road and the environment. In level 3, the car handles the driving, but the user is expected to remain ready to take over in case there is a problem. This is not the case in level 4, in which the user is not required to take over, but the system is working under a set of circumstances/conditions. Once again, the higher the automation levels, the higher the number of tasks handled by the automatic system.

2.4 Introduction of models in new disciplines

Other disciplines, which may be less obvious when thinking about automation, also released models or adapted existing ones. For example, Rühr et al. [21] developed a classification model for digital investment management systems. It is based on two axes: (1) different processes, and (2) a level made by combining the levels of decision (automation) and the level of delegation. Four processes were identified, and seven levels were created by combining decision and delegation. Hashimoto et al. [9] proposed automation levels derived from the SAE levels and adjusted them to fit the earthmoving domain. They then presented a model adapted for the excavation and loading process but specified that it could also be done for different machines.

With respect to digital forensics the closest we could find was work by Swofford and Champod [25] who proposed a model for automation in providing expert opinion in the traditional forensic discipline. Their model is based on 6 levels. L0 represents the usual expert opinion, where no automation is involved (called algorithm in the paper). L5 is the opposite, where the conclusion is purely done by the algorithm. L1 and L2 are based on human opinion, with the algorithm run after the expert opinion is formed (possible for L1, recommended for L2). In L3 and L4, the algorithm has the upper hand and is run before the human. While this is a recommendation to run it before forming the human opinion in L3, this is mandatory in L4. Note that the switchover takes place between L2 and L3 where the opinion goes from 'human generated supported by automation' to 'automatically generated supported by the human'.

2.5 Discussions and Surveys

Discussions also took place, with introductions of new concepts or visions. Wickens et al. [27] introduced the notion of 'degree of automation' that combines both the levels (Sheridan) and the stages (Parasuraman). Miller and Parasuraman [16] discussed the decompositions made by Parasuraman and mentioned that the level attributed to each stage was the average of the level of each sub-task accomplished during that particular stage. It would therefore also be possible to choose another division, with smaller sub-tasks.

Throughout the years, several surveys analyzing the current state of the 'levels of automation' have been made. Kaber et al. [13] investigated and discussed problems related to different models available at that time, Frohm et al. [8] explored the LoA concept, and Vagia et al. [26] provided a more systematic review of the levels themselves, showing which levels exist and in which model they are present.

While reviewing the literature it became obvious that most domains cannot use existing models but are required to create their models or at least adjust existing ones. Even the model specifically created for traditional forensic sciences cannot seamlessly be applied to digital forensics. A reason is that many traditional tasks are better defined, e.g., matching a bullet to a gun or footprints to a shoe. Depending on the number of overlapping features, probabilities can be calculated allowing decision-making. However, existing literature provides interesting ideas and concepts.

3 Towards a classification model

We therefore decided to create a new model, tailored for digital forensics. As mentioned, we use the term method to refer to any tool or piece of code that automates a task or part of a task. This section discusses two important elements that were considered during the development of the classification model: Finding relevant characteristics to describe the tools/methods and determining the best trade-off between the model granularity and usability. Subsequently, a preliminary version of the model is proposed, and its practicality is demonstrated through examples.

3.1 Characteristics of the automated methods

Section 2 summarized various characteristics that can be used to describe automation. Concerning automation for digital forensics, four of them were identified as noteworthy: (1) the complexity, (2) the decision-making level, (3) the level of automation, and (4) the impact on the investigation. Note, that the presented characteristics are not completely independent.

3.1.1 The complexity. The complexity could help describe automated methods. Usually, the more complex an automated method is, the higher the capabilities, but the higher the associated risks (the use of complex AI-based methods unlocks new capabilities along with new associated risks). However, while it is somewhat possible to determine which method is more complex when comparing the two of them, it is difficult to establish a complexity scale. Concrete units for such a scale could be the number of operations achieved by the method, the number of code lines, or the computation time. As providing clear and objective categories of complexity is difficult, we decided to discard this characteristic.

3.1.2 The decision-making level. Making decisions is an important aspect in particular for a domain such as digital forensics. It indicates what kind of decision the method/tool can take. The higher the decision level, the higher the capability and the higher the risks. Note that usually, a human will review important decisions to minimize that risk. This characteristic can be divided into different categories more easily than complexity.

3.1.3 The level of automation. This characteristic will provide insight into the level of autonomy. Throughout the levels, the workload shifts from the investigator to the automated method/tool.

3.1.4 The impact. The impact on the investigation is a key characteristic for understanding an automated method. To determine the impact, the most straightforward measurement is the amount of work undertaken by the tool. This can be done by measuring the number of phases automated or the number of operations or tasks achieved. The main problem with this approach is that every task/phase does not impact the investigation with the same magnitude, and counting them does not provide an accurate estimation of the actual impact on the case-solving. An alternative could be the use of a weighing system, with an 'impact value' attributed to each common task indicating how important that task is for the investigation. The impact would then be computed based on each achieved task and its associated 'impact value'.

3.2 Dimensionality of the model

While the relevance of the characteristics used to categorize the automated methods is straightforward, we think that it is important to describe the trade-off between the granularity and the usability (or readability) in more depth. The former is simply the number of different categories where an automated method could be classified. The latter is the extent to which a model is simple enough to be easily understood. Ideally, a model would be somewhat sufficiently granular, while remaining usable. Consequently, there is a trade-off between these two elements. When the model granularity is increased, the number of classes increases impacting the readability/understandability of the model and vice versa.

If the granularity is, for example, pushed to the extreme, every method will have its own category. The model would be particularly precise, but its understandability and usability would be low, directly reducing the added value of the model. In contrast, if the granularity is reduced as much as possible, only two categories remain: automated or not.

When exploring the different possibilities for the classification scheme, two of them came to mind: a linear representation based on a single characteristic (not particularly granular but easy to use) and a matrix representation combining several of them (granularity and understandability varying according to the dimension of the matrix). Note that we consider the linear representation as a 1-dimensional matrix.

3.2.1 1-dimensional model (linear). For 1-dimensional models, the granularity is identical to the chosen characteristics as showcased in Fig. 1. While this model is straightforward to understand, depending on only one characteristic has limitations. If one uses, for example, the impact of the tool on the investigation, you might get some insight into its capabilities, but the understanding of the risks related to its use remains unclear. Consequently, we do not consider it precise enough to provide the required amount of information and help grasp the capabilities and risks of a given automated method.

3.2.2 2-dimensional model. The 2-dimensional model can be visualized as a table/mathematical plane with two characteristics/axes describing automation. Here, the number of possible categories will be the number of categories for the first characteristic (X-axis) multiplied by the number of categories for the second characteristic (Y-axis). An example of such a model is provided in Fig. 2. This provides more granularity than the previous model while keeping a reasonable readability and usability (depending on the number of categories).

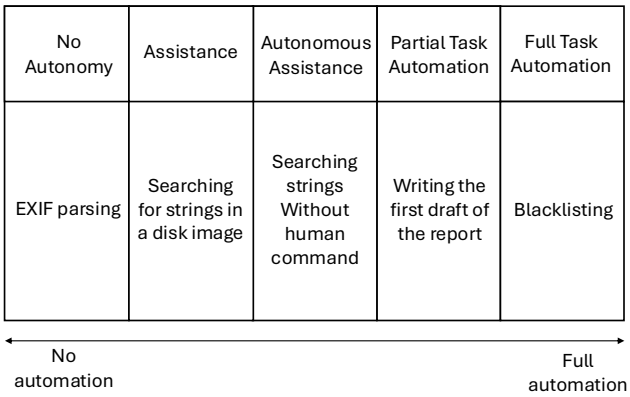


Figure 1: Example of 1-dimension model using the 'level of automation' characteristic

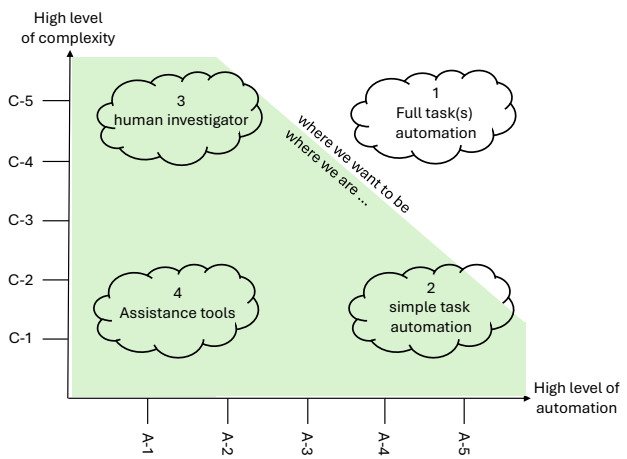


Figure 2: Example of 2-dimensions model using the 'complexity' and 'level of automation' as characteristics

3.2.3 3-dimensional model. The last explored solution is three dimensions which can be visualized or represented using a mathematical space. The granularity of this model is high, combining the categories of three different characteristics. While this is interesting, it would result in a complex model. For example, if three characteristics are used, each with five categories, the model results in 125 (5*5*5) categories. Moreover, space representations are suboptimal, as they reduce the model's readability and usability.

Note that with this list of downsides, we decided to stop the exploration of higher dimensionality models as the problems will amplify.

3.3 Proposed model

Based on this brief analysis, we decided to continue with the two dimensions option. It provides a balance between granularity with a reasonable amount of categories, a way to visualize/represent the model, and a simple use. The proposed model is displayed in Fig. 3. As for the characteristics, we decided on the level of automation and the decision-making level which are detailed in the upcoming sections.

3.3.1 Categories for the decision making level. The decision-making level consists of five levels, going from the 'no decision' level, for which no decision is taken, to the 'full decision' level, in which decisions made based on learning can be taken without any human control.

- **No decision**: No decision is made by the method, a simple operation or flow of operations is conducted. For example, a method reads every byte of a hard drive and copies it no matter the byte value.
- **Objective decision**: Decisions can be made by the method, but these decisions must be taken based on the result of a simple comparison. For example, a method highlighting files opened during a day of interest, and checking if the last access value is the same day as the day of interest.
- **Subjective decision**: Decisions can be made by the method. These decisions can be made based on a set of rules predefined by an investigator (no learning is involved). Note that input used for these rules can either be derived from data or provided as case context by the investigator. For example, a method retrieving/highlighting files that are usually of interest to the type of crime investigated.
- **Learning decision**: Decisions can be made by the method. These decisions are based on a (machine-)learning process and the set of rules is not known to the investigator. For example, email scam detection is based on the sending email address and the body of the email.
- **Full decision**: Decisions based on a (machine-)learning process can be made by the methods, and no feedback/control is required before submission or resuming the whole process.

The decision level provided to a method is the highest level it can achieve, meaning that a method mixing objective decisions, and learning-based decisions would classify as a method taking learning-based decisions.

3.3.2 Categories for the level of automation. The LoA comprises five levels, going from 'manual', in which everything is achieved manually, to 'full task automation', in which a complete forensic task is achieved automatically. These levels were adapted to better fit the digital forensics domain and provide the expected amount of information when combined with the decision-making level. Note that we make a difference between the accomplishment of (1) operations, (2) part of a forensic task, and (3) a complete forensic task.

An operation (or set of operations) refers to activities that are fundamental and (likely) have a minor impact on the investigation. Examples are the transformation and presentation of data, the parsing of data structures, or the filtering of files[2]. In contrast, a forensic task is considered more severe and is traditionally accomplished by the practitioner. Examples are identifying relevant messages, suggesting hypothetical relations between different users, understanding how a malicious piece of code works, or writing a report summarizing the results.

While the difference between an operation and a forensic task is clear, the limit between a set of 'advanced operations' and 'part of a forensic task' might sometimes be thin. It is important to keep in mind that the difference lies in the magnitude of workload accomplished, and we suggest considering it as 'partial task automation' if there is a doubt.

- **Manual**: The operation is achieved completely manually by the investigator. For example, the transformation of a datetime in hexadecimal value into a human-readable datetime.
- **Assistance**: The operation can be performed automatically, but only after a request from the investigator. For example, a carving module is launched after a button or command is entered by the investigator.
- **Autonomous assistance**: The method is automatically called to perform the operation if a process is finished or if a condition is met (not requested by the investigator). For example, a tab presenting the 50 last messages sent from a smartphone is opened once the file-system parsing module is finished.
- **Partial task automation**: The method is manually or automatically called, but part of a forensic task is performed, and not a single/few operations. For example, a tool allowing to write a section of the forensic report.
- **Full task automation**: The method is manually or automatically called, but a complete forensic task is performed. For example, a blacklisting of files is launched once a forensic image is detected.

Note that the automated workload increases as the level of automation increases. In the beginning, one or a few operations are achieved, while in the end, a complete forensic task is achieved.

3.3.3 Capabilities and risk assessments. Using the two chosen characteristics, it is possible to assess the risks and capabilities associated with an automated task. Generally, the higher the level of automation and decision-making level, the higher the risks and capabilities. However, it is important to keep in mind that the risk and capabilities increase is not linear. For example, with the decision level making, the rise of risk in the transition between level 0 (no decision) and 1 (objective decision) is smaller than the rise of risk in the transition between level 4 (learning decision) and 5 (full decision). In the first case, decisions are introduced into the process, but the risks remain minimal as basic comparison operators are applied. In the second scenario, the operator is completely ruled out of the decision-making (and checking) process. This also applies to the capabilities increase. Despite this non-linearity, we consider these choices useful to classify methods (tools) and quickly assess the associated capabilities/risks.

