Open Tools, Interfaces and Metrics for Implementation Security Testing
https://optimist-ose.org/

With contributions from
Aydin Aysu (NC State University)
Gaetan Cassiers (Université Catholique de Louvain)
Fetemeh Ganji (Worcester Polytechnic Institute)
Vincent Immler (Oregon State University)
Jens-Peter Kaps (George Mason University)
Trey Marcantonio (Worcester Polytechnic Institute)
Jean-Michel Picod (Google)
Patrick Schaumont (Worcester Polytechnic Institute)
Aurelien Vasselle (eShard)

WORKING DOCUMENT
FILE FORMAT FOR TRACES: REQUIREMENTS AND GLOSSARY

This document is a list of common terms and requirements relevant
to a file format for measurement data from implementation security testing,
with an emphasis on data that represents side-channel leakage

Version History

0.5	1/6/25	First version based on Optimist Hour Meeting 4/10/24

Glossary

This section enumerates and defines common terms related to the file format for traces of side-channel leakage.

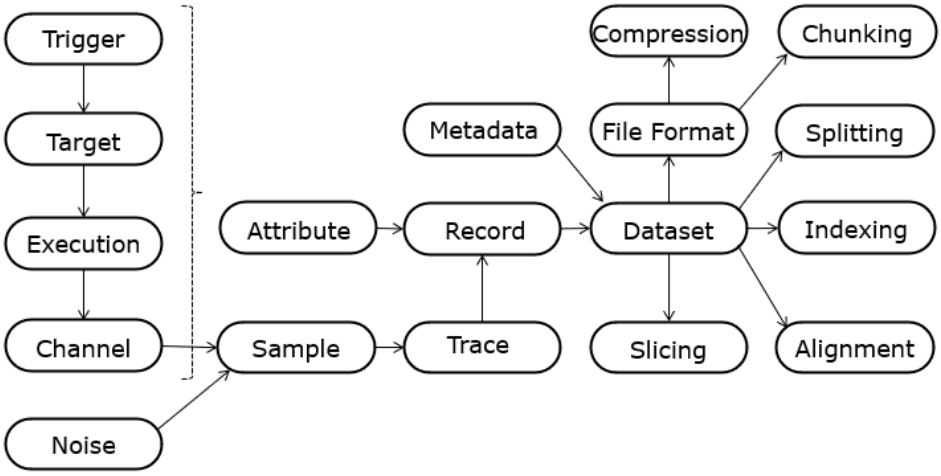

Figure 1: Common terms and their interconnections.

Channel: The source of a measurement of a physical value over time.

Trigger: A Channel used to synchronize measurements with specific operations in the Target.

Target: The object of side-channel leakage measurements.

Execution: The activity of a target associated with a single trigger.

Noise (Algorithmic vs. Environment): Unwanted variations in measurements that can obscure the desired signal in a trace.

> **Algorithmic**: Variations in the measured signal caused by the internal computations or processes of the device itself. This includes inherent randomness or variations due to the algorithm's design, state transitions, or other non-target data dependencies.
>
> **Environment**: External factors that introduce variability in the measurements, unrelated to the device's internal computations. This includes interference from the measurement setup, power supply noise, electromagnetic interference, or changes in the surrounding environment such as temperature drift.

Sample: A single measurement of a Channel corrupted by Noise.

Trace: Vector corresponding to a sequence of measurements over time of a Channel during one Execution.

Metadata: Metadata is used to provide context and supplementary information about trace sets, such as type of device being analyzed, cryptographic algorithm being executed, version of the algorithm implementation, sampling rate and resolution of the measurement equipment, environmental conditions during data collection, applied countermeasures, date and time of data acquisition.

Attributes: Attributes are named variables to store all data associated with a single execution, including the Trace. Named variables that apply to a complete dataset are Metadata.

Record: The values of all attributes specific for one execution.

Dataset: Sequence of Records with the Attributes, along with Metadata

File Format: A file format describes how the data is stored on a filesystem. In side-channel analysis, file formats are essential for managing, organizing, and processing large trace datasets and accompanying metadata.

Compression: Compression is a technique used to reduce the size of a file, which can be beneficial for storage, transmission, and computation.

Alignment: The process of adjusting traces to account for variations in timing or other properties.

Slicing: Creating a new dataset as a subset of an existing Dataset.

- **Record Slicing:** taking a subset of the Records of the dataset.
- **Vector/Trace Slicing:** taking a subset of the indices in the Vector associated to a Variable.

Indexing: Selecting a specific record, attribute and/or trace based on their position in the set, leading to Record Index, Attribute Index and Trace Index.

- **Fancy Indexing:** Indexing by means of an array or list of indices, allowing for non-contiguous or non-sequential selection of elements.
- Contiguous Indexing: Indexing corresponding to an integer interval of indices.

Splitting: Partitioning the records of a dataset for a specific purpose, such as for machine learning experiments.

- **Training Split:** A partition of records used for model training.
- **Validating Split:** A partition of records used for model validation.
- **Testing Split:** An exclusive partition of records used for model testing.

