An Evaluation of Forensic Similarity Hashes

By

Vassil Roussev

Presented At

The Digital Forensic Research Conference

DFRWS 2011 USA New Orleans, LA (Aug 1st - 3rd)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

http:/dfrws.org

Digital Forensic Research Conference
Aug 1-3, 2011 ◦ New Orleans, LA

An Evaluation of Forensic Similarity Hashes

Vassil Roussev
vassil@cs.uno.edu

Agenda

- Intro
  - Motivation, problems, goals, requirements, …
- High-level tool design
- Evaluation studies
- Current/planned ***sdhash*** infrastructure
- Quick demo (time permitting)
- Q & A

Motivation: Crypto Hash Filtering is Failing

- *Known file filtering:*
  - Crypto-hash known files, store in library (e.g. NSRL)
  - Hash files on target
  - Filter in/out depending on interest
- Challenges
  - Static libraries are falling behind
    - Dynamic software updates, trivial artifact transformations

    ➔ We need **version** correlation
  - Need to find embedded objects
    - Block/file in file/volume/network trace
  - Need higher-level correlations
    - Disk-to-RAM
    - Disk-to-network

Similarity Hash Requirements/Scenarios

- Identification of embedded/trace evidence
  - "Needle in a haystack"
- Identification of code versions
  - File-to-file correlation
- Identification of related documents
  - File-to-file correlation
- Correlation of RAM and disk sources
  - Different representation of same objects
- Correlation of network and disk sources
  - Fragmentation/alignment issues
  - **No** flow reconstruction

Existing Similarity Hashing: *ssdeep*

- "Context-triggered piecewise hashing"
  - Developed by Jesse Kornblum (2006)
  - An adaptation of an early spam filtering algorithm
- General idea
  - Break up the file into chunks
  - Generate a 6-bit hash for each chunk
  - Concatenate the hashes to obtain the file signature:

```
24576:fBovHm8YnR/tDn7uSt8P8SRLAD/5Qvhfpt8P8SRLm:mvHKnx5C868MAD/5uz68Mm,"file.pdf"
```

  - Treat the signatures as strings; use edit distance to estimate similarity

ssdeep: Problems

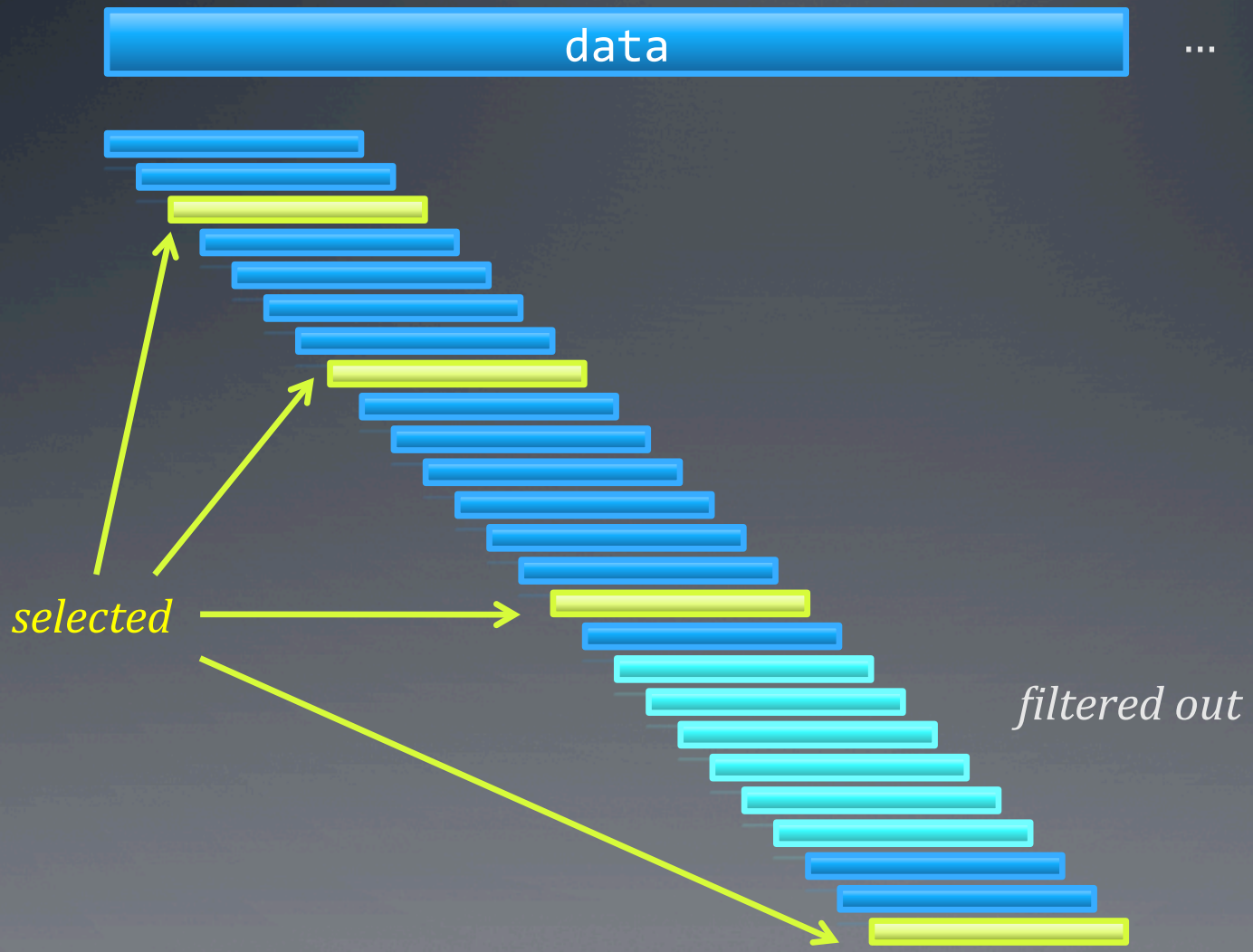
- Methodology (random polynomial fingerprinting)
  - Works ok on mid-/high-entropy data
    - Text/compressed data
  - Degenerates on lower-entropy data
    - Uneven coverage
    - Many false positives
  - Difficult to fix
- Design
  - Fixed-size signature (does not scale)
  - Distance metric choice (edit distance) is questionable

