FRASHER -- A Framework for Automated Evaluation of Similarity Hashing

By:
Thomas Göbel (Universität der Bundeswehr München), Frieder Uhlig (Technical University Darmstadt), Harald Baier (Universität der Bundeswehr München) and Frank Breitinger (University of Lausanne)

From the proceedings of

The Digital Forensic Research Conference

DFRWS USA 2022

July 11-14, 2022

Universität der Bundeswehr München

UNIL | Université de Lausanne

TECHNISCHE UNIVERSITÄT DARMSTADT

FRASHER - A Framework for Automated Evaluation of Similarity Hashing

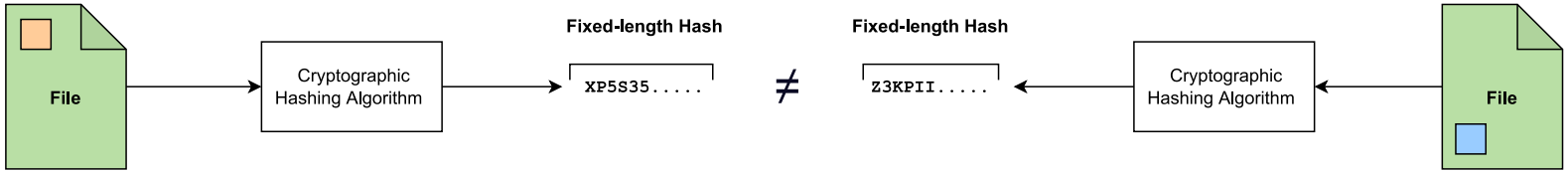
DFRWS USA 2022
Digital Forensics Research Workshop
July 11-14, 2022

Thomas Göbel, Universität der Bundeswehr, München, Germany
Frieder Uhlig, Technical University Darmstadt, Germany
Harald Baier, Universität der Bundeswehr, München, Germany
Frank Breitinger, School of Criminal Justice, University of Lausanne, Switzerland

Background

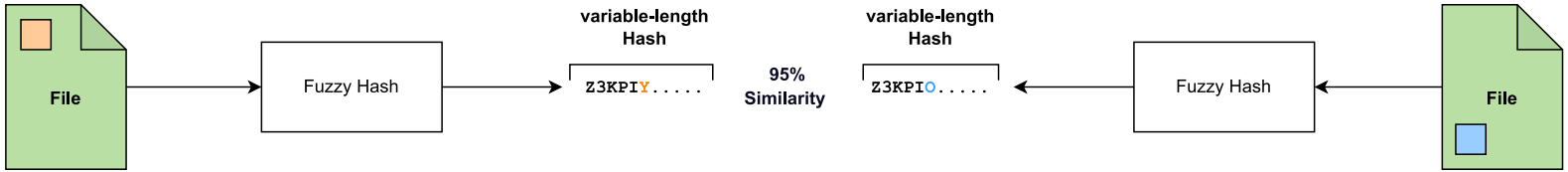
Cryptographic Hashes

- Deterministic, Collision resistant etc.
- Used to verify integrity
- Concise unique representation of a digital artifact

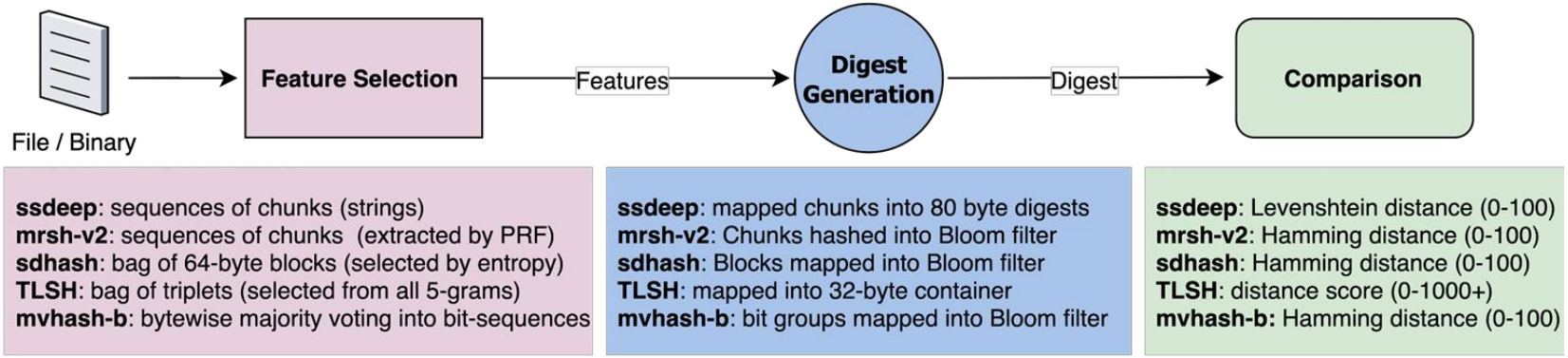


Fuzzy Hashes

- Non-cryptographic hashes (not collision resistant etc.)
- Used to determine similarity
- Concise similarity preserving representation of a digital artifact

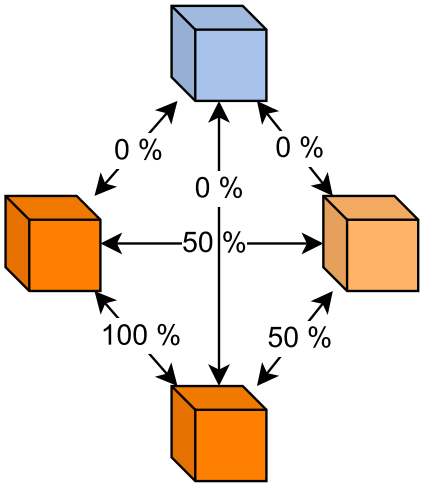


Fuzzy Hashing Schemes



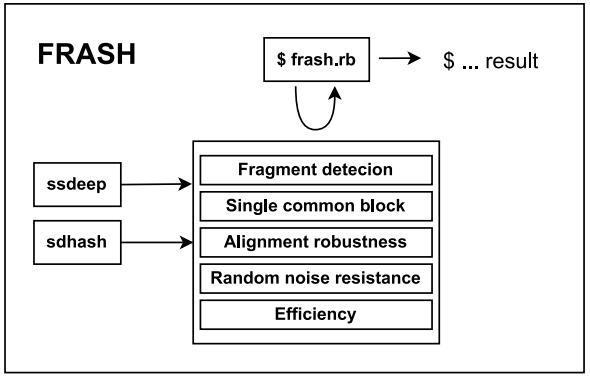
Uses of Fuzzy Hashes – Approximate Matching