[2]We acknowledge that these operations may also have a significant impact, e.g., a faulty keyword searching software. However, on average, they have less impact.

3.4 Applying the model

This section discusses methods and provides their classification as an example, see Fig. 3. The aim is to showcase how the classification can be applied while providing information about the capabilities and risks of a method. The chain of thought is explained in four of the provided examples.

Manually selecting files created at a certain date: The file selection is purely manual, which directly determines the level of automation of this method to L0. Decision-wise, filtering based on a date of interest is achieved through the use of single comparisons: is the creation date of this file the same as the date of interest? This classifies the decision level of this method to D1.

Searching using a list of keywords generated based on the investigated crime (provided to the tool): The amount of workload accomplished is closer to a single operation than a forensic task, or part of a forensic task. Moreover, this method is used as an assistant and launched by the investigator. It therefore qualifies as L1. The decisions made by the tool are based on rules and context provided by the investigator. The maximum level of decision is the subjective one D2.

Clustering textual documents based on their topic using machine/deep learning: Dividing documents based on their topics is a time-consuming step during an investigation. While it is helpful, it does not provide an evaluation of the document's relevance (the actual objective), which will be manually conducted by the investigators. Therefore, this method achieves an important part of the 'identification of relevant documents' forensic task and can be classified as L3. The decision process is based on a clustering algorithm and relates to machine learning. This method's maximum level of decision is D3.

Classifying an executable file maliciousness based on a classification model: Determining the maliciousness of an executable can be considered a forensic task, in particular during incident response or particular investigations. The level of automation of this method is L4. As for the previous method, the process is based on a classification model obtained through a machine-learning procedure. The maximum level of decision-making for this method is therefore D3.

Overall, while the difference between L2 and L3 may not be entirely intuitive, it is expected that non-expert users would become comfortable with the model in a short amount of time. A careful reading of Section 3.3.1 might even be sufficient.

4 Discussion and next steps

It is important to mention that the proposed model is a working model that we consider interesting and usable, but not fully matured. Its purpose is not to become a standard but to serve as a basis for discussions within the community. Only a few researchers were involved in the creation process of this classification schema. More feedback, discussions, and involvement from other members of the community are required before getting a more sound and globally accepted model. As an example, some might think that a third dimension would be required and useful, and in that case, it could be interesting to explore different ways to provide additional information (dimensions) while keeping representability and usability, e.g., by representing that third dimension using colors or text fonts. Determining if that kind of adjustment is needed or wanted by practitioners/researchers would require additional opinions.

Therefore, the next step is to gather several members of the community interested in advancing automation for our discipline to collectively discuss how the preliminary classification model should be improved and enhanced. Another interesting aspect that would help make progress is the division of the overall digital forensic process, ideally reaching a list of atomic tasks. This would allow us to analyze which one can be automated, and improve the current state of the automation in digital forensics.

5 Conclusion

Definitions and classification schemes are essential for enhancing the current state. Consequently, automation in digital forensics requires a classification system to better understand its present status and assist practitioners in determining the capabilities and risks of a method or tool when selecting them. A 2-dimensional matrix using the characteristics of the level of automation and decision-making enables the organization and comparison of various methods. It is important to note that as the level of automation or decision-making increases, so does the risk of error. If a task can be solved with the same quality of outcome, preference should be given to the tool with lower levels. While the proposed classification model may not be flawless, it offers a strong foundation and encourages further discussion within the community.

Author contribution statement

Gaëtan Michelet: Conceptualization, Methodology, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization.
Frank Breitinger: Conceptualization, Methodology, Investigation, Writing - Review & Editing, Visualization, Supervision.

Use of AI writing assistance

The authors of this paper have used ChatGPT-4 and the Grammarly plugin to correct typographical and grammatical errors, improve the wording of certain sentences, and get inspiration for some of the examples provided in Fig 3. All recommendations were thoroughly evaluated and modified as necessary before being incorporated into this document.

Acknowledgments

We would like to thank Graeme Horsman for the discussions that concretized ideas and contributed to the development of this paper. We also thank the DFDS reviewers, whose comments greatly helped improve this paper.

References

[1] National Highway Traffic Safety Administration. 2022. https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-05/Level-of-Automation-052522-tag.pdf
[2] John V Draper. 1995. Teleoperators for advanced manufacturing: Applications and human factors challenges. *International Journal of Human Factors in Manufacturing* 5, 1 (1995), 53–85.
[3] Xiaoyu Du and Mark Scanlon. 2019. Methodology for the automated metadata-based classification of incriminating digital forensic artefacts. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 1–8.
[4] Mica R. Endsley. 1987. The Application of Human Factors to the Development of Expert Systems for Advanced Cockpits. *Proceedings of the Human Factors Society Annual Meeting* 31, 12 (1987), 1388–1392. doi:10.1177/154193128703101219 arXiv:https://doi.org/10.1177/154193128703101219

[5] Mica R. Endsley and David B. Kaber. 1999. Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics* 42, 3 (1999), 462–492. doi:10.1080/001401399185595 arXiv:https://doi.org/10.1080/001401399185595 PMID: 10048306.

[6] Mica R. Endsley and Esin O. Kiris. 1995. The Out-of-the-Loop Performance Problem and Level of Control in Automation. *Human Factors* 37, 2 (1995), 381–394. doi:10.1518/001872095779064555 arXiv:https://doi.org/10.1518/001872095779064555

[7] Afreza Fereidunian, Caro Lucas, Hamid Lesani, Matti Lehtonen, and Mikael Nordman. 2007. Challenges in implementation of human-automation interaction models. In *2007 Mediterranean Conference on Control & Automation*. 1–6. doi:10.1109/MED.2007.4433895

[8] Jörgen Frohm, Veronica Lindström, Johan Stahre, and Mats Winroth. 2008. Levels of automation in manufacturing. *Ergonomia-an International journal of ergonomics and human factors* 30, 3 (2008).

[9] Takeshi Hashimoto, Mitsuru Yamada, Genki Yamauchi, Yasushi Nitta, and Shinichi Yuta. 2020. Proposal for Automation System Diagram and Automation Levels for Earthmoving Machinery. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, Vol. 37. IAARC Publications, 347–352.

[10] David B. Kaber. 1996. *The effect of level of automation and adaptive automation on performance in dynamic control environments*. Ph. D. Dissertation. https://www.proquest.com/dissertations-theses/effect-level-automation-adaptive-on-performance/docview/304288893/se-2 Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Dernière mise à jour - 2023-02-21.

[11] David B. Kaber and Mica R. Endsley. 1997. The Combined Effect of Level of Automation and Adaptive Automation on Human Performance with Complex, Dynamic Control Systems. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 41, 1 (1997), 205–209. doi:10.1177/107118139704100147 arXiv:https://doi.org/10.1177/107118139704100147

[12] David B Kaber and Mica R Endsley. 1997. Out-of-the-loop performance problems and the use of intermediate levels of automation for improved control system functioning and safety. *Process safety progress* 16, 3 (1997), 126–131.

[13] David B Kaber, Emrah Onal, and Mica R Endsley. 2000. Design of automation for telerobots and the effect on performance, operator situation awareness, and subjective workload. *Human factors and ergonomics in manufacturing & service industries* 10, 4 (2000), 409–430.

[14] Gaëtan Michelet, Frank Breitinger, and Graeme Horsman. 2023. Automation for Digital Forensics: Towards a definition for the community. *Forensic Science International* (2023), 111769.

[15] P. Milgram, A. Rastogi, and J.J. Grodski. 1995. Telerobotic control using augmented reality. In *Proceedings 4th IEEE International Workshop on Robot and Human Communication*. 21–29. doi:10.1109/ROMAN.1995.531930

[16] Christopher A. Miller and Raja Parasuraman. 2003. Beyond Levels of Automation: An Architecture for More Flexible Human-Automation Collaboration. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 47, 1 (2003), 182–186. doi:10.1177/154193120304700138 arXiv:https://doi.org/10.1177/154193120304700138

[17] Society of Automotive Engineers. 2021. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. https://www.sae.org/standards/content/j3016_202104/

[18] Raja Parasuraman, Thomas B. Sheridan, and Christopher D. Wickens. 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 30, 3 (2000), 286–297. doi:10.1109/3468.844354

[19] Ryan W Proud, Jeremy J Hart, and Richard B Mrozinski. 2003. *Methods for determining the level of autonomy to design into a human spaceflight vehicle: a function specific approach*. Technical Report. NASA.

[20] Victor Riley. 1989. A General Model of Mixed-Initiative Human-Machine Systems. *Proceedings of the Human Factors Society Annual Meeting* 33, 2 (1989), 124–128. doi:10.1177/154193128903300227 arXiv:https://doi.org/10.1177/154193128903300227

[21] Alexander Rühr, David Streich, Benedikt Berger, and Thomas Hess. 2019. A classification of decision automation and delegation in digital investment management systems. In *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS)*. 1435–1444.

[22] Luca Save, Beatrice Feuerberg, and E Avia. 2012. Designing human-automation interaction: a new level of automation taxonomy. *Proc. Human Factors of Systems and Technology* 2012 (2012).

[23] Thomas B. Sheridan. 1992. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press. https://books.google.ch/books?id=eu41_M2Do9oC

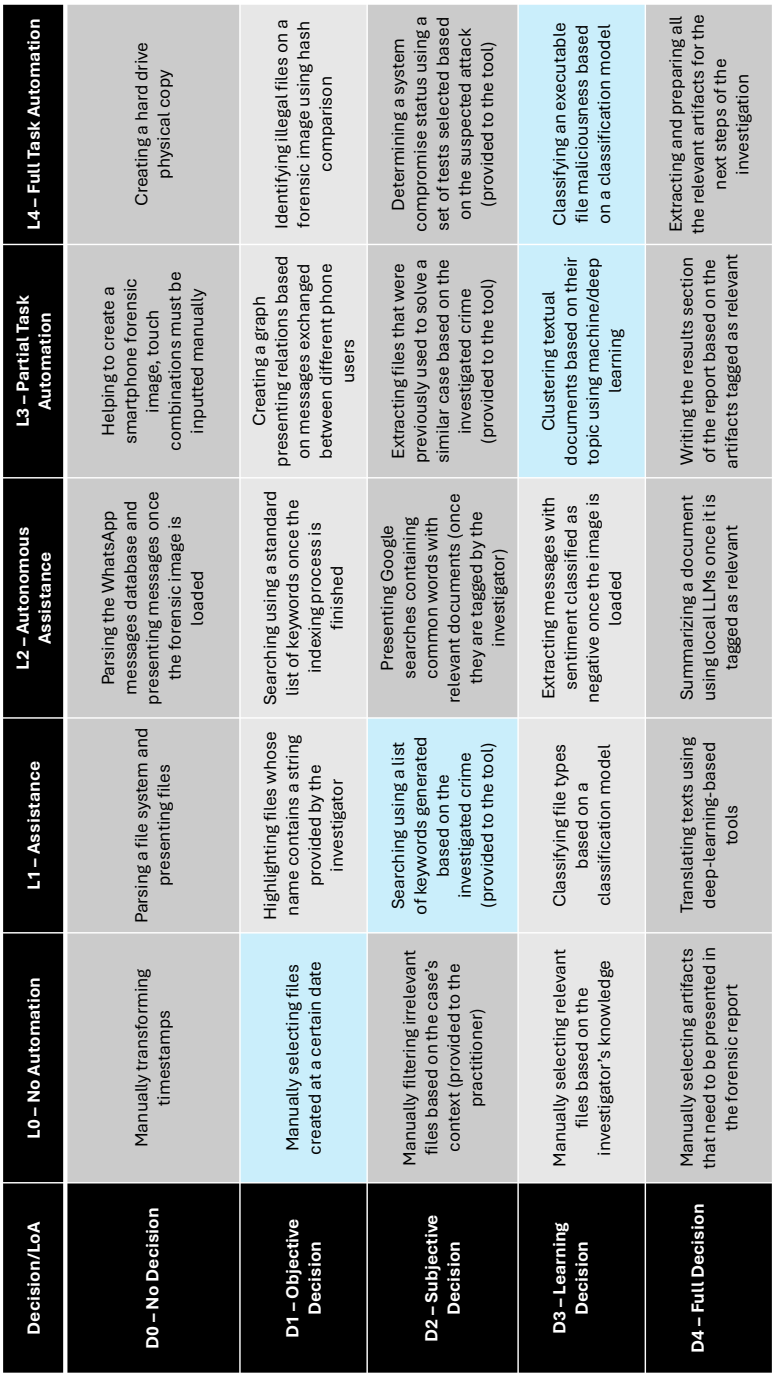
[24] Thomas B Sheridan and William L Verplank. 1978. *Human and computer control of undersea teleoperators*. Technical Report. Massachusetts Inst of Tech Cambridge Man-Machine Systems Lab. https://apps.dtic.mil/sti/citations/ADA057655

[25] Henry Swofford and Christophe Champod. 2021. Implementation of algorithms in pattern & impression evidence: A responsible and practical roadmap. *Forensic Science International: Synergy* 3 (2021), 100142. doi:10.1016/j.fsisyn.2021.100142

[26] Marialena Vagia, Aksel A. Transeth, and Sigurd A. Fjerdingen. 2016. A literature review on the levels of automation during the years. What are the different taxonomies that have been proposed? *Applied Ergonomics* 53 (2016), 190–202. doi:10.1016/j.apergo.2015.09.013