➔ Fixes essentially require a new tool

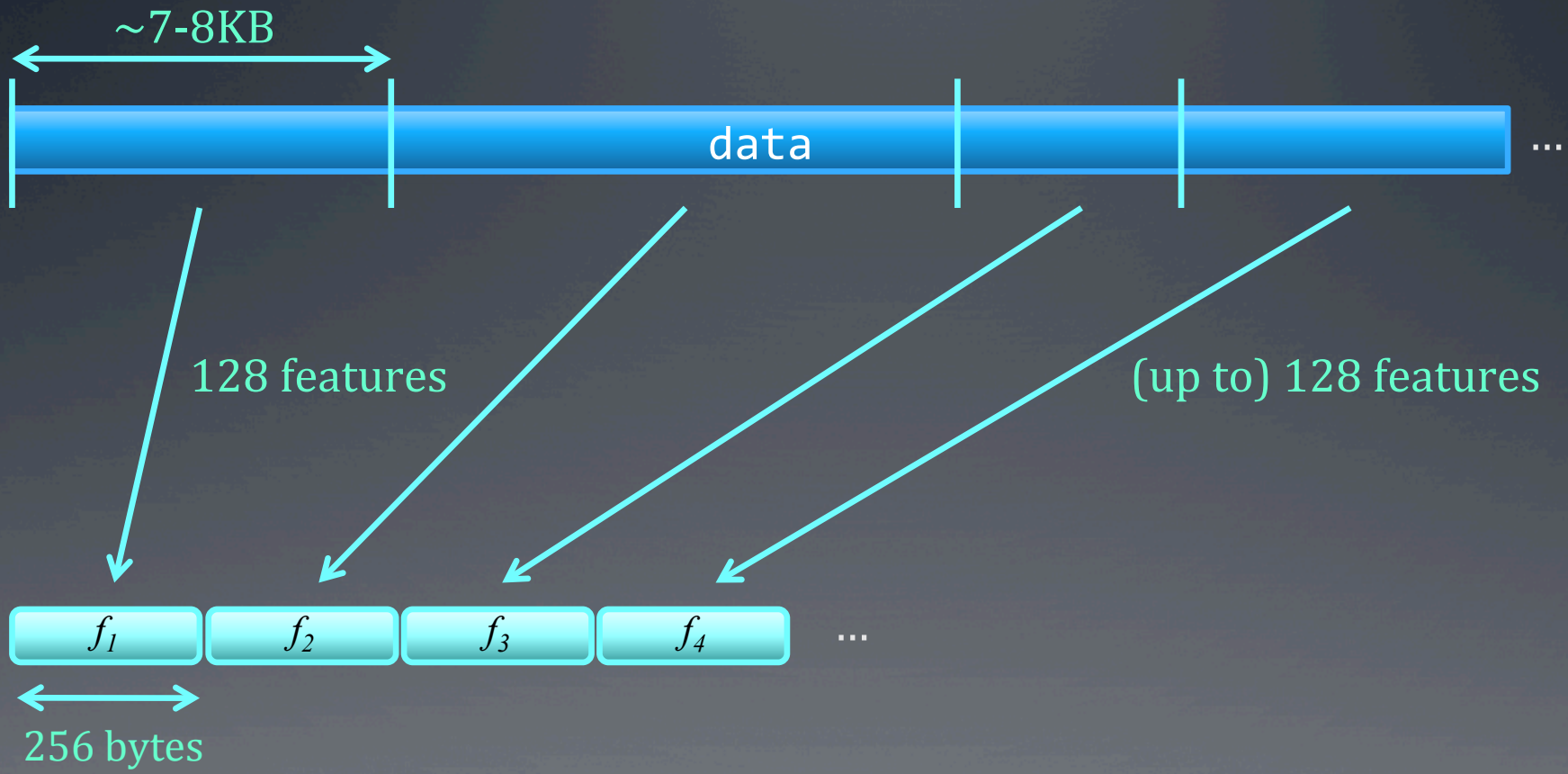
sdhash: *Similarity Digests*

- Terminology:
  - Feature: *a 64-byte sequence*
    - (*other variations are possible*)
- Idea:
  - Consider all features:
    - Compute rolling entropy measure
  - Filter out low-entropy/extreme high entropy ones
  - From each neighborhood, pick the rarest ones
    - Based on entropy score and empirical observations
  - Hash selected features and put into a *Bloom* filter
    - *Bloom filter* == probabilistic, compressed set representation
  - Create more filters as necessary
  - Signature is a sequence of *Bloom* filters