- Many names – one concept (Fuzzy hashing, Approximate matching, Similarity digest comparison, Locality Sensitive Hashing)
- Powerful concept to determine relations
  - Based on text
  - Based on raw bytes



FRASH Framework

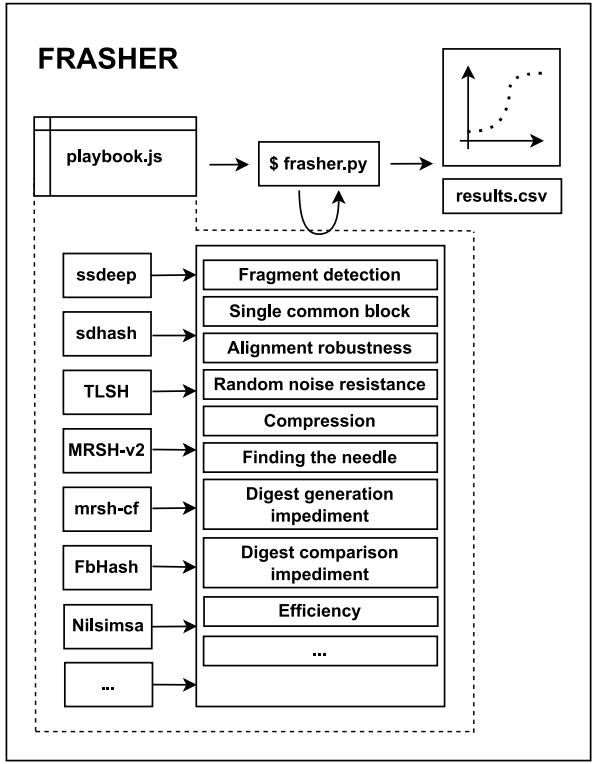
- Framework to test algorithms of similarity hashing
- Offers the following features:
  - 2 fuzzy hashing algorithms
  - 5 test cases
  - Written in Ruby



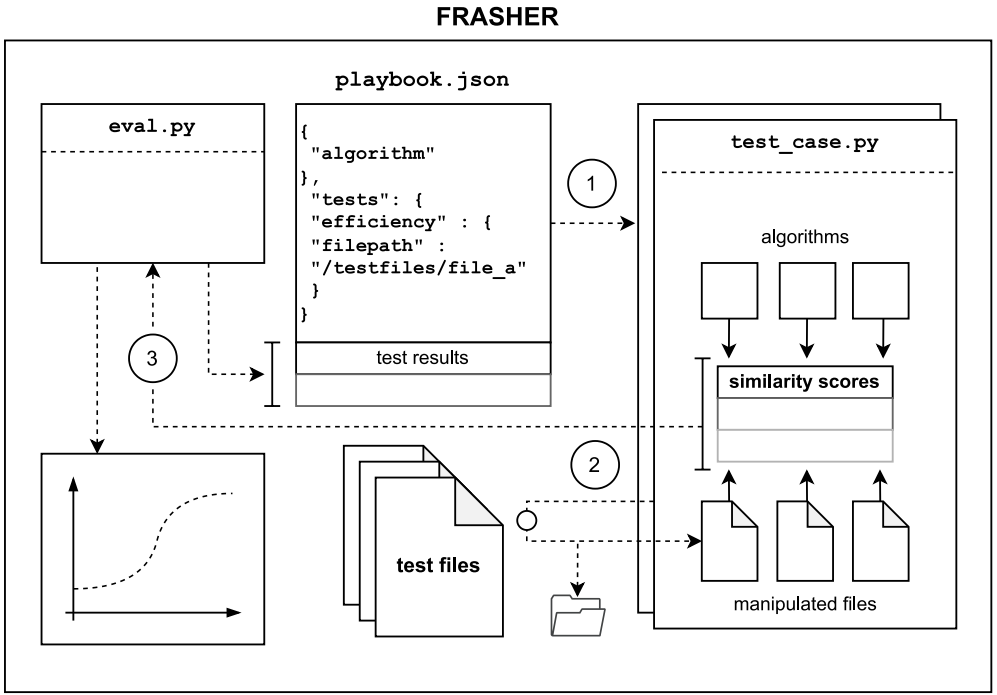
[8] Breitinger, F., Stivaktakis, G., & Baier, H. (2013). FRASH: A framework to test algorithms of similarity hashing. Digital Investigation, 10 , S50–S58. doi:10.1016/j.diin.2013.06.006. The Proceedings of the Thirteenth Annual DFRWS Conference.

FRASHER Framework (1/2)

- 7 fuzzy hashing algorithms available
  - `ssdeep`, `sdhash`, `TLSH`, `MRSH-v2`, `mrsh-cf`, `FbHash`
  - `Nilsimsa` was not evaluated (slow speed)
  - `TLSH`'s similarity score was capped at 300 and inversed
- 9 test cases so far
- Modular framework architecture (i.e., extensible in test cases, algorithms, visualization, etc.)
- Graphical representation of results
- Tests are repeatable and can be adapted
- Written in Python, runs on Linux Ubuntu 21.04



FRASHER Framework (2/2)



Efficiency

- **Generation Efficiency**
  - How *fast* can an algorithm generate a hash from a given file?
  - Algorithms have to hash the *t5-corpus* (1.9 GB, various file types)
- **Comparison Efficiency**
  - How fast can an algorithm compare given files?
  - Algorithms must compare the entire *t5-corpus* with itself (*all-vs-all*)
  - Algorithms must compare a single file with entire *t5-corpus* (*one-vs-all*)
- **Compression Efficiency**
  - How *effective* can an algorithm compress a given input?
  - Algorithms must hash the entire *t5-corpus*

Efficiency

Generation Efficiency & Comparison Efficiency

Algorithm	Digest generation (sec)	Comparison (all-vs-all) (sec)	Comparison (one-vs-all w/ t5/000001.doc) (sec)
MRSH-v2	9.084s	144.746s	1.086s
sdhash	6.393s	110.592s	0.298s
TLSH	14.791s	2.617s	0.01s
FbHash	1838.597s	NA	463.721s
mrsh-cf	10.771s	12.569s	0.507s
ssdeep	15.576s	22.688s	0.01s

Table 4: *Generation efficiency* and *Comparison efficiency* in case of All-vs-All and One-vs-All comparisons; carried out 10 times each and averaged.