[27] Christopher D. Wickens, Huiyang Li, Amy Santamaria, Angelia Sebok, and Nadine B. Sarter. 2010. Stages and Levels of Automation: An Integrated Meta-analysis. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 54, 4 (2010), 389–393. doi:10.1177/154193121005400425 arXiv:https://doi.org/10.1177/154193121005400425

A Model



Decision/LoA	L0 – No Automation	L1 – Assistance	L2 – Autonomous Assistance	L3 – Partial Task Automation	L4 – Full Task Automation
D0 – No Decision	Manually transforming timestamps	Parsing a file system and presenting files	Parsing the WhatsApp messages database and presenting messages once the forensic image is loaded	Helping to create a smartphone forensic image, touch combinations must be inputted manually	Creating a hard drive physical copy
D1 – Objective Decision	Manually selecting files created at a certain date	Highlighting files whose name contains a string provided by the investigator	Searching using a standard list of keywords once the indexing process is finished	Creating a graph presenting relations based on messages exchanged between different phone users	Identifying illegal files on a forensic image using hash comparison
D2 – Subjective Decision	Manually filtering irrelevant files based on the case's context (provided to the practitioner)	Searching using a list of keywords generated based on the investigated crime (provided to the tool)	Presenting Google searches containing common words with relevant documents (once they are tagged by the investigator)	Extracting files that were previously used to solve a similar case based on the investigated crime (provided to the tool)	Determining a system compromise status using a set of tests selected based on the suspected attack (provided to the tool)
D3 – Learning Decision	Manually selecting relevant files based on the investigator's knowledge	Classifying file types based on a classification model	Extracting messages with sentiment classified as negative once the image is loaded	Clustering textual documents based on their topic using machine/deep learning	Classifying an executable file maliciousness based on a classification model
D4 – Full Decision	Manually selecting artifacts that need to be presented in the forensic report	Translating texts using deep-learning-based tools	Summarizing a document using local LLMs once it is tagged as relevant	Writing the results section of the report based on the artifacts tagged as relevant	Extracting and preparing all the relevant artifacts for the next steps of the investigation

Figure 3: Proposed model using the "level of automation" and "decision-making level" characteristics

FEAR: A Novel Framework for Representing Digital Forensic Artifacts in Knowledge Graphs

Allan Korol
School of Science
Edith Cowan University
Perth, WA, Australia
a.korol@ecu.edu.au

Leslie F. Sikos
School of Science
Edith Cowan University
Perth, WA, Australia
l.sikos@ecu.edu.au

Abstract

In digital forensics, knowledge graphs have demonstrated significant potential via software agent automation and knowledge discovery using encoded expert knowledge in, for example, the form of Semantic Web rules. These advancements have been limited in terms of efficiently encoding extracted digital artifacts into graph representation, the associated overhead of implementing frameworks to handle digital forensic evidence, and the lack of sharing of such code that is often a preamble to other research. This paper introduces a digital forensic framework, Forensic Extraction and Representation (FEAR), that enables a simplified process of accessing and encoding extracted digital forensic artifacts in semantic knowledge graphs. The adoption of such a framework can facilitate the sharing of expert knowledge and reduce the burden of development for researchers exploring the application of knowledge graphs, software agents, and automated reasoning in the field of digital forensics, while accelerating the adoption of emerging research by practitioners.

CCS Concepts

• **Applied computing** → **System forensics**; *Network forensics*; • **Computing methodologies** → **Semantic networks**; *Reasoning about belief and knowledge*; *Ontology engineering*; • **Software and its engineering** → *Very high level languages.*

Keywords

digital forensic artifacts, semantic knowledge graph, digital forensic software agent, digital forensic framework, declarative language

ACM Reference Format:
Allan Korol and Leslie F. Sikos. 2025. FEAR: A Novel Framework for Representing Digital Forensic Artifacts in Knowledge Graphs. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3712716.3712726

1 Introduction to Knowledge Graphs in Digital Forensics

Digital forensics is a diverse and rapidly evolving domain where knowledge graphs have shown significant potential for automating processes and enabling knowledge discovery [15]. Despite their promise, advancements have been restricted by the time and complexity required to develop and implement frameworks to handle digital forensic evidence and map artifacts extracted from disk images, memory dumps, and packet captures to semantic knowledge graphs. In turn, this limits implementing emerging research in the field of semantic knowledge graphs due to no single comprehensive framework aligned with de facto industry standard tools. Further, the limited ability to share the logic of artifact-to-knowledge-graph mappings restricts researchers and practitioners from examining and encoding relationships into rules, as well as exploring automated software agents to perform knowledge discovery based on that encoded knowledge. To address the issue of software agents not having access to data that can be readily interpreted, knowledge needs to be encoded in formats that are machine-interpretable [16]. In this case, the representation of digital forensic artifacts using Web Ontology Language (OWL)[1]—a language that allows the machine-interpretable representation in knowledge graphs—as *RDF*[2] statements achieves this, and enables automated reasoning over digital forensic knowledge.

Semantic knowledge graphs are a set of statements (triples) about subjects and their properties and relationships—formally referred to as predicates—to an object.[3] The most widely used framework for representing subject-predicate-object knowledge statements in this way is RDF. Listing 1 is an example of how metadata of a file artifact extracted from a disk image could be represented in RDF utilizing terms from the Unified Cybersecurity Ontology[4] (UCO), using Turtle[5] serialization; a widely used standard for representing triples in text form. These triples can then be stored in a triple store—a database or database-like structure such as Jena,[6] Stardog,[7] and Neo4j[8]—that can be accessed and queried by methods similar to relational databases.

Listing 1: Example RDF/Turtle representation of a file using UCO terms.

```
1  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
2  @prefix uco: <https://ontology.unifiedcyberontology.
       org/uco/observable/> .
3
4  #Defines a subject, consisting of predicate-objects
```


DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712726

[1] https://www.w3.org/OWL/
[2] https://www.w3.org/RDF/
[3] Such triples can be complemented by a graph identifier to form quadruples (quads), thereby capturing statement-level context or provenance.
[4] https://ontology.unifiedcyberontology.org/documentation/
[5] https://www.w3.org/TR/turtle/
[6] https://jena.apache.org/
[7] https://www.stardog.com/
[8] https://neo4j.com/

```
5  :file-8ebd0cc0-2f4fcbeccbf3
6    #Defines the concept this subject is an instance of
7    a uco:File ;
8
9    #Entries of dates and times related to the subject
10   uco:accessedTime
         "2020-09-19T03:43:12.368Z"^^xsd:dateTime ;
11   uco:creationTime
         "2020-09-19T03:40:00.691Z"^^xsd:dateTime ;
12   uco:modifiedTime
         "2020-09-19T03:40:00.863Z"^^xsd:dateTime ;
13
14   #Entry for the size and the termination marker (.)
15   uco:sizeInBytes "7168"^^xsd:integer .
```

Listing 1 defines namespace prefixes which are used for shorthand notation for the full *Uniform Resource Identifier* (URI) of a subject, predicate, or object. A subject is defined (: file –8ebd0cc0–2 f4fcbeccbf3), Line 5, as an instance of the File concept, from the obsv namespace-prefix defined on Line 2. The predicate-object-pairs represent the time this subject was accessed (uco:accessedTime), created (uco:creationTime), and last modified (uco:modifiedTime)—all using a standard XML Schema Definition's date/time format (^^xsd:dateTime). The uco: sizeInBytes represents the size of the file, in bytes, with value being of type integer (^^xsd: integer).

A further limitation of research in the field is the lack of a core reference ontology that captures the relationship between concepts of ontologies from related knowledge domains. A recent scoping review detailed ontology use in the digital forensic domain, including two *Unified Security Ontologies* and seven specific scenario ontologies [13], with these ontologies widely recognized in the domain [11]. A recent work aims to address this via the NORIA Ontology, with the authors recognizing that there are areas in which the ontology can still be extended further—namely with respect to Cyber Threat Intelligence (CTI) [19].

2 Review of the State of the Art for Semantic Knowledge Graphs in Digital Forensics

In recent years, *knowledge graphs* have grown in popularity, primarily due to the emergence and proliferation of Google's Knowledge Graph [18]. However, the concept of knowledge graphs well predates *Google's* initiative, being traced back to the 1970s [20]. In the past two decades, approaches have been published for representing digital forensic artifacts in semantic knowledge graphs through numerous ontologies [17]. Benefits of such representations include: the ability to aggregate heterogeneous evidence stored in different formats, to discover relationships, and support automated reasoning to derive deductions from implicit knowledge [8]. To achieve reasoning and knowledge inference the Semantic Web Rule Language (SWRL)[9] has previously been proposed [1, 8]. Building on these benefits, several researchers have examined how sources of evidence can be used in the process of constructing knowledge graphs from digital forensic artifacts [3, 6, 7, 10, 14, 21].

Based on research examining methods of applying data mining techniques to digital forensic investigations, a framework has been proposed that leverages an ontology with rule and pattern matching [10]. The desired outcome of this framework is to assist analysts by automatically extracting information for initial analysis.

While developing the *Ontology for the Representation of Digital Incidents and Investigations* (ORD2I), a detailed description of a framework has been provided, along with methods used in the automated extraction using the *Log2Timeline* tool—a component of *Plaso*,[10] for timeline analysis [6]. The research builds on previous work [4, 5] involving the *Semantic Analysis of Digital Forensic Cases* (SADFC) approach and shows the process of populating the ORD2I ontology into a triple store, and using data extracted from crime scenes. The research also proposed a framework that allows the automated extraction of digital forensic data from disk images, for the purpose of populating a triple store using an ontology. It is further shown that *SPARQL*[11] queries can be used to retrieve information about events that interact with malware [6].

Leeds Beckett University and the West Yorkshire Police Department present research aimed at developing "a standardized data storage format for digital forensic evidence data that can be queried to allow for links to be created between cases and exhibits, both historic and current" [3]. The research outlines methods to collect heterogeneous information from existing forensic tools and combine this into a data store that supports queries via SPARQL. The research makes use of parsers to convert the outputs of established tools into more accessible formats and while an *Autopsy*[12] module was considered, the researchers decided against this method due to the time constraints and "simplicity in which to create new parsers for HTML reports" [3, p. 16].

ForensicFlow is a proposed system with a pluggable architecture that supports extension to additional data sources as the need arises [7]. The system that serializes artifacts based on an ontology the researchers developed to provide uniform representation of data sources for storage. The ontology represents events and artifacts in such a way "to allow easy adoption of any new data sources or artifact types that may appear in the future" [7, p. 71]. The proposed system leverages the open-source Autopsy tool, Plaso Log2Timeline, and Volatility[13] for extracting semi-structured digital forensic artifacts from the original data capture, which constitutes unstructured data.

A recent work proposes the use of knowledge graph question answering (KGQA) applied to IoT forensics and the rapidly expanding field of large language models (LLM), aiming to enable a more simplified interface with artifacts "using natural language questions facilitated by a deep-learning-powered KGQA model" [22, p. 1] for investigators.

Examining the recent advancements, the use of knowledge graphs has been detailed in the field of cyber security with various reasoning and deduction techniques, including rule-based approaches and machine learning-based methods [12]. The research also examines methods of construction and entity extraction for use in the, general, cybersecurity field. While this does focus more on data from unstructured text, it does highlight the various use cases, and the lack of methods related to digital forensics.

Within these and other research conducted in the field, a common approach has been to use log files extracted from disk images of

[9] https://www.w3.org/submissions/SWRL/

[10] https://github.com/log2timeline/plaso

[11] https://www.w3.org/TR/sparql11-query/

[12] https://www.autopsy.com/

[13] https://volatilityfoundation.org/

a storage medium, or the metadata of the file system through existing tools or suites, such as the *Forensic Toolkit (FTK)*,[14] Autopsy, *EnCase*,[15] or the *Plaso* tool and *Log2Timeline*. The use of these tool outputs highlight the complexity of building methods of extraction, but does underscore the ability for the progressive enrichment of data via incremental steps. While tools such as FTK, Autopsy, and EnCase provide graphical interfaces for investigators, the popular use of Log2Timeline indicates that utilizing intermediate formats is invaluable for creating semantically enriched knowledge graphs.

3 Knowledge Graph Adoption Challenges in Digital Forensics

Digital forensics faces significant challenges due to disparate data sources—including disk images, memory dumps, cloud artifacts, and network captures—each providing partial information in varied formats. For example, the *Forensic Artifacts*[16] project catalogs over 250 artifact sources for Windows systems, illustrating the complexity and diversity of data in forensic analysis. A digital forensic investigation may involve several data sources derived from a single set of investigation files, each progressively adding small amounts of detail. The extensive range of data types and formats that investigators must handle is exemplified by the comprehensive, though not exhaustive, list of artifact sources in the Forensic Artifacts project. Additionally, it is important to note that with advancements in technology, such as novel hardware tools and cloud services, there is a need to use new and emerging metadata to capture the associated semantics, with this body of artifacts ever-growing.