Chunking: Structuring the File Format to optimize for specific access patterns and to align with compression, such that a selection of the data only requires decompression of selected chunks, as opposed to the whole data set.

File Format Evaluation Criteria (in alphabetic order)

This section enumerates the relevant criteria to evaluate a file format for traces of side-channel leakage.

Access Speed: Speed by which consecutive or non-consecutive traces can be loaded

Backward Compatibility: The ability of tools or systems to support older versions of a file format.

Batching: Ability to store a dataset in multiple files, in order to not exceed a maximal file size.

Interoperability: The ability of tools or systems to read and process a file format created by a different tool or system.

Dataset Integrity: Ensuring the accuracy and unaltered nature of the collected records and metadata.

Extensibility: The ability of a file format to accommodate new features and extensions without requiring major changes to the tools that process the file format.

Flexibility: The ability of a file format to support new use cases.

Network Friendliness: Ability of the file format to support access to partial datasets (batches and slices).

Open: The availability of an openly published file format specification (cfr https://en.wikipedia.org/wiki/Open_file_format).

Reproducibility: The ability of a file format to support replication of measurements.

Resiliency: Capability to detect and recover from file corruption (e.g. after an interrupted acquisition).

Scalability: The ability of the file format to handle increasing amounts of data effectively.

Simplicity: Ease of implementation to read, write and process the file format.

Storage density: File size as a function of the amount of data stored.

Support: The number of tools that can read or write the file format.

Versatility: Ability to perform a variety of functions or adapt to different tasks and environments. A versatile framework can meet diverse user needs or operate effectively in multiple contexts.

Examples

This section provides examples of existing file formats for traces of side-channel leakage, including pointers to source code and reference implementations.

Format	Open	Link
Numpy	Yes	https://github.com/numpy/numpy
TRS	Yes	https://github.com/Keysight/python-trsfile
SQLite	Yes	https://sqlite.org/
HDF5	Yes	https://github.com/HDFGroup/hdf5
Zarr	Yes	https://github.com/zarr-developers/zarr-python
Avro	Yes	https://github.com/apache/avro
Apache ORC	Yes	https://github.com/apache/orc
Parquet	Yes	https://parquet.apache.org/
json	Yes	https://www.json.org/json-en.html

Framework	File Format	Link
SCARR	Zarr	https://github.com/decryptofy/scarr
SCARED	HDF5	https://github.com/eshard/scared
LASCAR	HDF5	https://github.com/Ledger-Donjon/lascar
Chip Whisperer	Numpy	https://github.com/newaetech/chipwhisperer
SCAPEgoat	Numpy/json	https://github.com/vernamlab/SCApeGoat
SCALib	Numpy	https://scalib.readthedocs.io/en/stable/
Sedpack/SCAAML	json	https://pypi.org/project/sedpack/
RamDPA	custom	https://github.com/fuentessec/RamDPA

Concrete Example from Chip Whisperer Documentation

This section aims to provide an example of how the criteria defined in the previous section can be applied to identify aspects to consider when selecting a file format for a dataset. “Trace files in the ChipWhisperer software are defined through a configuration file, with the suffix .cfg. Any trace added to the ChipWhisperer project will have a configuration file - this file does not store data, but tells the software where it is stored and what format it is stored in. While ChipWhisperer has a "native" file format, you can also interface to existing files” [8].
This part demonstrates a potential for improvement in terms of reproducibility, flexibility, extensibility, and support. It is worth mentioning that each trace is a Numpy array.

References

[1] Tobias Schneider, Amir Moradi: Leakage Assessment Methodology - a Clear Roadmap for Side-channel Evaluations. IACR Cryptol. ePrint Arch. 2015: 207 (2015)
[2] Kostas Papagiannopoulos, Ognjen Glamocanin, Melissa Azouaoui, Dorian Ros, Francesco Regazzoni, Mirjana Stojilovic:The Side-channel Metrics Cheat Sheet. ACM Comput. Surv. 55(10): 216:1-216:38 (2023)
[3] Jonah Bosland, Stefan Ene, Peter Baumgartner, Vincent Immler: High-Performance Design Patterns and File Formats for Side-Channel Analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2024(2): 769-794 (2024)
[4] eShard. "eSTRACES: Open-source Set of Traces for Side-channel Snalysis." Accessed [11/14/24]. https://eshard.gitlab.io/estraces/index.html.
[5] Riscure Security Solutions. "TRSFile documentation." Accessed [11/24/24] https://trsfile.readthedocs.io/en/latest/index.html.
[6] Metadata explained for research data. Accessed [1/6/25]. https://zenodo.org/records/10222165
[7] Open Data & Metadata Quality. Access [1/6/25]. https://data.europa.eu/sites/default/files/d2.1.2_training_module_2.2_open_data_quality_en_edp.pdf
[8] NewAE. “File Formats” Accessed [01/08/2025]. http://wiki.newae.com/File_Formats