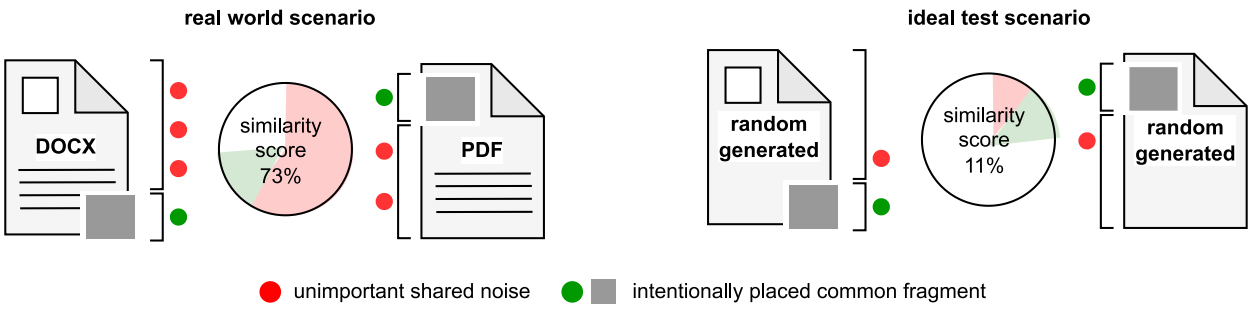
Compression Efficiency

Algorithm	Digest file size (bytes)	Compression ratio (%)
MRSH-v2	28.67 MB	1.500%
sdhash	61.52 MB	3.218%
TLSH	394.11 KB	0.021%
FbHash	19.81 GB	1036.087%
mrsh-cf	33.55 MB	1.755%
ssdeep	485.45 KB	0,025%

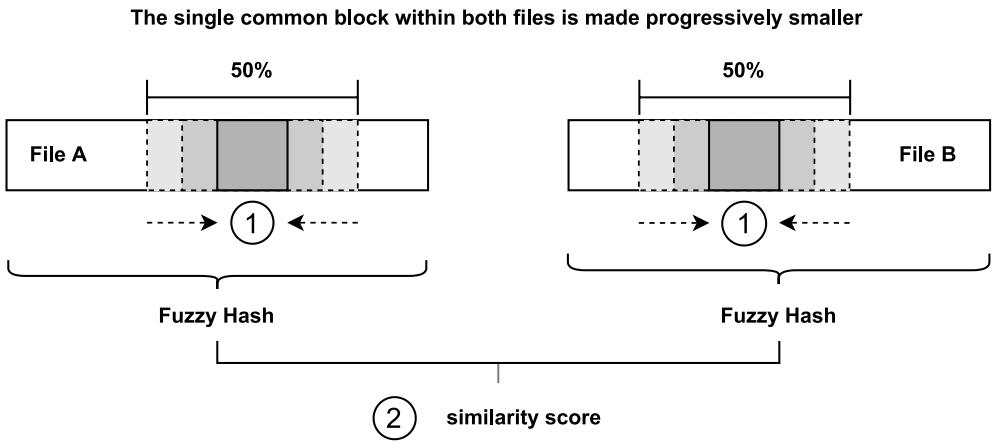
Table 5: *Compression efficiency* of tested algorithms using *t5-corpus* with total size of 1911.81 MB.

Sensitivity & Robustness Tests

- How **resilient** and **robust** are the fuzzy hashes?
- Performed with random generated data
  - Lowers the risk of unintended commonalities distorting the similarity score
  - Idealized but important setting