In conducting pre-research activities, the current landscape relies heavily on bespoke frameworks tailored to individual projects. Researchers often develop unique methods for constructing knowledge graphs specifically suited to their immediate objectives—these methods are often opaque and closed-source with little detail in publications about the process. While these frameworks encode extracted artifacts into RDF, there is no common method for expressing the mapping or codification of artifacts. Such approaches can be effective in achieving project-specific goals, however it becomes problematic when attempting to integrate findings with other researchers or to generalize the results for broader, industry-wide applications. This leads to the following challenges:

- Current methods rely on bespoke, case-specific applications for mapping digital forensic artifacts extracted from disk images, memory dumps, and network packet captures into RDF. These, often closed-source or proprietary, implementations limit broader applicability and the development of, and use as a, generalized solutions.
- No known resources or frameworks, in the space of knowledge graph construction, integrate tightly with standard industry tools, such as Autopsy, to streamline knowledge graph construction and exploration of extracted forensic artifacts.
- There are no known projects sharing comprehensive methods for mapping extracted digital forensic artifacts to knowledge graphs, leaving researchers without standardized methods or examples, making it challenging to achieve consistency and interoperability in their efforts.

With an absence of methods for providing standardized access to digital forensic artifacts for knowledge graph construction along with no standardized representation for sharing the mapping of digital forensic artifacts to knowledge graphs, we examine how a holistic digital forensics framework for interfacing with de facto industry standard tools, and efficiently expressing the logic of mapping extracted digital forensic artifacts into knowledge graphs can reduce time-consuming tasks performed by many researchers relying on these for their research.

4 A Framework for Mapping Digital Forensic Artifacts to Semantic Knowledge Graphs

The FEAR Framework will act as a research-to-practice conduit and is intended to be used across all aspects of semantic knowledge graph construction processes for digital forensics. By bridging the gap between academic research and the practical demands of real-world forensic investigations, the FEAR Framework will ensure that academic advancements can be seamlessly translated into real-world investigations. Within this field, there are three distinct stakeholders of the FEAR Framework:

- Ontology Engineers: Develop ontologies for representing digital forensic concepts, and the relationships between them. These engineers may develop scripts for the *Graph-Codify Forensic Extraction and Representation (GFEAR)* language to map artifacts to knowledge graphs.
- Researchers: Use the ontologies developed by the Ontology Engineers and apply software agents, automated reasoning, and knowledge discovery techniques—such as through the use of SWRL rules. These researchers may also develop GFEAR scripts mapping extracted artifacts to knowledge graphs. Further, researchers may also develop knowledge graph queries and tools for use by practitioners.
- Practitioners: Digital forensic investigators who use knowledge graphs generated for cases to query relationships between artifacts and generate reports for use in court or other legal settings. Practitioners may use knowledge graph queries and tools developed by researchers to identify interesting relationships between artifacts or extract knowledge discovered through software agents or automated reasoning.

The FEAR Framework is designed to be light-touch for researchers and practitioners needing knowledge graph construction—requiring only basic configuration of the mapping library being used, and the endpoint to the triple store. With digital forensic artifacts being transformed into RDF, and stored in an accessible location, the development of queries and the deployment of tools such as software agents and automated reasoners—that rely on semantic knowledge graphs—can be more easily integrated into the digital forensic investigation process for use by practitioners.

Figure 1 proposes a process of integrating the FEAR Framework with digital forensic investigations, encompassing the following:

[14] https://www.exterro.com/forensic-toolkit
[15] https://www.opentext.com/products/encase-forensic
[16] https://github.com/ForensicArtifacts/artifacts

(1) Within digital forensic frameworks, a phase of collection or acquisition is common [2]. This includes investigators collecting evidence from a suspect, victim, or compromised systems for analysis.
(2) Forensic tools, such as Autopsy, are used to extract artifacts from the evidence collected.
(3) The FEAR Framework Autopsy module imports the extracted artifacts from the Autopsy Blackboard.[17] A command-line tool allows outputs from other tools in common serialization formats—such as JSON, CSV, and XML.
(4) The framework identifies and executes the applicable mapping script(s) for each artifact submitted to the endpoint from Autopsy—and other forensic tools.
(5) The intermediate representation of the artifact, coupled with information in the mapping scripts, is used by the framework to search the triple store for existing entities—ensuring no duplicates.
(6) Once existing entities in the triple store are loaded, the intermediate representation is merged into the triple store, adding any missing knowledge statements.
(7) Researchers can develop software agents that perform automated reasoning, and knowledge discovery—committing this information back into the triple store for extraction through queries by practitioners.
(8) Practitioners can use tools and queries developed by researchers against the semantic knowledge graph, to generate visualizations and tabulated outputs of artifact relationships—this may include knowledge that has been inferred or discovered through software agents, automated reasoning, or SWRL rules.
(9) The visualizations and outputs of automated reasoning can be incorporated into reports to better convey evidence and the relationship between artifacts.

4.1 Integration with De Facto Standard Forensic Tools

One of the key challenges in the field is ensuring new research innovations can be adopted by practitioners working on real-world cases. By providing an integration with industry standard tools, it is possible to provide consistent, and simple access to digital forensic artifacts researchers need to drive further developments through the same tools practitioners use.

The FEAR Framework will be released with an Autopsy module to automate the process of accessing the extracted digital forensic artifacts from the Autopsy Blackboard. While this assists researchers to quickly gain access to the artifacts for research purposes, being built into the framework that is used by practitioners means that there is a consistent way of accessing the data, whether it is being used as part of research activities, or in real-world investigations. This eliminates the need for manual, tool-specific coding by researchers or practitioners.

To communicate the artifacts out of Autopsy, the module uses web-based requests. While developed to operate with a specific web service for the FEAR Framework, this method for exporting individual artifacts used by the module allows this data to be sent to any endpoint desired. Therefore, the module operates in Autopsy as a webhook producer [9], and could be directed to any consumer to access Autopsy artifacts, further standardizing access for research use.

4.2 Digital Forensic Artifact Mapping Logic Language

Knowledge graph construction methods developed through the use of programming languages and software libraries require substantial programming experience. Software libraries do reduce the effort needed to construct semantic knowledge graphs and interface with triple stores. However, knowledge related to mapping extracted artifacts to knowledge graphs often remains locked within the code itself, obscured by the functions and personal coding styles of individual developers.

The GFEAR language was developed and integrated into the FEAR Framework to promote its adoption and support the open sharing of knowledge related to mapping digital forensic artifacts through a standardized way to create and share mapping scripts. The primary purpose is to define how data fields from each artifact are transformed into entities. Additionally, it expresses how to find existing entities in a graph using those fields to ensure duplicates are not created. While not exhaustive, Listing 2 demonstrates the most common language constructs expected to be used by GFEAR scripts.

As a declarative language, GFEAR mapping scripts are agnostic to any system or programming language, with the goal of being used in any system that implements a GFEAR compiler or interpreter. The language, and its integration into the FEAR Framework, present the following benefits to researchers and practitioners:

- A declarative language significantly reduces programming proficiency required and is designed to simplify the representation of basic mapping logic from an extracted artifact to knowledge graph representation.
- The mapping logic in scripts can be shared between researchers and practitioners to quickly generate semantic knowledge graphs of digital forensic artifacts.
- In-built notation for identifying entities abstracts away complex code to perform entity searching and merging into existing knowledge graphs.

These benefits will accelerate the translation from research to practice in the field of digital forensic semantic knowledge graphs by streamlining their construction from well-known artifacts provided by industry standard tools.

4.3 Case Study: Knowledge Graphs with Encoded Expert Knowledge

Listing 2 is an example of a GFEAR script using an ontology that defines Autopsy terms.[18] The result of executing Listing 2 against an Autopsy case is a knowledge graph composed of entities for each metadata artifact on the Autopsy Blackboard, related to the host they were derived from, and the MD5 hash of the file contents—as calculated by Autopsy.

[17] https://www.sleuthkit.org/sleuthkit/docs/framework-docs/basics_page.html

[18] https://purl.org/ontology/autopsy/

Listing 2: Example of a GFEAR script to map a file artifact in the Autopsy Ontology.

```
1  // Defines a codifier with a type and unique path name
2  define codifier("TSK_METADATA", "Autopsy/Metadata");
3
4  // Provides prefixes for ontologies used in the script
5  ontology("https://purl.org/ontology/autopsy/", "aut");
6
7  // Core reference ontology prefixes used in the script
8  include rdf, rdfs, xsd, owl, xml;
9
10 // Defines the expected fields of an artifact accepted
11 // by the script which can used as values
12 accepts(HostSha1, SourceFile, SourceFileMd5,
       CreatedDateTime);
13
14 // Defines the entity and type created by the script
15 result Metadata as aut:Metadata with {
16   // Create 3 properties with specific predicates,
17   // types and field-values from the artifact.
18   create property aut:sourceFile as xsd:string with
         SourceFile;
19   create property aut:sourceFileMd5 as xsd:hexBinary
         with SourceFileMd5;
20   create property aut:createdDateTime as xsd:dateTime
         with CreatedDateTime;
21
22   // Create an entity, to represent the host the
23   // file was found on, with a specific predicate,
24   // type and a value that can be used to uniquely
25   // identify the entity in the triple store.
26   create entity sourceHost as aut:Host with {
27     create property aut:hostSha1 as xsd:hexBinary with
           HostSha1;
28   } identified by required aut:hostSha1;
29
30   // Creates a predicate-object relationship to
31   // the created, sourceHost, entity.
32   // Attaches provenance information about the file.
33   create property aut:sourceHost with entity
         sourceHost;
34
35   // To prevent duplicates, defines the aut:sourceFile
36   // and aut:sourceHost as predicate-objects that
37   // uniquely identify the entity in a triple store.
38 } identified by required aut:sourceFile, required
       entity aut:sourceHost sourceHost;
```

To retain file provenance information, Listing 2 creates an entity for the host the metadata was extracted (Lines 26–28). This entity is uniquely identified by the SHA1 hash produced by Autopsy; every metadata artifact Autopsy extracts from this host image will have the same hash value, thereby ensuring all metadata from the same host is linked the graph. The relationship between the metadata artifact being processed and the source host is established by the aut:sourceHost predicate-object pair—Line 33. The identified by clause—Line 38—ensures that the metadata artifact is uniquely identified by the sourceFile path and sourceHost properties; this allows files of the same sourceFile path from a different sourceHost, to exist within the graph.

With artifacts encoded to knowledge graphs, researchers can develop software agents and knowledge discovery methods, to infer relationships between artifacts through the use of encoded expert knowledge—from practitioners—in the form of Semantic Web rules. Listing 3 presents a SWRL rule that encodes knowledge about the relationship of two files based on the time they were created.

Listing 3: A SWRL rule to infer that two files are the same file, with one being created earlier than the other.

```
Metadata(?A) ^ Metadata(?B) ^ sourceFileMd5(?A,?hash)
  ^ sourceFileMd5(?B,?hash) ^ createdDateTime(?A,?
  dateA) ^ createdDateTime(?B,?dateB) ^ swrlb:lessThan
  (?dateA,?dateB) -> sameFileCreatedEarlier(?A,?B)
```

The SWRL rule in Listing 3 encodes expert knowledge to infer relationships between artifacts. It determines that if the Metadata of **File A** and **File B** have the same sourceFileMd5 hash, but **File A** has an earlier createdDateTime than **File B**, a relationship can be inferred. This rule could be used on a set of artifacts to infer knowledge about file transfers between hosts. Figure 2 shows an example of a knowledge graph generated from the GFEAR script in Listing 2 with the SWRL rule in Listing 3 applied. In this example, the graph shows three files extracted from two hosts, with the SWRL rule inferring the relationship and source of the files—based on the time they were created—that share the same MD5 hash.

By further expanding the artifacts that can be added to the knowledge graph, and combining this with more SWRL inference rules from expert knowledge, more abstract activities can be extrapolated from artifacts, such as:

- Network communication with a known malicious IP address, indicating that a malicious actor may control the host.
- Memory artifacts indicating the presence of a known malicious process, potentially communicating with a malicious IP address, suggesting host compromise.
- Communication between two hosts prior to the creation of a malicious file on one host, indicating that the file may have been transferred.
- Combining these indicators to infer lateral movement of a malicious actor within a network.

Practitioners can use queries developed in conjunction with these rules by researchers to query knowledge from the graph, and generate reports that can be used in court or other legal settings.

The FEAR Framework Autopsy module provides an interface for executing SPARQL queries that can be used to extract and visualize information, including any knowledge that may have been inferred through SWRL rules.

4.4 Digital Forensic Artifact Mapping Library

By creating an open-source library of scripts based on existing, well-known ontologies in the digital forensics space, researchers and practitioners gain access to a comprehensive, out-of-the-box framework requiring minimal configuration for constructing knowledge graphs from digital forensic artifacts. For researchers, this enables a focus on core research goals rather than on preliminary tasks—such as artifact extraction and knowledge graph construction.

As a part of the research project, an ontology—defining Autopsy terms—is being developed to integrate seamlessly with the proposed framework and language. This ontology, and its implementation in the language, will provide researchers the ability to generate semantic knowledge graphs using known Autopsy terms from the framework out-of-the-box without the need for any programming. Researchers interested in exploring digital forensic knowledge

graphs—particularly through knowledge discovery, or software agents—can leverage existing ontologies and mapping scripts; bypassing ontology creation, integration, and framework development. This enables more rapid semantic knowledge graph generation, allowing for a focus on exploring use cases, refining methodologies, and pursuing novel research directions.
A standardized method for mapping digital artifacts into semantic knowledge graphs streamlines the research-to-practice pipeline, enabling practitioners to quickly adopt emerging research without needing custom environments for specific outputs.