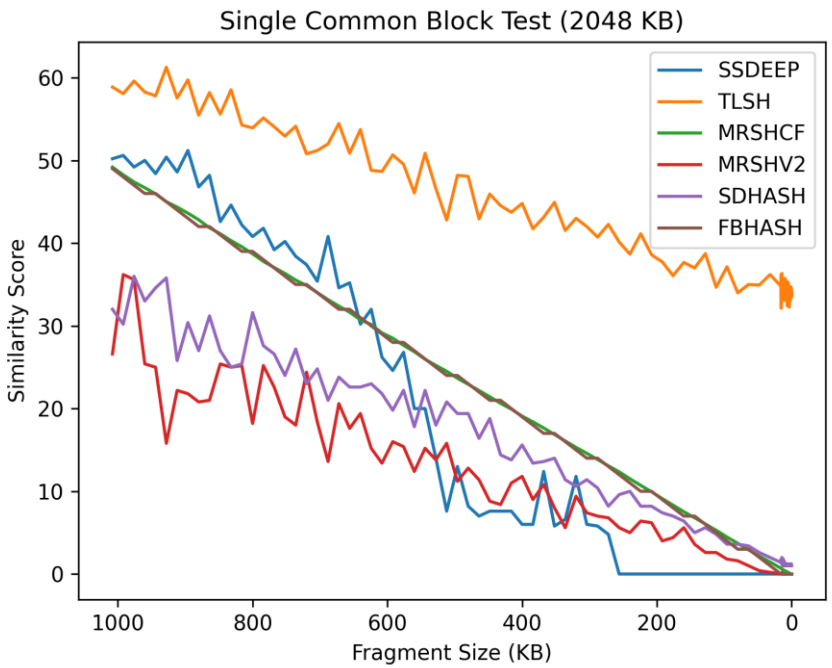


Single Common Block Test

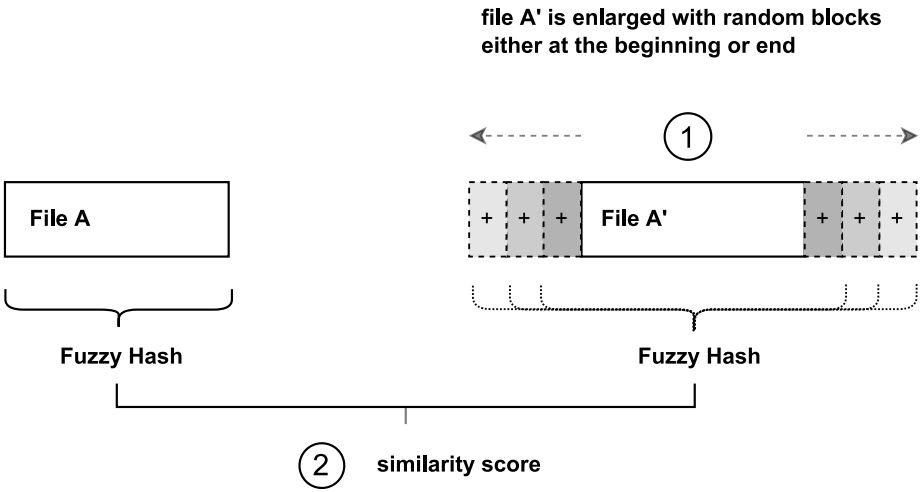


Sensitivity & Robustness

Single Common Block Test

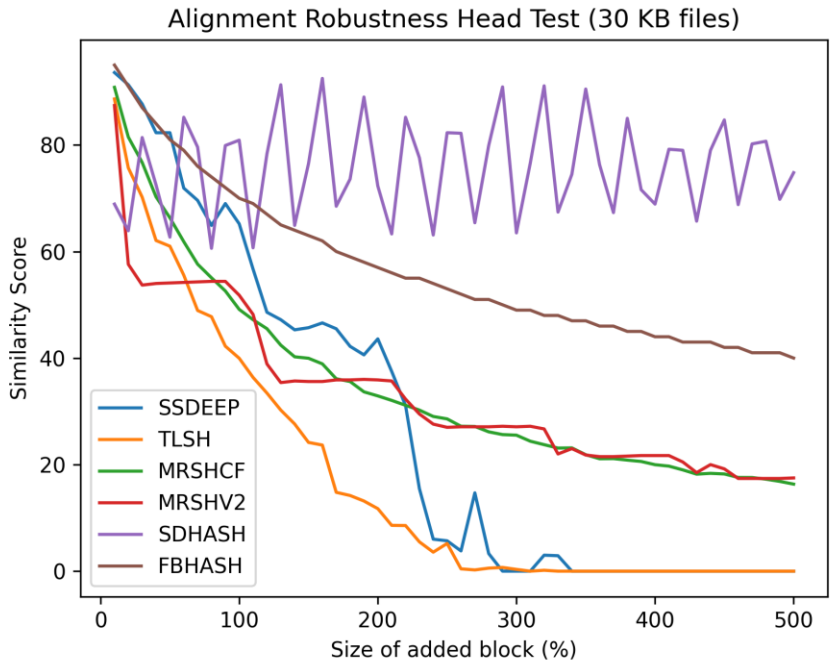


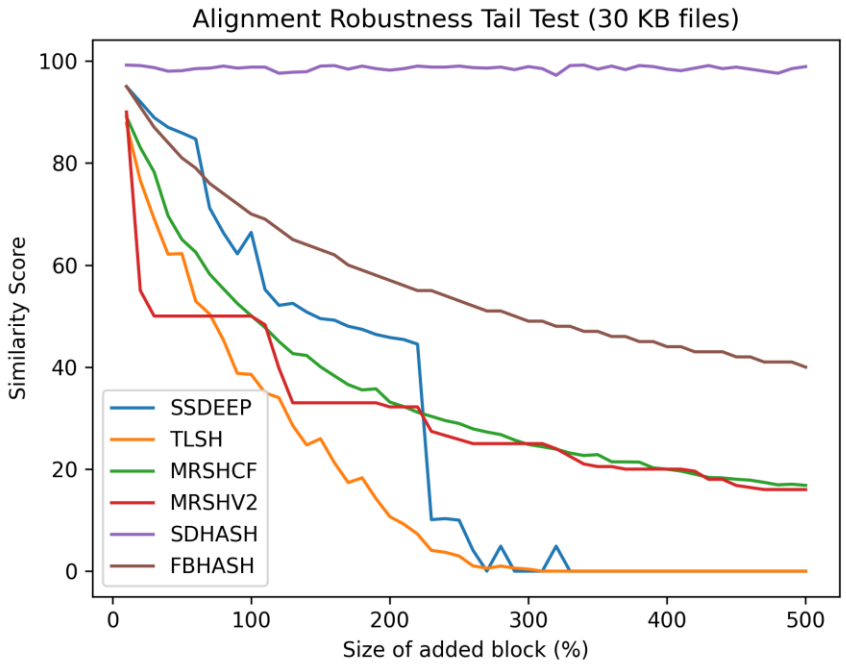
Alignment Robustness Test



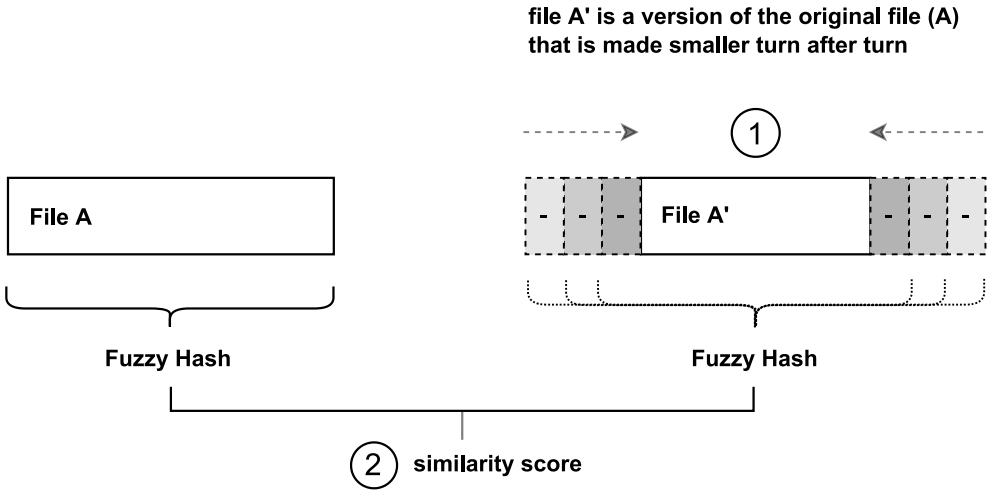
Sensitivity & Robustness

Alignment Robustness Test

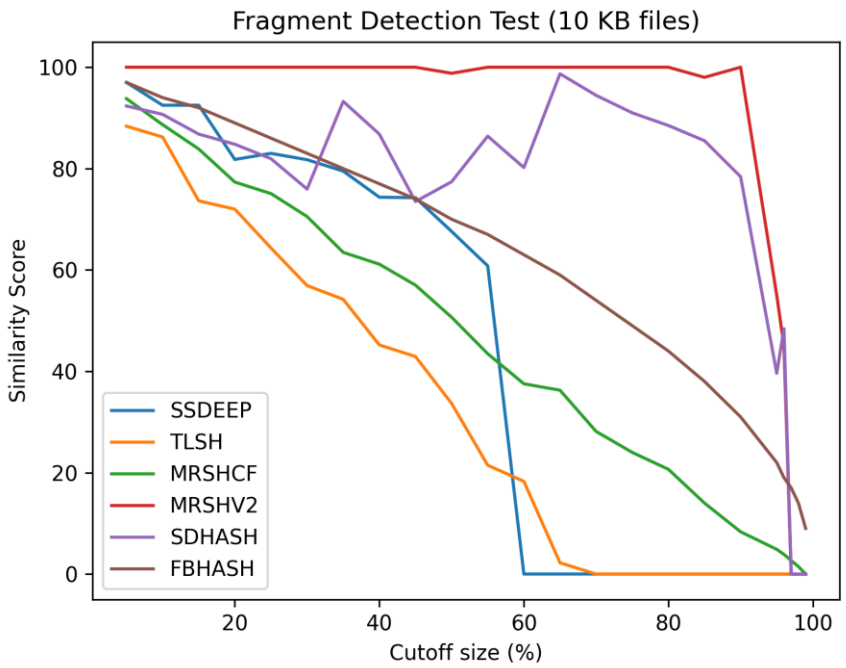




Fragment Detection Test



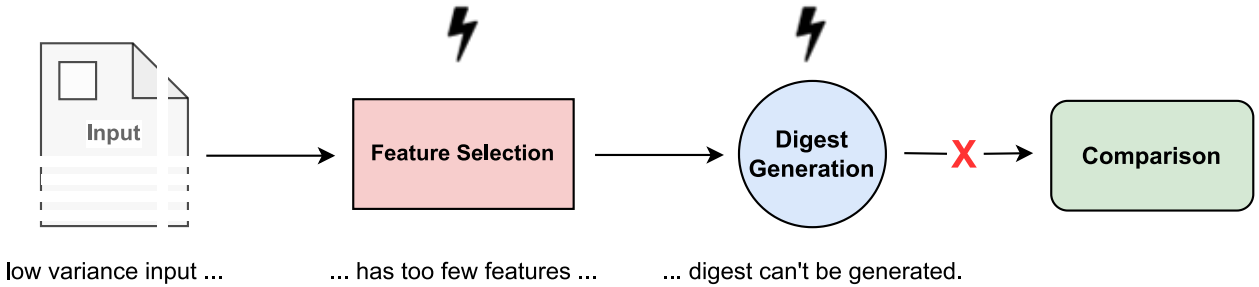
Fragment Detection Test



Adversarial Resilience Test

- **Digest Generation Impediment**
  - Could an attacker generate low variance content that cannot be hashed?
    - Text files (same length) but with varying levels of input characters are hashed (up to 100 ASCII chars)