5 Discussion

Research using semantic knowledge graphs in the field of digital forensics currently requires either (1) proprietary, closed-source, or proof-of-concept code to construct knowledge graphs to commence research, or (2) use specially built tools for transforming data on a one-off basis for a research activity with very specific schemas.
While current methods can produce RDF representations, extending them to handle new artifacts typically demands ongoing programming effort. Simplifying the encoding of forensic artifacts will significantly advance research by enabling efficient use of software agents, automated reasoning, knowledge discovery techniques such as SWRL, and other emerging applications in the field.
For researchers in the digital forensics field who focus on knowledge graphs and their various use cases, a language that supports the sharing of mapping knowledge is crucial. Such a language can significantly reduce the burden of developing frameworks for producing knowledge graphs from artifacts, without the need for extensive programming skills. This advantage is particularly beneficial when developing new ontologies, as a standardized mapping language integrated with industry standard tools for the extraction of digital forensic artifacts can simplify the research-to-practice process.
Regarding the adoption of the FEAR Framework, the widespread access via an open-source licensing model of both the framework and the library of knowledge mappings helps to mitigate challenges relating to cost barriers. Usability is also a challenge that has been considered. The development and distribution of the GFEAR mapping library—containing scripts for widely used ontologies such as UCO and CASE,[19] as well as emerging ones like Autopsy and NORIA—will enable immediate knowledge graph construction without further configuration or programming. This will assist in the adoption of the framework and serve as a shared, ever-evolving resource to be easily incorporated into existing digital forensic workflows. Further, we recognize that not all researchers and practitioners will use Autopsy or the suites the framework is tested with. Adoption within these system poses a challenge, but the self-contained nature of artifact processing from standard serialization formats—such as JSON, CSV, and XML—offers significant potential to simplify and expedite the process. By leveraging standalone components for processing digital forensic artifacts, the complexity of building integrations for knowledge graph construction can be mitigated. Moreover, the availability of an open-source library of mapping scripts helps alleviate some of the development burden. However, integrating these components and mapping scripts effectively into existing workflows may still require effort and expertise, which could be a barrier for widespread adoption.
The sharing of mapping scripts throughout the community can facilitate collaboration among researchers and practitioners while fostering innovation within the field. Using the FEAR Framework and its pre-built libraries, researchers can easily create semantic knowledge graphs from digital forensic artifacts with minimal programming. This process seamlessly integrates with industry tools, ensuring research and discoveries are applied in real-world investigations.
By supporting various data formats and allowing data import via a command-line tool, the visualization component of the FEAR Framework Autopsy module can query information from a broad range of sources beyond Autopsy. By drawing in data from memory dumps, network packet captures, and other forensic artifacts, researchers and practitioners gain a centralized interface to interact with evidence from all aspects of an investigation.
The research continues to refine the process of importing such data, ensuring that the FEAR Framework can seamlessly leverage diverse forensic inputs for more comprehensive and efficient investigations.

6 Conclusion

The FEAR Framework and GFEAR language, developed as part of a doctoral research project, aim to address challenges with integrating semantic knowledge graph advancements in the field of digital forensics by sharing knowledge and democratizing access to extracted artifacts from de facto industry standard tools. Providing an integration with industry standard tools and incorporating a domain-specific language will help foster further research, and accelerating the widespread adoption of semantic knowledge graphs in digital forensics.
The contributions of the proposed FEAR Framework include:

- Simplified access to digital forensic artifacts extracted from de facto industry standard tools through a tight integration that removes the need to duplicate programming work to access common artifacts.
- A standardized approach for representing knowledge relating to mapping extracted artifacts into knowledge graphs through a domain-specific declarative language: GFEAR.
- Facilitating the sharing of knowledge within the digital forensics community through open-source libraries of mapping scripts, thereby reducing time-consuming development for researchers, improving the research-to-practice process.

The framework's open-source nature and integration strategy reduces the burden on both researchers and practitioners. Researchers can focus on developing novel methods, models, agents, and ontologies while leveraging the frameworks standardized integration, querying and visualization capabilities. This accelerates access to cutting-edge research outputs for practitioners using existing industry tools, removing the need to learn or implement new systems or custom data conversions. By bridging the gap between research and practical applications, the framework promotes a more efficient and collaborative research-to-practice pipeline in digital forensics.

[19] https://ontology.caseontology.org/documentation/

References

[1] F. Amato, G. Cozzolino, A. Mazzeo, and N. Mazzocca. [n. d.]. Correlation of digital evidences in forensic investigation through semantic technologies. In *Proceedings - 31st IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2017*, T. Enokido, M. Takizawa, C. Y. Lin, H. H. Hsu, and L. Barolli (Eds.). Institute of Electrical and Electronics Engineers Inc., 668–673. doi:10.1109/WAINA.2017.4

[2] A. Asasfeh, N. A. Al-Dmour, H. Al Hamadi, W. Mansoor, and T. M. Ghazal. 2023. Exploring Cyber Investigators: An In-Depth Examination of the Field of Digital Forensics. In *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. 0084–0088. doi:10.1109/DASC/PiCom/CBDCom/Cy59711.2023.10361449

[3] Emlyn Butterfield, Mark Diljon, Stephen Miller, and Z. Cliffe Schreuders. 2018. Automated Digital Forensics. https://core.ac.uk/download/pdf/159081654.pdf

[4] Y. Chabot, A. Bertaux, T. Kechadi, and C. Nicolle. 2014. Reconstruction and semantic analysis of digital forensic timelines. In *Revue des Nouvelles Technologies de l'Information*, Vol. E.26. Hermann-Editions, 521–524.

[5] Y. Chabot, A. Bertaux, C. Nicolle, and T. Kechadi. 2014. Automatic timeline construction and analysis for computer forensics purposes. In *Proceedings - 2014 IEEE Joint Intelligence and Security Informatics Conference, JISIC 2014*. Institute of Electrical and Electronics Engineers Inc., 276–279. doi:10.1109/JISIC.2014.54

[6] Yoan Chabot, Aurélie Bertaux, Christophe Nicolle, and Tahar Kechadi. 2015. An ontology-based approach for the reconstruction and analysis of digital incidents timelines. *Digital Investigation* 15 (2015), 83–100. doi:10.1016/j.diin.2015.07.005

[7] P. Chikul, H. Bahsi, and O. Maennel. 2021. An Ontology Engineering Case Study for Advanced Digital Forensic Analysis. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, C. Attiogbé and S. Ben Yahia (Eds.), Vol. 12732 LNCS. Springer Science and Business Media Deutschland GmbH, 67–74. doi:10.1007/978-3-030-78428-7_6

[8] Alfredo Cuzzocrea and Giuseppe Pirrò. 2016. A semantic-web-technology-based framework for supporting knowledge-driven digital forensics. Association for Computing Machinery, 58–66. doi:10.1145/3012071.3012099

[9] Brenda Jin, Saurabh Sahni, and Amir Shevat. 2018. *Designing Web APIs: Building APIs That Developers Love.* " O'Reilly Media, Inc.".

[10] D. Kahvedžić, N. Kuncik, and T. Kechadi. [n. d.]. A data mining framework for digital forensics investigations. In *WMSCI 2007 - The 11th World Multi-Conference on Systemics, Cybernetics and Informatics, Jointly with the 13th International Conference on Information Systems Analysis and Synthesis, ISAS 2007 - Proc.*, Vol. 3. 158–163. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84869763621&partnerID=40&md5=9cbfea1b23344bf43a6b679266f71fd5

[11] Kai Liu, Fei Wang, Zhaoyun Ding, Sheng Liang, Zhengfei Yu, and Yun Zhou. 2022. Recent progress of using knowledge graph for cybersecurity. *Electronics* 11, 15 (2022), 2287.

[12] Yuke Ma, Yonggang Chen, Yanjun Wang, Jun Yu, Yanting Li, Jinyu Lu, and Yong Wang. 2024. The Advancement of Knowledge Graphs in Cybersecurity: A Comprehensive Overview. In *Computational and Experimental Simulations in Engineering*, Shaofan Li (Ed.). Springer International Publishing, 65–103.

[13] Martin Morgenstern, Johannes Fähndrich, and Wilfried Honekamp. 2022. Ontology in the digital forensics domain: A scoping review. In *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft für Informatik*, Vol. P-326. 71–80. doi:10.18420/inf2022_05

[14] D. J. Schelkoph, G. L. Peterson, and J. S. Okolica. 2019. Digital Forensics Event Graph Reconstruction. In *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, F. Breitinger and I. Baggili (Eds.), Vol. 259. Springer Verlag, 185–203. doi:10.1007/978-3-030-05487-8_10

[15] Leslie F. Sikos. 2020. AI in digital forensics: Ontology engineering for cybercrime investigations. *WIREs Forensic Science* 3, 3 (2020). doi:10.1002/wfs2.1394

[16] L. F. Sikos. 2023. Cybersecurity knowledge graphs. *Knowledge and Information Systems* (2023). doi:10.1007/s10115-023-01860-3

[17] T. J. Silva. 2024. How Ontologies Have Supported Digital Forensics: Review and Recommendations. *Forensic Science Review* 36, 2 (2024), 99–125.

[18] Amit Singhal. 2012. Introducing the knowledge graph: things, not strings. Google blog. https://blog.google/products/search/introducing-knowledge-graph-things-not/

[19] Lionel Tailhardat, Yoan Chabot, and Raphael Troncy. 2024. *NORIA-O: An Ontology for Anomaly Detection and Incident Management in ICT Systems.* Springer Nature Switzerland, 21–39. doi:10.1007/978-3-031-60635-9_2

[20] W. Xu and D. Xu. 2022. Visualizing and Reasoning about Presentable Digital Forensic Evidence with Knowledge Graphs. In *2022 19th Annual International Conference on Privacy, Security and Trust, PST 2022*. doi:10.1109/PST55820.2022.9851972

[21] Ruipeng Zhang and Mengjun Xie. 2023. ForensiQ: A Knowledge Graph Question Answering System for IoT Forensics. In *Proceedings of the 8th ACM/iEEE Conference on Internet of Things Design and Implementation*. Springer Nature Switzerland, 300–314. doi:10.1007/978-3-031-56583-0_20

[22] Ruipeng Zhang and Mengjun Xie. 2023. A Knowledge Graph Question Answering Approach to IoT Forensics. In *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation* (San Antonio, TX, USA) *(IoTDI '23)*. Association for Computing Machinery, New York, NY, USA, 446–447. doi:10.1145/3576842.3589161

7 Appendices

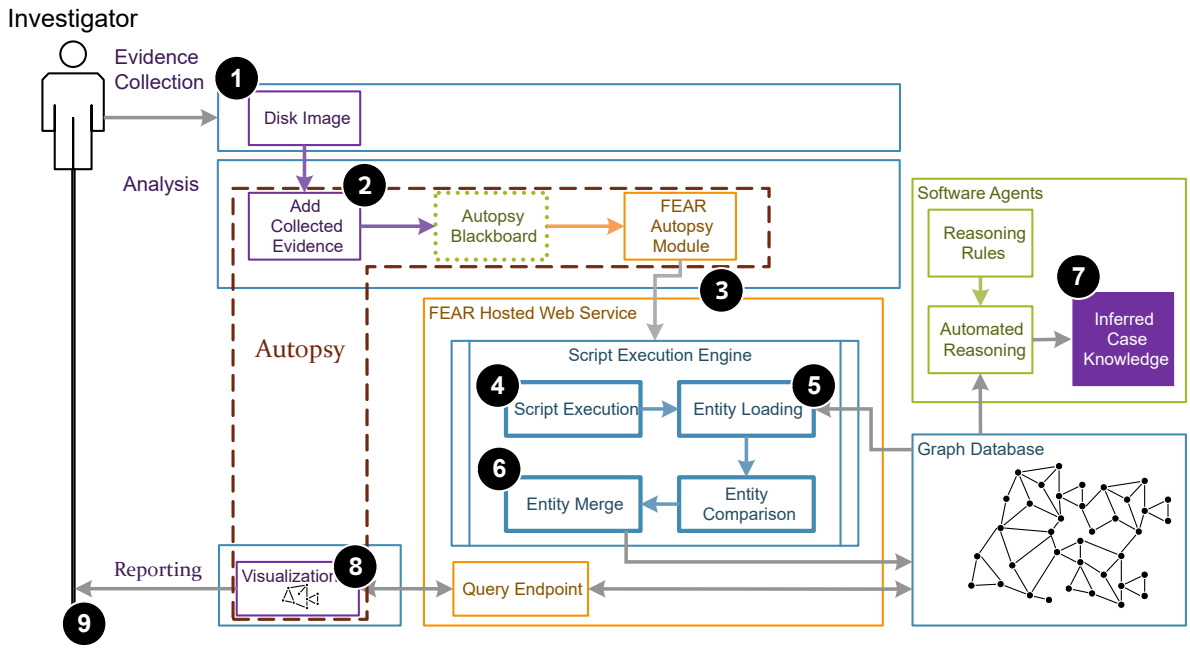


Figure 1: A demonstration of the workflow for how Forensic Extraction and Representation can be integrated into the digital forensic investigation process.