  - What is the smallest input size for fuzzy hashes to process?
    - Minimum file size 10 bytes, maximum 3000 bytes
- **Digest Comparison Impediment**
  - Can fuzzy hashing deal with repetitive content?
    - 5000 bytes randomly generated content are chained together multiple times

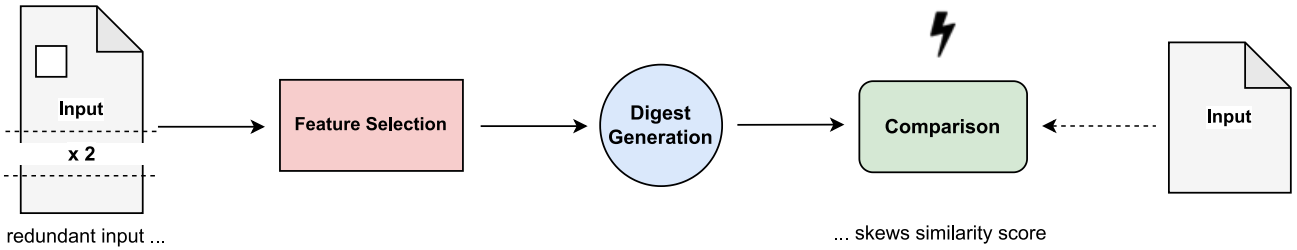
Digest Generation Impediment



Digest Generation Impediment

Algorithms	Limitations
ssdeep & mrsh-cf	Any input > 10 bytes (minimum file size in test)
TLSH	Minimum 50 bytes and more than 2 characters variance
MRSH-v2	Files smaller than 2900 bytes must contain more than 100 characters
sdhash	Minimum 512 bytes to be hashed, and at least 8 characters variance < 750 bytes
FbHash	Minimum 3 characters variance for any input size