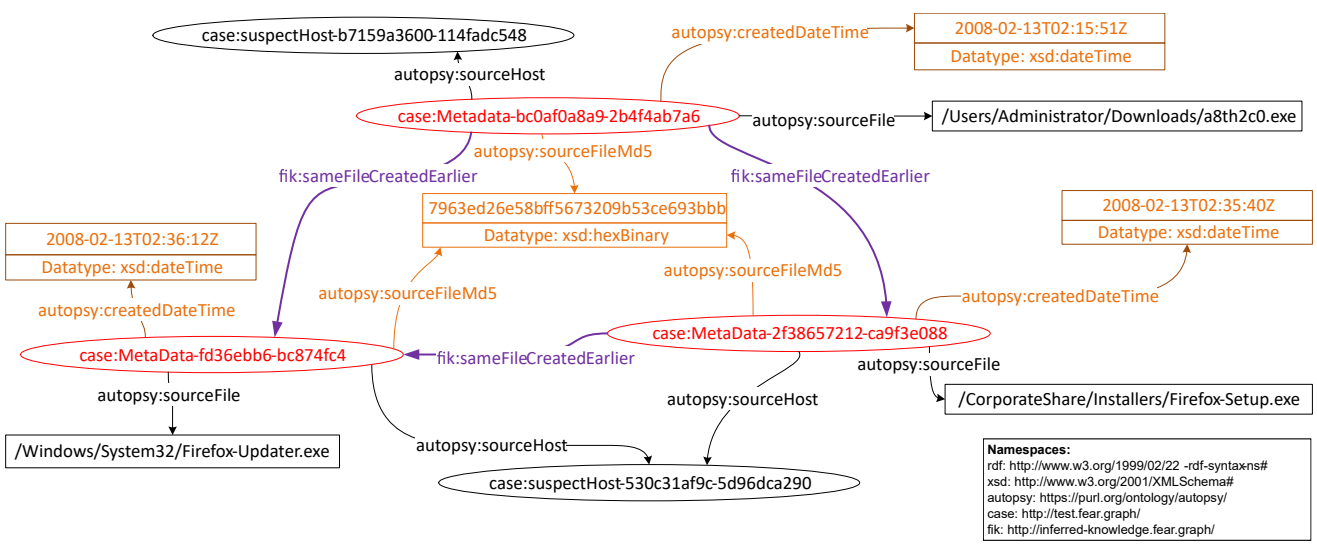


Figure 2: The graph, colored for emphasis, shows a scenario of a file moving through a network, of which the SWRL rule in Listing 3 could infer new knowledge about where a file has originated from. The relevant metadata nodes' (in red) and properties' reference in the SWRL rule (in orange) are used to infer relationships between metadata nodes—in purple—about files that share the same MD5 hash, created at different times.

Understanding Strategies and Challenges of Timestamp Tampering for Improved Digital Forensic Event Reconstruction

Céline Vanini
School of Criminal Justice
University of Lausanne
Lausanne, Switzerland
celine.vanini@unil.ch

Jan Gruber
Department of Computer Science
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)
Erlangen, Germany
jan.gruber@fau.de

Christopher Hargreaves
Department of Computer Science
University of Oxford
Oxford, United Kingdom
christopher.hargreaves@cs.ox.ac.uk

Zinaida Benenson
Department of Computer Science
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)
Erlangen, Germany
zinaida.benenson@fau.de

Felix Freiling
Department of Computer Science
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)
Erlangen, Germany
felix.freiling@fau.de

Frank Breitinger*
Institute of Computer Science
University of Augsburg
Augsburg, Germany
frank.breitinger@uni-a.de

Abstract

Timestamps play a pivotal role in digital forensic event reconstruction, but due to their non-essential nature, tampering or manipulation of timestamps is possible by users in multiple ways, even on running systems. This has a significant effect on the reliability of the results from applying a timeline analysis as part of an investigation. We investigate the problem of users tampering with timestamps on a running ("live") system. While prior work has shown that digital evidence tampering is hard, we focus on the question of *why* this is so. By performing a qualitative user study with advanced university students, we derive factors that influence the reliability of successful tampering, such as the individual knowledge about temporal traces, and technical restrictions to change them. These insights help to assess the reliability of timestamps from individual artifacts that are used for event reconstruction and subsequently reduce the risk of misinterpretations.

CCS Concepts

• **Applied computing** → **Investigation techniques**; **Evidence collection, storage and analysis**; *System forensics.*

Keywords

Event reconstruction, Tampering, User study, Tamper resistance factors, Digital forensics investigation

*The work was performed while being affiliated with the University of Lausanne.

DFDS 2025, Brno, Czech Republic

ACM ISBN 979-8-4007-1076-6/25/04
https://doi.org/10.1145/3712716.3712727

ACM Reference Format:
Céline Vanini, Jan Gruber, Christopher Hargreaves, Zinaida Benenson, Felix Freiling, and Frank Breitinger. 2025. Understanding Strategies and Challenges of Timestamp Tampering for Improved Digital Forensic Event Reconstruction. In *Digital Forensics Doctoral Symposium (DFDS 2025), April 01, 2025, Brno, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3712716.3712727

1 Introduction

The dangers of evidence tampering, i.e., the intentional act of altering, concealing or falsifying evidence [21], are of great concern to law enforcement agencies since fictitious or fake traces can easily alter or sabotage a criminal investigation [4]. While there is a high awareness of the risks of tampering with physical evidence, tampering of digital evidence is much less understood. While some scholars consider tampering to be easier when dealing with evidence that is in digital form [1, 11], others claim that it is at least similarly difficult in special cases [22]. Understanding the risks of digital evidence tampering is particularly important for *timestamps* because, firstly, timestamps play a pivotal role in digital forensic event reconstruction to establish the order in which certain actions happened, and secondly, timestamp manipulation is a commonly applied indicator removal technique in security incidents [15, 23].

1.1 Related work

Despite work that has structured tampering activities under the heading of anti-forensics [5, 9, 10], and confirmed by the literature review of Neale [18], unfortunately, the understanding of digital evidence tampering in general and of timestamp tampering in particular is rather shallow. Previous research has primarily focused on specific technical contexts of timestamp tampering, such as manipulating file metadata on NTFS [8, 17, 19], or on technical approaches for timestamp tampering detection. These attempts aim to find inconsistencies between timestamps, e.g., the violation of general time rules [8], of causal relationships between timestamps [13, 27], or inconsistent relations to implicit timing information like sequence numbers [6]. Even if such inconsistencies are detected, it may still be unclear whether these are due to intentional tampering.

For example, if some timestamps have been set to January 1, 1970, explanations can vary from intentional "timestomping" [15] to an unintentional non-initialized Unix timestamp [12]. Thus, it is still necessary to have a solid understanding of adversarial (tampering) behaviors such that investigators can assess the effect of tampering on the meaning of evidence.

Some previous user studies [7, 16, 22] also focused on the tampering detection problem. In these experiments, participants had to produce convincing forgeries that should be taken as originals when analyzed by other participants in the study. This allowed to assess empirically how convincing forgeries were. Due to their empirical study design, these experiments did not attempt to investigate the difficulties of tampering with specific artifacts. Furthermore, the experiment setup considered the extreme case where a perpetrator has *full control* over every bit of the system, an approach we call *dead tampering*. Still, the quantitative insights from these works indicate that tampering may not be as easy as it can be expected, but it is highly unclear *why* this may be so.

In contrast to the worst case assumptions often made in the literature, in practice adversaries do not operate under idealized circumstances. When accessing a compromised system, perpetrators usually have less control because they are under time pressure, lack knowledge of alternative methods, or need to modify the system while logged in remotely. Also, while less experienced users may be able to perform actions such as changing a value in a database or editing some text in a file, they may not be capable of booting to an alternative environment and performing low-level manipulations.

In such situations, adversaries are forced to manipulate data on the system they are currently using, a process we call *live tampering*. Unlike dead tampering, live tampering is arguably not only more realistic than dead tampering (e.g., remote access scenario, or full volume encryption), it also introduces new challenges, as the act of tampering itself generates traces on the system being manipulated. When these traces are directly embedded in the manipulated data itself, we refer to them as *first-order traces*. For example, manipulating browser evidence tends to result in contradictory information in the browser history and browser cache [7]. In addition to first-order traces, tampering actions can also leave indirect indicators, which we call *second-order traces*, such as traces of anti-forensic tool usage.

1.2 A qualitative look at live tampering

In this paper, we report on the results of a user study in live tampering. While prior work has shown that digital evidence tampering is hard, we focus on the question of *why* this is so and therefore have chosen to apply qualitative research methods, i.e., questionnaires and semi-structured interviews. Our general goal was to understand how study participants chose their strategies and allocate their resources while solving a live timestamp tampering task. To investigate this, we conducted a user study with 10 advanced university students, who tampered with a live system based on a fictitious scenario, in which an adversary attempts to swap two events to cover their tracks. Not all adversaries are specialist hackers or advanced persistent threats (APT) in practice, so our protagonist was assumed to be a regular user rather than a sophisticated adversary. As we will show, the exploration of the dynamics of such tampering also leads to understanding the difficulty of tampering with specific artifacts. This can help develop further strategies for reliable event reconstruction, since methods for representing the uncertainty of traces, e.g., the C-Scale ('Strength of Evidence scale') [3], include an estimate of the number of sources that agree, but also the difficulty of tampering with those sources.

Our focus was on the following research questions:

- RQ1 **Approach to tampering:** What strategies do adversaries employ in planning and executing tampering with the temporal order of events?
- RQ2 **Awareness and precautions of traces left by the manipulation**: How do adversaries deal with (new) traces stemming from their manipulations?
- RQ3 **Barriers to the tampering process:** What makes an artifact more difficult to tamper with compared to another?

Overall, this work provides the following contributions: (1) we present the design, implementation, and assessment of the first user study on tampering with timestamps on a running system (live tampering), (2) we provide clear indications that adversaries differentiate between first-order and second-order traces and adapt their tampering strategy accordingly, (3) we identify strategies of live tampering that involve the opportunistic application of tampering actions along the hierarchical abstraction layers, which indicates the mental application of a rational *tampering budget*, leading to a concentration on artifacts being easier to manipulate, and (4) we establish an understanding of the reliability of tampering indicators and derive factors that influence the tamper resistance of an artifact.

For additional details and arguments, we refer the reader to the extended version of this paper [25].

2 User study design

This section opens with a description of the tampering task scenario, followed by a detailed explanation of the user study, which consisted of four phases: (1) a pre-tampering questionnaire, (2) a tampering task, (3) a post-tampering questionnaire, and (4) a set of semi-structured interviews.

2.1 Participants

The user study was conducted during an advanced lecture on digital forensics at the Friedrich-Alexander Universität (FAU) in the fall of 2023, with 35 registered students. Of these, 10 students completed phases (2) and (3), and three of them additionally participated in the interviews (4).

2.2 Scenario of the tampering task

The user study assumes that a perpetrator who has full control over a running system (administrator) wants to cover the tracks of their deeds by swapping two events E_2 and E_3, as depicted in Figure 1, i.e., make an examiner believe that the download of illegal material (E_3) occurred before the perpetrator became aware of its illegality (E_2). More details about the scenario can be found in Appendix A.

In the real world, swapping events may impact the criminal intent ('Mens rea'). Here, the updated sequence ($E_1 - E_3 - E_2$) suggests that the criminal liability of the downloaded material was unknown to the suspect when the web browsing was conducted. This particular type of tampering allows us to learn more than

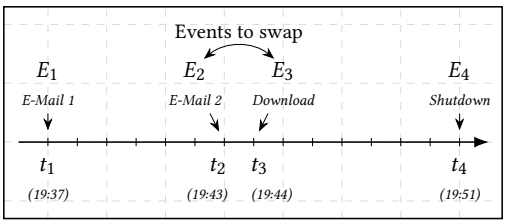


Figure 1: Artificial sequence of events $E_1 - E_4$ imagined for the tampering task.

demanding trace deletion or addition, as we are interested in the resilience of artifacts and the probability that they are taken into account.

2.3 Pre-tampering questionnaire

The user study started with a pre-tampering questionnaire aiming at gathering a set of background information about participants. The questionnaire included a total of 15 single/multiple-choice(s) questions centered around the respondents' experience. The questions concerned their teaching curriculum such as their course of study or graduate degree (Q1-Q4, Q10, and Q11), their experience with digital forensics lectures (Q8-Q9), and practical work (Q12-Q14). The remaining questions aimed at capturing their workload and motivation (Q5, Q6), as well as the effort they were willing to put into the study (Q7, Q15).

2.4 Tampering task

Participants were provided with the full virtual machine image of a Windows 10 Home workstation (VM turned off, administrator account credentials provided) after events $E_1 - E_4$ had been performed. They were asked to act like the suspect and swap events E_2 and E_3. All actions had to be performed on the running system (working on the disk images directly was prohibited) and the time boundaries E_1 and E_4 had be to respected (E_2, E_3 had to remain within E_1, E_4). Participants were given two weeks to complete the tampering task. They had to return a logbook containing notes of their actions and the VM itself.

2.5 Post-tampering questionnaire

Participants who returned a forged VM were asked to fill out a post-tampering questionnaire, which aimed at capturing *how* the students went about the task. It comprised 22 questions and was separated into three sections referring to a different time of the tampering task: *before*, *during*, and *after*. Questions relating to *before* and *after* focused on the assessment of specific artifacts[1] that the respondent knew or manipulated (Q1-Q10 and Q15-Q22). For example, some questions asked participants which artifacts they manipulated and to rank them according to the perceived difficulty of tampering. The middle part targeted performed activities, e.g., how certain actions were performed (Q11-Q14). Respondents were asked to place themselves at these times and provide detailed insights into their experiences, decisions, and observations. They were encouraged to use their logbook to answer the questions.