Digest Comparison Impediment



Adversarial Resilience

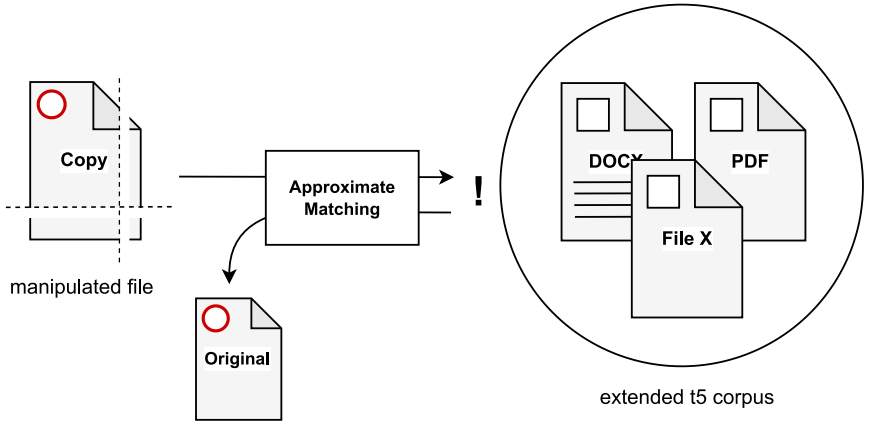
Digest Comparison Impediment

	Similarity scores					
Multiplication factor	ssdeep	TLSH	mrsh-cf	MRSH-v2	sdhash	FbHash
x 1 (5 KB)	100	100	100	100	100	100
x 2 (10 KB)	69	67	96.3	100	100	99
x 4 (20 KB)	0	39	94.3	100	100	99
x 8 (40 KB)	0	11	93.3	100	100	99
x 16 (80 KB)	0	0	92.8	100	100	99
x 32 (160 KB)	0	0	92.6	100	100	99

Table 6: Similarity scores for the same 5000 bytes of randomly generated data; concatenated several times; averaged over 10 test runs.

Finding the needle

- Real file types
- Finding the original file among many similar one's
- Different manipulation techniques: files are shortened or made bigger, compressed, partially overwritten, mixed up with similar files, etc.
- 10 *needles* in total that challenge the algorithm's ability to find the original file



Finding the needle

Finding the needle – Key takeaways

- **`ssdeep`, `TLSH`** are only usable to a certain extend
- **`MRSH-v2`** performed best followed by **`mrsh-cf`**
- **`FbHash`** is too slow to be used at scale
- **`sdhash`** is ineffective for most file types
- **Multi-Resolution-Hashing** can deal best with gzip file compression

Algorithm	File Type	needle_1	needle_2	needle_3	needle_4	needle_5	needle_6	needle_7	needle_8	needle_9	needle_10
ssdeep	PDF	10/10	10/10	10/10	10/10	2/10	1/10	10/10	10/10	0/10	10/10
	DOC	10/10	10/10	9/10	8/10	2/10	3/10	6/10	7/10	0/10	10/10
	GIF	10/10	10/10	10/10	10/10	2/10	2/10	10/10	10/10	0/10	10/10
	HTML	9/10	10/10	5/10	5/10	0/10	0/10	9/10	9/10	0/10	9/10
	JPG	10/10	10/10	10/10	10/10	5/10	9/10	9/10	9/10	5/10	10/10
	PPT	10/10	10/10	10/10	10/10	8/10	6/10	10/10	10/10	0/10	10/10
	TEXT	10/10	10/10	4/10	9/10	2/10	1/10	10/10	10/10	0/10	9/10
	XLS	10/10	10/10	5/10	6/10	0/10	0/10	8/10	9/10	0/10	10/10
	DOCX	10/10	10/10	10/10	10/10	6/10	8/10	9/10	8/10	1/10	10/10
	XLSX	10/10	10/10	10/10	10/10	7/10	5/10	10/10	8/10	0/10	10/10
	PPTX	10/10	10/10	9/10	10/10	5/10	6/10	9/10	10/10	0/10	10/10
TLSH	PDF	8/10	6/10	9/10	1/10	0/10	0/10	8/10	9/10	0/10	0/10
	DOC	5/10	5/10	5/10	4/10	0/10	0/10	1/10	1/10	0/10	2/10
	GIF	10/10	10/10	10/10	10/10	0/10	0/10	10/10	10/10	2/10	8/10
	HTML	10/10	10/10	7/10	9/10	6/10	5/10	5/10	7/10	0/10	10/10
	JPG	9/10	10/10	10/10	8/10	0/10	0/10	9/10	9/10	4/10	10/10
	PPT	8/10	5/10	10/10	5/10	0/10	0/10	7/10	7/10	5/10	4/10
	TEXT	10/10	10/10	9/10	6/10	0/10	1/10	8/10	7/10	0/10	10/10
	XLS	5/10	8/10	8/10	6/10	1/10	3/10	9/10	9/10	0/10	5/10
	DOCX	10/10	10/10	9/10	9/10	0/10	0/10	5/10	6/10	1/10	9/10
	XLSX	10/10	8/10	10/10	10/10	0/10	0/10	1/10	1/10	0/10	9/10
	PPTX	9/10	9/10	9/10	10/10	0/10	0/10	9/10	7/10	4/10	7/10
mrsh-cf	PDF	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	DOC	10/10	10/10	10/10	10/10	10/10	10/10	9/10	9/10	4/10	10/10
	GIF	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	HTML	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	JPG	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	5/10	10/10
	PPT	10/10	9/10	10/10	9/10	10/10	9/10	10/10	10/10	9/10	8/10
	TEXT	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	XLS	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	DOCX	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	1/10	10/10
	XLSX	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	PPTX	10/10	10/10	7/10	10/10	10/10	10/10	9/10	9/10	10/10	10/10
MRSH-v2	PDF	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	DOC	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	4/10	10/10
	GIF	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	HTML	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	JPG	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	3/10	10/10
	PPT	10/10	9/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	8/10
	TEXT	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	XLS	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	DOCX	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	1/10	10/10
	XLSX	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	10/10
	PPTX	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	9/10	10/10
sdhash	PDF	0/10	10/10	0/10	0/10	0/10	10/10	0/10	0/10	0/10	0/10
	DOC	0/10	10/10	0/10	0/10	0/10	10/10	0/10	0/10	0/10	0/10
	GIF	1/10	5/10	0/10	0/10	1/10	7/10	0/10	0/10	0/10	6/10
	HTML	0/10	10/10	0/10	0/10	0/10	10/10	0/10	0/10	0/10	0/10
	JPG	0/10	10/10	0/10	0/10	0/10	10/10	0/10	0/10	0/10	0/10
	PPT	0/10	10/10	0/10	0/10	0/10	10/10	0/10	0/10	0/10	0/10
	TEXT	0/10	10/10	0/10	0/10	0/10	10/10	0/10	0/10	0/10	0/10
	XLS	0/10	10/10	0/10	0/10	0/10	10/10	0/10	0/10	0/10	0/10
	DOCX	0/10	8/10	0/10	0/10	0/10	9/10	0/10	0/10	0/10	0/10
	XLSX	0/10	8/10	0/10	0/10	1/10	10/10	0/10	0/10	0/10	0/10
	PPTX	0/10	10/10	0/10	0/10	0/10	10/10	0/10	0/10	0/10	0/10