2.6 Semi-structured interviews

Respondents were invited to take part in semi-structured interviews where three individuals accepted. These face-to-face interviews followed a generic outline with a set of open-ended questions articulated around the preliminary findings from the post-tampering questionnaire. The intention was to extract further insights into their experiences during the tampering task and qualitatively evaluate the difficulty of tampering with different artifacts.

2.7 Ethics

The ethics commission at the university in which the user studies and interviews were conducted does not handle non-medical studies for explicit approval, and therefore instead the experimental protocol followed their general data protection and ethics rules. Participation in the study was voluntary and integrated into the course as a non-graded exercise.

2.8 Data analysis

We adopted an iterative approach for the analysis of qualitative data derived from the questionnaires, logbooks, and interview transcriptions. Open-ended responses were inductively coded to extract thematic patterns or trends [20]. The logbooks were not coded as they primarily described the tampering process undertaken by each participant but were used to improve our understanding of their manipulations.

2.9 Limitations

The user study has three limitations: (1) it included only 10 participants which is small and may not sufficiently capture the variability of behaviors and strategies; (2) all participants were students sharing similar backgrounds and experiences; and (3) only one tampering scenario was developed which is only one possible instance of manipulation, and therefore gives only answers to the research question from the viewpoint of the scenario. An adversary may have different strategies such as hiding data or artifact wiping [5]. Given these limitations, it is important to understand that our goal was a qualitative and explorative study which allows for a detailed examination of each case. It can be seen as a first step towards better understanding strategies, challenges, and artifacts, as well as revealing new research directions such as factors influencing the trustworthiness of timestamps. While some may argue that identical backgrounds are a disadvantage, our study found that participants followed different strategies which would be less informative with a large/diverse group. Nevertheless, we acknowledge that a larger-scale study would add value and therefore we provide detailed descriptions of our tampering task.[2]

[1] We compiled a list of relevant, existing Win10 artifacts based on the *Plaso* documentation [14], i.e., existing parsers which can be found in Appendix B.

[2] All details to repeat this study are publicly available here [24].

3 Results: Tampering Preparation

This and the following two sections summarize the results we extracted from the questionnaires and interviews. We begin with insights on how participants prepared for the tampering task.

3.1 Participant background, experience, and knowledge

We anticipated that participants have differences in prior knowledge of (Windows) forensics which was confirmed through the post-tampering questionnaire (Q3). Based on their responses, participants can be categorized into two knowledge levels: novices (Participants 5, 6, 8, and 9) who had limited Windows forensics experience, and semi-experienced participants, who had a generic knowledge of browser-related artifacts. With few exceptions, participants consistently marked Windows Registry sources, the `$USNjrnl`, and the Thunderbird Global Database as unknown. In contrast, most were familiar with Firefox-related data, Thunderbird's Inbox file, the `$MFT`, the `$RECYCLE.BIN`, and Windows event logs. During the interviews, we learned that most participants regularly used Linux.

3.2 Participant initial thoughts and designs

The first question of the post-tampering questionnaire invited the participants to reflect on their initial strategies, before starting the task. All participants agreed that accomplishing the task of making it appear as if E_3 happened before E_2 required modifying one of the two events. The common strategy was therefore to select a fixed event that would act as a pivot point for re-arranging other events. Interestingly, we observed that half chose E_2 (opening the second email) as their reference event while the other half preferred E_3 (browsing and downloading with Firefox). Participants within groups using the same pivot point expressed similar decisive factors in their choice to re-arrange the other event:

Level of knowledge: Participants 4, 6, and 7, who chose Thunderbird (E_2) as their pivot point, mentioned their greater familiarity with Firefox—gained from their introductory lecture—as a key factor in their decision of manipulating Firefox. In contrast, Participant 10 chose Firefox as the pivot point, having limited prior knowledge and having heard that Thunderbird would be easier to manipulate.

Volume of linked artifacts: The volume of linked artifacts refers to the number of artifacts associated with E_2 and E_3 that would need to be modified. All participants who selected E_3 as their pivot point agreed that Thunderbird appeared easier to manipulate than Firefox because it involved modifying a smaller volume of data. Four participants expressed concerns about the higher number of files to manipulate with the Firefox strategy, which they felt increased the likelihood of errors. Participant 2 noted: "My idea was to change as little as possible to minimize the potential for errors". Similarly, Participants 1 and 9 were worried about generating excessive second-order traces.

Correlation with remotely stored information: Another important consideration was the limited control over external data storage. This concern originated from the possibility that an unaltered "true copy" of the targeted data might exist in a remote location. This concern dissuaded Participants 3, 5, and 6 of the E_2 pivot point group from manipulating Thunderbird. For instance, Participant 5 indicated that the "full analysis of other sources than just the machine [would] probably give the real course of events away". These sources include an ISP, a DNS, or a mail server.

Maintaining internal artifact consistency: This factor refers to the relationship between the organization of an artifact and the data it contains. For instance, SQLite databases organize their entries following a specific allocation strategy. Hence, tampering with timestamps in entries may be exposed when looking at the order of (raw) entries and identifiers in the database. Participant 9 highlighted this issue: manipulating Firefox artifacts can be "identified by logical inconsistencies such as timestamps not matching the order in which events are listed in a cache/log file".

3.3 Preparation

In the post-tampering questionnaire, 9 out of 10 students reported that they had undertaken preparation beforehand. We learned that the types of preparation steps varied from one participant to another. Some mentioned gathering information through literature, forums, and/or tools review, while others took a hands-on, 'learning by doing' approach within the VM.

To evaluate the influence of preparation on their knowledge of Windows artifacts, we compared responses from Q3 and Q15. We observed that informed participants showed minimal changes in knowledge, while novices experienced significant gains, reaching similar overall scores. The former relied on literature reviews, testing and AI-based inquiries, focusing on artifacts related to the event to modify (Firefox- or Thunderbird-related artifacts). Overall, all participants, except Participant 9, achieved a comparable baseline knowledge before commencing the task.

4 Results: Tampering Actions

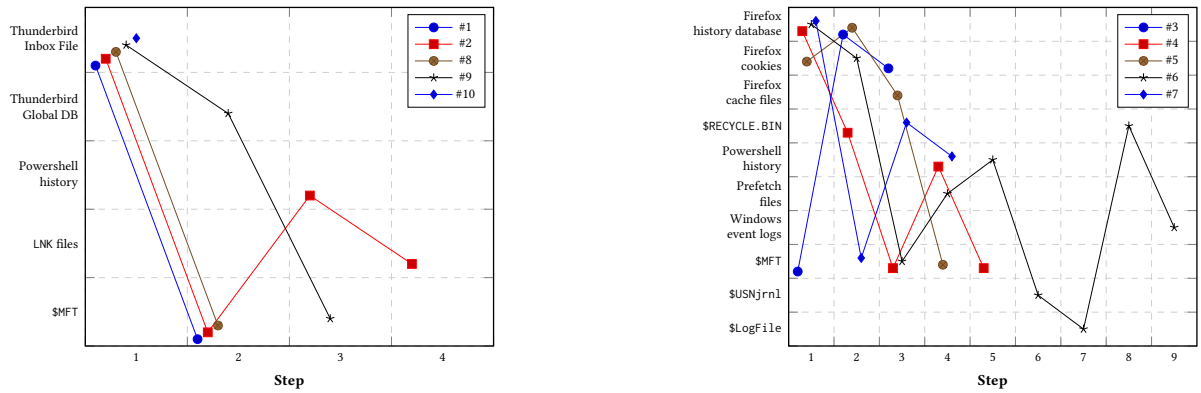
This section describes the execution of the tampering task: the tampering process itself, as well as the handling of second-order traces.

4.1 Tampering approaches

As discussed before, all participants shared the common strategy to decide on a reference event between E_2 and E_3 that is used as a pivot point to swap the other event.

Figure 2 illustrates the sequence of manipulations performed within the groups using the same pivot point. On the y-axis are artifacts that were manipulated and/or removed by students within each group, ordered according to their hierarchical position in the abstraction layers of the system (higher levels: application to lower levels: file system) [2]. The x-axis illustrates the sequence of manipulation steps undertaken by participants.

While the sequence of manipulations is intrinsically linked to the conflicting goal of dealing with second-order traces, as further discussed in Section 4.2, the graphs show a visible trend of descent through abstraction layers among participants. After choosing their pivot point, all students started by manipulating artifacts directly related to the unfixed event to re-arrange, at the application level. For instance, all participants in the Firefox pivot point group started with the manipulation of timestamps within Thunderbird's Inbox file. We then observe that further manipulations concerned artifacts in lower layers, starting with artifacts at the OS layer and addressing file system artifacts (apart from the `$MFT`) last.



(a) Thunderbird group

(b) Firefox group

Figure 2: Manipulation sequences showing participants as connected lines over time.

In regards to specific timestamp manipulations, participants attempted to maintain consistency across manipulations to fit the scenario, e.g., avoiding instances where a picture's download time precedes the visit to the corresponding web page. To achieve consistency, most participants added or removed (according to the strategy) constant time offsets.

4.2 Dealing with second-order traces

All participants showed concern about second-order traces. This carefulness is reflected in their strategy design (Section 3.2), choice of tooling, and sequence of manipulations. Overall, three distinct approaches emerged from the findings.

Clandestine approach: Participants using the clandestine approach proactively minimized the creation of second-order traces, rather than focusing on their removal after they had been generated. Three participants anticipated the creation of second-order traces by carefully choosing the tools they would use in the task. This included preferring tools that hypothetically leave fewer traces (e.g., command prompt over Powershell), benign-looking tools like DB Browser for SQLite, avoiding installing anti-forensic software, and using portable versions loaded on a USB stick.

Tampering-focused approach: This approach concerns two participants who pertained to the act of tampering itself without considering the generation of second-order traces, as "it would be practically impossible and would continue indefinitely". The only action undertaken by both participants was the update of the file system metadata of Thunderbird's Inbox file and the transfer of files to the `$RECYCLE.BIN`.

Mixed approach: The mixed approach involves participants who were mindful of second-order traces generation and actively engaged in recursively removing them from the system. Participants began by focusing on their main target (Firefox- or Thunderbird-related artifacts). They then adopted a wrapping-the-onion method, systematically working to erase not only the second-order traces of their tampering but also the subsequent layers that emerged from their efforts to conceal this tampering, and so on, in an iterative process. They also showed sophisticated efforts to minimize the generation of second-order traces by employing methods such as tool name obfuscation and downloading scripts from a web server.

5 Results: Tampering Difficulties

Participants experienced failures that forced them to adjust their strategy which can be divided into two classes: (1) practical challenges and (2) perceptions. On top of that, most participants expressed having encountered technical difficulties when manipulating specific artifacts, which are discussed in Section 6.3.

5.1 Practical challenges in tampering

These aspects cover the range of issues that the participants faced when working on the task, such as the installation or use of tools. When facing such practical challenges, we can see that participants were forced to make compromises on certain aspects of their strategy. For instance, Participant 3 encountered difficulties when installing Powershell modules, forcing them to install external software and deviating from their original plan of minimizing suspicions. Another participant had their external software recognized as a virus by Windows Defender and was forced to obfuscate its name and download it via a personal web server.

5.2 Perceptions of knowledge and relevance

Some participants adjusted their strategy in response to soft factors like their knowledge level and the perceived relevance of manipulating a specific artifact. For instance, Participant 6 chose not to alter the Windows event logs due to the absence of clear indications of their actions. Similarly, Participant 4 did not manipulate these logs, lacking the knowledge about where to find pertinent information and how to modify it.

6 Discussion and reflection

This section revisits the research questions in light of our findings. To generalize our results, we also discuss the technical aspects identified that affect the tamper resistance of specific artifacts.

6.1 Adversary's perspective

RQ1: What strategies do adversaries employ in planning and executing tampering with the temporal order of events? When tasked with swapping two events, we observed that all tamperers followed the same strategy: deciding on a reference event and using it as a pivot point to re-order the other event. The sequence of manipulations in re-ordering this unfixed event appeared to be closely related to the placement of each connected artifact in the abstraction layers. Only a few participants manipulated traces in the lower layers along this hierarchy, which from a forensic perspective, is relevant. Despite planning, several participants encountered unexpected difficulties during their manipulation process or reported various factors influencing their ability to tamper with certain artifacts. Those are either inherent to the intrinsic nature of the targeted artifact, such as its complexity, or the operating system/setting in which the artifact resides, e.g., the availability of tools to facilitate the manipulation in that environment.

RQ2: How do adversaries deal with (new) traces stemming from their manipulations? Adversarial actions generate new artifacts. It is particularly difficult to maintain a comprehensive overview of all newly created artifacts, and attempting to manipulate every subsequent artifact can become an endless endeavor. In addition, the manipulation may create inconsistencies, such as relative sequences or implicit timestamps, that can be detected. Of the approaches used by participants, the mixed approach is the most sophisticated, where participants not only anticipated the creation of second-order traces but also recursively removed the traces of their deeds. For the tamperers, this is a never-ending conflict of goals between the manipulation of n-order traces associated with the event to re-order and reducing the generation of $n + 1$-order traces originating from the manipulation process itself.

6.2 Artifacts

RQ3: What makes an artifact more difficult to tamper with compared to another? Findings suggest that several artifacts are more "tamper-proof" compared to others based on the technical challenges and difficulties faced by our tamperers: (1) the correlation with remotely stored information, maintaining internal artifact consistency and the volume of linked artifacts (defined in Section 3.2 and combined here as a *implicit time information* factor), as well as additional challenges such as (2) the placement in the abstraction layers (see Section 4.1), (3) the existence of integrity checks, (4) the assigned permissions, (5) encryption, and (6) the availability of software to edit artifacts on the system. These technical aspects are complemented by soft factors such as perception or knowledge which depend on the experience/sophistication of an adversary.

6.3 From tamper-proof to resistance factors

As artifacts differ in their suitability to be manipulated, this means that they have special features (or factors) that make them easy or difficult to manipulate. These factors are examined briefly in this section and are discussed in more detail in our previous work [26].

Permissions: Various operations on Windows are protected via User Account Control (UAC) and require Administrator privileges. Consequently, one factor is the level of permissions required to modify an artifact. On many system configurations, including the one in this study, the user is an Administrator of the system in question. Therefore, in many cases, the UAC interface presents a little barrier to accessing the protected files, other than clicking 'Allow'. On the other hand, running a command as Admin may trigger other events or be logged.

Integrity checks: An artifact might have embedded mechanisms used to verify that data has not been altered or corrupted. For example, email signatures are generated over the content of the email, which may include time information. Modifying this information may result in an invalid signature and trigger suspicion. This becomes even more challenging when monitoring systems such as auditd (Linux) are used.

Software availability: Manipulations require some sort of tool, which may be a text editor, regedit (both available on most Windows systems), or more specialized tools such as a hex editor or database modification tool. New tools may require an installation creating artifacts of their existence. Some may qualify as anti-forensics tools according to Conlan et al. [5] while others may be less suspicious. If no tool is available, an adversary may have to reverse engineer an artifact, which requires sophisticated knowledge.

Placement within the software stack: This factor refers to the level at which an artifact or process is positioned within the hierarchical layers of software architecture. As discussed in Section 6.1, this impacts the modification or accessibility of an artifact.

Implicit timing information: In addition to timestamps, manipulations may lead to logical inconsistencies within an artifact. For instance, a database appends new entries with an increasing ID which means potential timestamps in a column should also increase. This implicit timing information may not be known to the tamperer and can now be integrated into digital forensic timelines [6]. In addition, implicit timing information may also be evidence that cannot be controlled due to its residence on an external source.

Encryption/format: The artifact requiring manipulation may be in a proprietary format or even encrypted (this is related to software availability) impeding a modification. Considerations include the type of encryption software implementation, as well as whether the keys are available, recoverable, or not.

The possibility of evidence tampering should be considered during the investigation, encouraging examiners to look for inconsistencies. The factors presented here directly impact the tamper resistance of traces and offer a vast potential to improve the interpretation of tampering because they provide clear guidance in determining the reliability of artifacts and the likelihood of them being changed. Given that adversaries have a finite amount of resources leading to a confined *tampering budget*, they will likely fail to produce perfect forgeries. Consequently, these tamper resistance factors can be used to evaluate artifacts that contain such discrepancies and reconstructing what may have caused them.

7 Conclusion

There is great interest in concealing crime. One way to do this is to tamper with digital traces afterward. This tampering poses a significant threat to the reliability of forensic event reconstruction. Our user study sheds light on previously unexplored aspects of live system tampering. Through a user study involving 10 graduate students tasked with swapping two past events, we identified a

general tampering strategy which is to decide on one event that would act as a pivot point to re-arrange the other event. We also concluded that manipulations generate new traces that need to be hidden or manipulated as well, resulting in an endless cycle of manipulations. Compared to dead tampering, this conflict of goals between tampering and removing the traces of tampering increases the difficulty of creating perfect forgeries. Furthermore, we generalized our results and derived factors that influence the tamper resistance of artifacts, such as embedded integrity checks and artifact placement within the software stack. These factors need more discussion, however, they guide practitioners about the reliability of an artifact especially if two contradicting artifacts are found. We believe that the qualitative findings from our study on live tampering will improve the understanding of criminal efforts to conceal their activities and aid in their reconstruction.

Use of AI writing assistance

At least one author of this paper used ChatGPT-4 and the Grammarly plugin to assist in correcting typographical and grammatical errors and refining the phrasing of certain sentences. All recommendations were thoroughly evaluated and modified when needed before being integrated into this paper.

Acknowledgments

We wish to thank the students from the course on "Advanced Forensic Computing" at FAU for their participation. We also thank the anonymous reviewers for their helpful comments on previous versions of the paper. This work was supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (grant number 393541319/GRK2475/2-2024).

References

[1] Michael A. Caloyannides. 2003. Digital "evidence" and reasonable doubt. *IEEE Security & Privacy* 1, 6 (2003), 89–91. https://doi.org/10.1109/MSECP.2003.1266366

[2] Brian D. Carrier. 2003. Defining Digital Forensic Examination and Analysis Tool Using Abstraction Layers. *International Journal of Digital Evidence* 1, 4 (2003), 1–12. http://www.utica.edu/academic/institutes/ecii/publications/articles/A04C3F91-AFBB-FC13-4A2E0F13203BA980.pdf

[3] Eoghan Casey. 2020. Standardization of forming and expressing preliminary evaluative opinions on digital evidence. *Forensic Science International: Digital Investigation* 32 (March 2020), 200888. https://doi.org/10.1016/j.fsidi.2019.200888

[4] William J. Chisum and Brent E Turvey. 2000. Evidence dynamics: Locard's exchange principle & crime reconstruction. *Journal of Behavioral Profiling* 1, 1 (2000), 1–15.

[5] Kevin Conlan, Ibrahim Baggili, and Frank Breitinger. 2016. Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy. *Digital Investigation* 18 (Aug. 2016), S66–S75. https://doi.org/10.1016/j.diin.2016.04.006

[6] Lisa M. Dreier, Céline Vanini, Christopher J. Hargreaves, Frank Breitinger, and Felix Freiling. 2024. Beyond timestamps: Integrating implicit timing information into digital forensic timelines. *Forensic Science International: Digital Investigation* 49 (2024), 301755. https://doi.org/10.1016/j.fsidi.2024.301755

[7] Felix Freiling and Leonhard Hösch. 2018. Controlled experiments in digital evidence tampering. *Digital Investigation* 24 (March 2018), S83–S92. https://doi.org/10.1016/j.diin.2018.01.011

[8] Michael Galhuber and Robert Luh. 2021. Time for Truth: Forensic Analysis of NTFS Timestamps. In *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES 21)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3465481.3470016

[9] Simson Garfinkel. 2007. Anti-forensics: Techniques, detection and countermeasures. In *2nd International Conference on i-Warfare and Security*, Vol. 20087. 77–84.

[10] Ryan Harris. 2006. Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem. *Digital Investigation* 3, Supplement (2006), 44–49. https://doi.org/10.1016/j.diin.2006.06.005

[11] Xiaodong Lin. 2018. *Introductory Computer Forensics: A Hands-on Practical Approach*. Springer International Publishing. https://doi.org/10.1007/978-3-030-00581-8

[12] Farhad Manjoo. 2001. Unix Tick Tocks to a Billion. *Wired* (2001). https://www.wired.com/2001/09/unix-tick-tocks-to-a-billion/

[13] Andrew Marrington, Ibrahim Baggili, George Mohay, and Andrew Clark. 2011. CAT Detect (Computer Activity Timeline Detection): A tool for detecting inconsistency in computer activity timelines. *Digital Investigation* 8 (2011), S52–S61. https://doi.org/10.1016/j.diin.2011.05.007 The Proceedings of the Eleventh Annual DFRWS Conference.

[14] Joachim Metz. 2024. Welcome to the Plaso documentation. https://plaso.readthedocs.io/en/latest/

[15] MITRE ATT&CK. 2020. MITRE ATT&CK v15.1, Indicator Removal: Timestomp. https://attack.mitre.org/versions/v15/techniques/T1070/006/

[16] Christian Moch. 2015. *Automatisierte Erstellung von Übungsaufgaben in der digitalen Forensik*. Ph. D. Dissertation. University of Erlangen-Nuremberg. https://d-nb.info/1068781181

[17] Alji Mohamed and Chougdali Khalid. 2019. Detection of Timestamps Tampering in NTFS using Machine Learning. *Procedia Computer Science* 160 (Jan. 2019), 778–784. https://doi.org/10.1016/j.procs.2019.11.011

[18] Christopher Neale. 2023. Fool me once: A systematic review of techniques to authenticate digital artefacts. *Forensic Science International: Digital Investigation* 45 (June 2023), 301516. https://doi.org/10.1016/j.fsidi.2023.301516

[19] David Palmbach and Frank Breitinger. 2020. Artifacts for Detecting Timestamp Manipulation in NTFS on Windows and Their Reliability. *Forensic Science International: Digital Investigation* 32 (April 2020), 300920. https://doi.org/10.1016/j.fsidi.2020.300920

[20] Johnny Saldaña. 2021. *The coding manual for qualitative researchers*. SAGE Publications.

[21] Chris W. Sanchirico. 2004. Evidence Tampering. *Duke Law Journal* 53, 4 (2004), 1215–1336.

[22] Janine Schneider, Julian Wolf, and Felix Freiling. 2020. Tampering with Digital Evidence is Hard: The Case of Main Memory Images. *Forensic Science International: Digital Investigation* 32 (2020), 300924. https://doi.org/10.1016/j.fsidi.2020.300924

[23] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. 2020. *MITRE ATT&CK: Design and philosophy*. Technical Report MP180360R1. The MITRE Corporation.

[24] Céline Vanini, Jan Gruber, Christopher J. Hargreaves, Zinaida Benenson, Felix Freiling, and Frank Breitinger. 2024. Guidelines and Questionnaires for a User Study on Live Timestamp Tampering in Digital Forensic Event Reconstruction. https://doi.org/10.48657/ydrk-qa98 (Version 1.0).

[25] Céline Vanini, Jan Gruber, Christopher Hargreaves, Zinaida Benenson, Felix Freiling, and Frank Breitinger. 2024. Strategies and Challenges of Timestamp Tampering for Improved Digital Forensic Event Reconstruction (extended version). https://doi.org/10.48550/arXiv.2501.00175 arXiv:2501.00175 [cs.CR]

[26] Céline Vanini, Chris Hargreaves, and Frank Breitinger. 2024. Evaluating tamper resistance of digital forensic artifacts during event reconstruction. https://doi.org/10.48550/arXiv.2412.12814 arXiv:2412.12814 [cs.CR]

[27] Svein Yngvar Willassen. 2008. Finding Evidence of Antedating in Digital Investigations. In *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security (ARES '08)*. IEEE Computer Society, USA, 26–32. https://doi.org/10.1109/ARES.2008.149

A Scenario details

The following case scenario and task were given to the participants:

> Albert A. is accused of the illegal possession of "rhinoceros" images. In November 2023, the police seized his computer in his home and found several rhino images. Albert A. uses his computer at home for private purposes. Albert A. claims that he came across these images by accident, not knowing that they were illegal. In contrast, the prosecution claims that Albert A. knew that rhino images were illegal before he downloaded the images.
>
> You are given the full computer (shut down virtual machine) of Albert A.'s computer after actions $E_1 - E_4$ have been finished. After completing E_3, Albert A. thinks that sequence $E_1 - E_2 - E_3$ does not look good. He wants to switch actions E_2 and E_3. Play the role of Albert A. Boot the system and manipulate the

computer such that "it looks as if" E_3 happened before E_2. Results will be analyzed by experts assessing the sequence of actions E_1, E_2, and E_3.

In this synthetic scenario:

- E_1 (at time t_1): An email is received which asks the receiver, whether he has already seen 'Rhinocerotidae', which is 'really, really hot material'.
- E_2 (at time t_2): Another email is received in which the sender clearly states that 'Rhinocerotidae' is a term referring to illegal material. The user opens the message and views the attachment.
- E_3 (at time t_3): The user opens a browser and issues a search query for 'Rhinocerotidae', visits the Wikipedia website on 'Rhinoceros', and downloads several rhino images.
- E_4 (at time t_4): The user shuts down the computer.

B Windows 10 Forensic Artifacts

Before designing the questionnaire, we compiled a list of relevant, existing Windows 10 artifacts based on the *Plaso* documentation [14]. This included artifacts within the registry and other user application or OS-related artifacts such as LNK files or the `$RECYCLE.BIN`. In addition, we added several artifacts that we deemed relevant regarding our tampering scenario, but are currently not considered by Plaso such as Thunderbird's Inbox file and Global database (message index system). The complete list can be found below in Table 1.

Table 1: Catalog of Windows artifacts derived from Plaso parsers.

Layers	Sources
Application	Files internal metadata Firefox cache files Firefox cookies Firefox history and downloads database Microsoft Edge cache files Microsoft Edge history and downloads database OneDrive synchronization logs Thunderbird Inbox file Thunderbird Global database
OS	Amcache (registry) Bam (registry) Jumplists LNK files OpenSavePIDMRU / LastVisitedPIDMRU (registry) Prefetch files `setupapi.dev.log` Shellbags (registry) ShimCache (registry) USB/USBSTOR (registry) UserAssist (registry) Windows Event Logs Windows timeline database

File system	`$LogFile`
	`$MFT`
	`$RECYCLE.BIN`
	`$UsnJrnl`