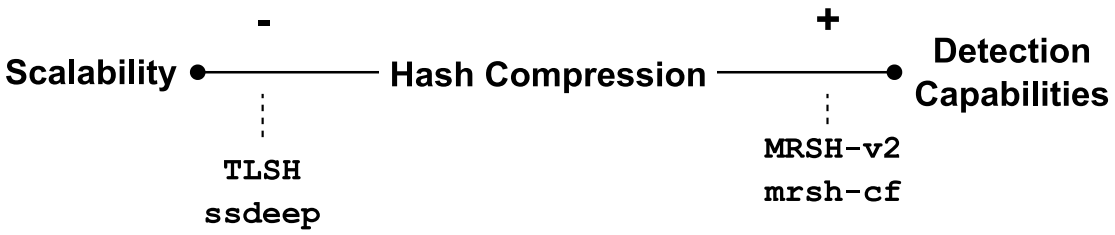
Table 7: Results for different *needle in a haystack* test cases.

A new look on fuzzy hashing

- **`FRASHER`** greatly simplifies the evaluation of byte-wise approximate matching
  - Determining limitations of algorithms and examining their performance with randomly generated files (single-common-block correlation, fragment detection, alignment robustness) or with real-world data (finding the needle test cases)
- **Multi-Resolution-Hashing (`MRSH-v2`, `mrsh-cf`)**
  - Is preferable in digital forensic field work
  - Can detect very small similarities
- **`TLSH` and `ssdeep`**
  - Allow for better integration at scale (effective compression, fast)
  - Users need to be aware of minimum shared content necessary for successful matching
- **`FbHash`** does not compress and is too slow (as of now)
- **`sdhash`** is outperformed in most test cases by newer algorithms

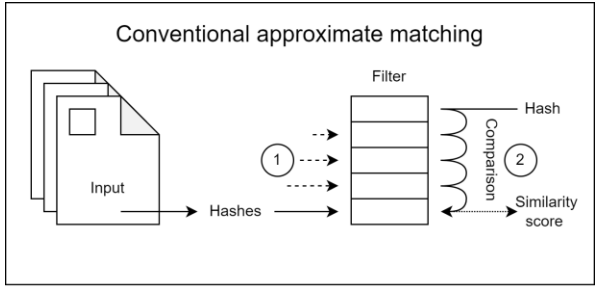
Conclusion

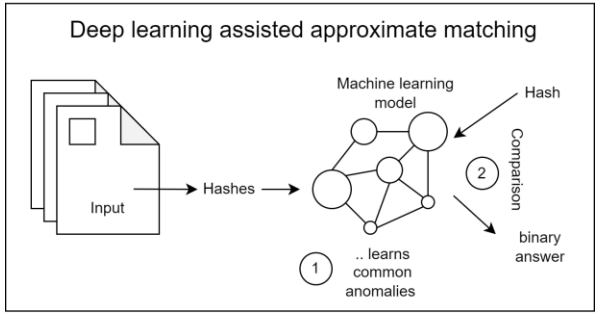
Fuzzy Hashes and their trade-offs



Future research

- **Finding collisions that preserve the information in a file**
- **Deep learning assisted approximate matching:**
  - Hashes are treated like language (NLP)
  - Enables the finding of smaller commonalities
  - **FRASHER** reveals the threats to validity to the machine learning process

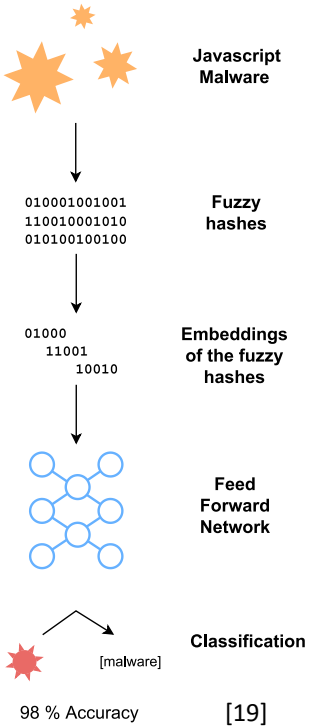




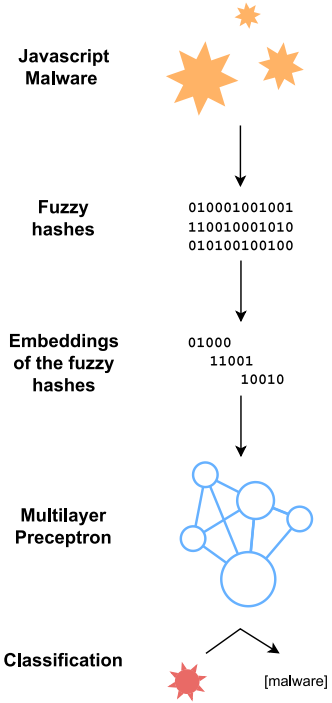
Future Work

Deep learning assisted approximate matching

Current research impressions



[19]



[14]

Microsoft Defender

Bibliography I

[1] Baier, H., & Breitinger, F. (2011). Security aspects of piecewise hashing in computer forensics. In 2011 Sixth International Conference on IT Security Incident Management and IT Forensics (pp. 21–36).

[2] Breitinger, F., & Baier, H. (2011). Performance issues about context-triggered piecewise hashing. In P. Gladyshev, & M. K. Rogers (Eds.), Digital Forensics and Cyber Crime - Third International ICST Conference, ICDF2C 2011, Dublin, Ireland, October 26-28, 2011, Revised Selected Papers (pp. 141–155). Springer volume 88 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. .

[3] Breitinger, F., & Baier, H. (2012a). Properties of a similarity preserving hash function and their realization in sdhash. In 2012 Information Security for South Africa (pp. 1–8).

[4] Breitinger, F., & Baier, H. (2012b). Similarity preserving hashing: Eligible properties and a new algorithm mrsh-v2. In M. K. Rogers, & K. C. Seigfried-Spellar (Eds.), Digital Forensics and Cyber Crime - 4th International Conference, ICDF2C 2012, Lafayette, IN, USA, October 25-26, 2012, Revised Selected Papers (pp. 167–182). Springer volume 114 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.

[5] Breitinger, F., Baier, H., & Beckingham, J. (2012). Security and implementation analysis of the similarity digest sdhash. In First international baltic conference on network security & forensics (nesefo).

[6] Breitinger, F., Guttman, B., McCarrin, M., Roussev, V., White, D. et al. (2014). Approximate matching: definition and terminology. NIST Special Publication, 800 , 10–6028.

[7] Breitinger, F., & Roussev, V. (2014). Automated evaluation of approximate matching algorithms on real data. Digit. Investig., 11 , S10–S17.

[8] Breitinger, F., Stivaktakis, G., & Baier, H. (2013). FRASH: A framework to test algorithms of similarity hashing. Digital Investigation, 10 , S50–S58.

[9] Chang, D., Sanadhya, S. K., Singh, M., & Verma, R. (2015). A collision attack on sdhash similarity hashing. In Proceedings of 10th intl. conference on systematic approaches to digital forensic engineering (pp. 36–46).

Bibliography II

[10] Davies, S. R., Macfarlane, R., & Buchanan, W. J. (2022). Napierone: A modern mixed file data set alternative to govdocs1. Forensic Science International: Digital Investigation, 40 , 301330.

[11] Garfinkel, S., Farrell, P., Roussev, V., & Dinolt, G. (2009). Bringing science to digital forensics with standardized forensic corpora. Digital Investigation, 6 , S2–S11. doi:10.1016/j.diin.2009.06. 016. The Proceedings of the Ninth Annual DFRWS Conference.

[12] Gupta, V., & Breitinger, F. (2015). How cuckoo filter can improve existing approximate matching techniques. In J. I. James, & F. Breitinger (Eds.), Digital Forensics and Cyber Crime (pp. 39–52). Cham: Springer International Publishing.

[13] Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. Digital Investigation, 3 , 9197. doi:10.1016/j.diin.2006.06.015. The Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS ’06).

[14] Lazo, E. G. (2021). Combing through the fuzz: Using fuzzy hashing and deep learning to counter malware detection evasion techniques. Microsoft 365 Defender Research Team.

[15] Lee, A., & Atkison, T. (2017). A comparison of fuzzy hashes: Evaluation, guidelines, and future suggestions. In Proceedings of the SouthEast Conference ACM SE ’17 (p. 18–25). New York, NY, USA: Association for Computing Machinery.

[16] Martın-Perez, M., Rodríguez, R. J., & Breitinger, F. (2021). Bringing order to approximate matching: Classification and attacks on similarity digest algorithms. Forensic Science International: Digital Investigation, 36 , 301120.

[17] Martinez, V. G.,´Alvarez, F. H., & Encinas, L. H. (2014). State of the art in similarity preserving hashing functions. In 2014 International Conference on Security and Management (SAM’14), Worldcomp 2014 (pp. 139–145).

Bibliography III

[18] Oliver, J., Forman, S., & Cheng, C. (2014). Using randomization to attack similarity digests. In L. Batten, G. Li, W. Niu, & M. Warren (Eds.), Applications and Techniques in Information Security (pp. 199–210). Berlin, Heidelberg: Springer Berlin Heidelberg.

[19] Peiser, S. C., Friborg, L., & Scandariato, R. (2020). Javascript malware detection using locality sensitive hashing. In M. H˙olbl, K. Rannenberg, & T. Welzer (Eds.), ICT Systems Security and Privacy Protection (pp. 143–154). Cham: Springer International Publishing.

[20] Ribeiro, B., Moia, V. H. G., & Henriques, M. A. A. (2017). Similarity digest search: A survey and comparative analysis of strategies to perform known file filtering using approximate matching. Security and Communication Networks, 2017 , 1306802.

[21] Roussev, V. (2010). Data fingerprinting with similarity digests. In K. Chow, & S. Shenoi (Eds.), Advances in Digital Forensics VI Sixth IFIP WG 11.9 International Conference on Digital Forensics, Hong Kong, China, January 4-6, 2010, Revised Selected Papers (pp. 207–226). Springer volume 337 of IFIP Advances in Information and Communication Technology.

[22] Roussev, V. (2011). An evaluation of forensic similarity hashes. Digital Investigation, 8 , S34–S41.

[23] Roussev, V., Richard, G. G., & Marziale, L. (2007). Multi-resolution similarity hashing. Digital Investigation, 4 , 105–113.

[24] Singh, M. (2021). Essential characteristics of approximate matching algorithms: A survey of practitioners opinions and requirement regarding approximate matching.

[25] Singh, M., Khunteta, A., Ghosh, M., Chang, D., & Sanadhya, S. K. (2022). FbHash-E: A time and memory efficient version of fbhash similarity hashing algorithm. Forensic Science International: Digital Investigation, 41 , 301375

Thank you.

Contact us:

thomas.goebel@unibw.de

frieder.uhlig@stud.tu-darmstadt.de

harald.baier@unibw.de

frank.breitinger@unil.ch

FRASHER is available via GitHub:
https://github.com/warlmare/FRASHER