Feature Selection

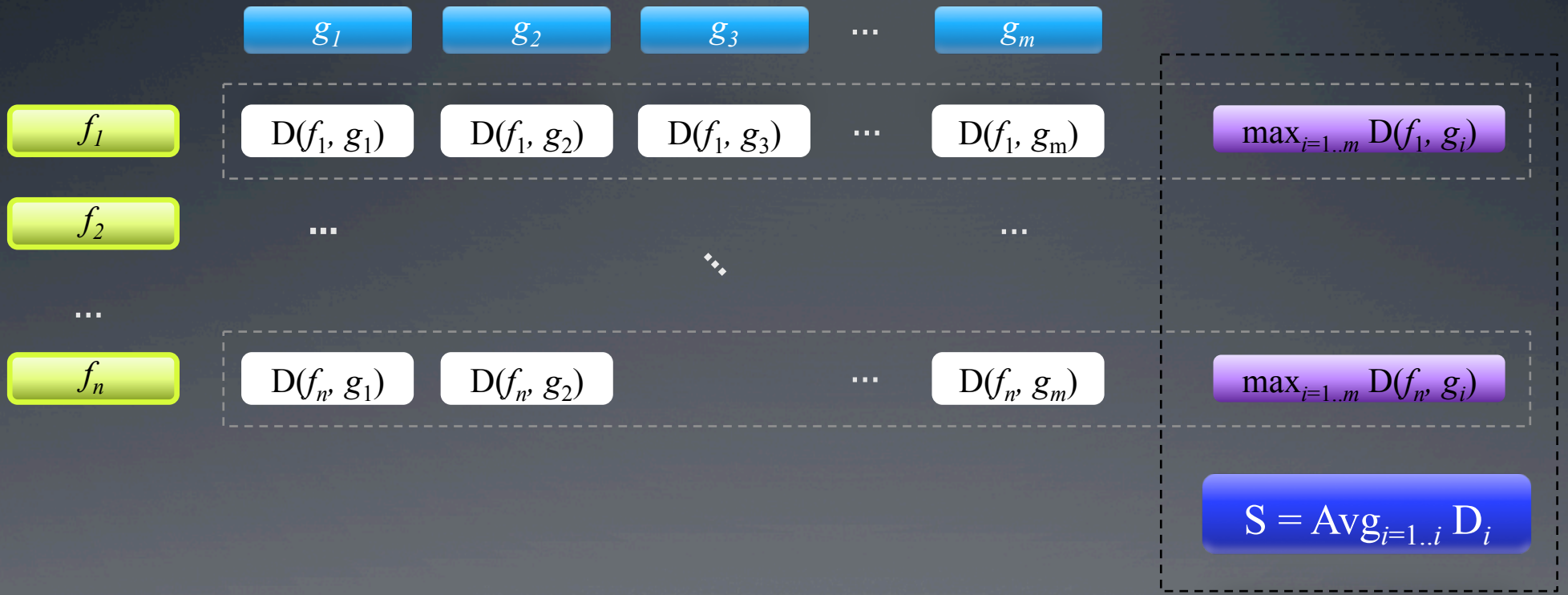


Similarity Digest Signature



- *On average*, a 256-byte filter represents 7-8KB chunk of the original data.
- Digest size is ~3% of original data (could be smaller).
- (No original data is stored.)

Similarity Digest Comparison



ssdeep vs. *sdhash*
Round 1: *Controlled Study*

- Controlled study
  - All targets generated using random data
  - Allows for precise control of common data
  - Provides a baseline for the tools' capabilities
  - Best case scenario
- Scenarios
  - Embedded object detection
  - Single-common-block file correlation
  - Multiple-common-blocks file correlation

Embedded Object Detection

- Scenario implementation
  - Generate target & object
  - Place object randomly in target
  - Run tools on <object, target>
  - Do 1,000 runs changing target, object, and placement
- Evaluation criterion
  - Given: target of fixed size
  - Q: What is the smallest embedded object that can be ***reliably detected***?
    - Reliable detection == 95%$^{+}$ successful correlations

Min Embedded Block Correlation (KB)

(smaller **is better**)

Object Size (KB)	**Max** Target Size (KB): 95% Detection	
	ssdeep	**sdhash***
64	192	65,536
128	384	131,072
256	768	262,144
512	1,536	524,288
1,024	3,072	1,048,576

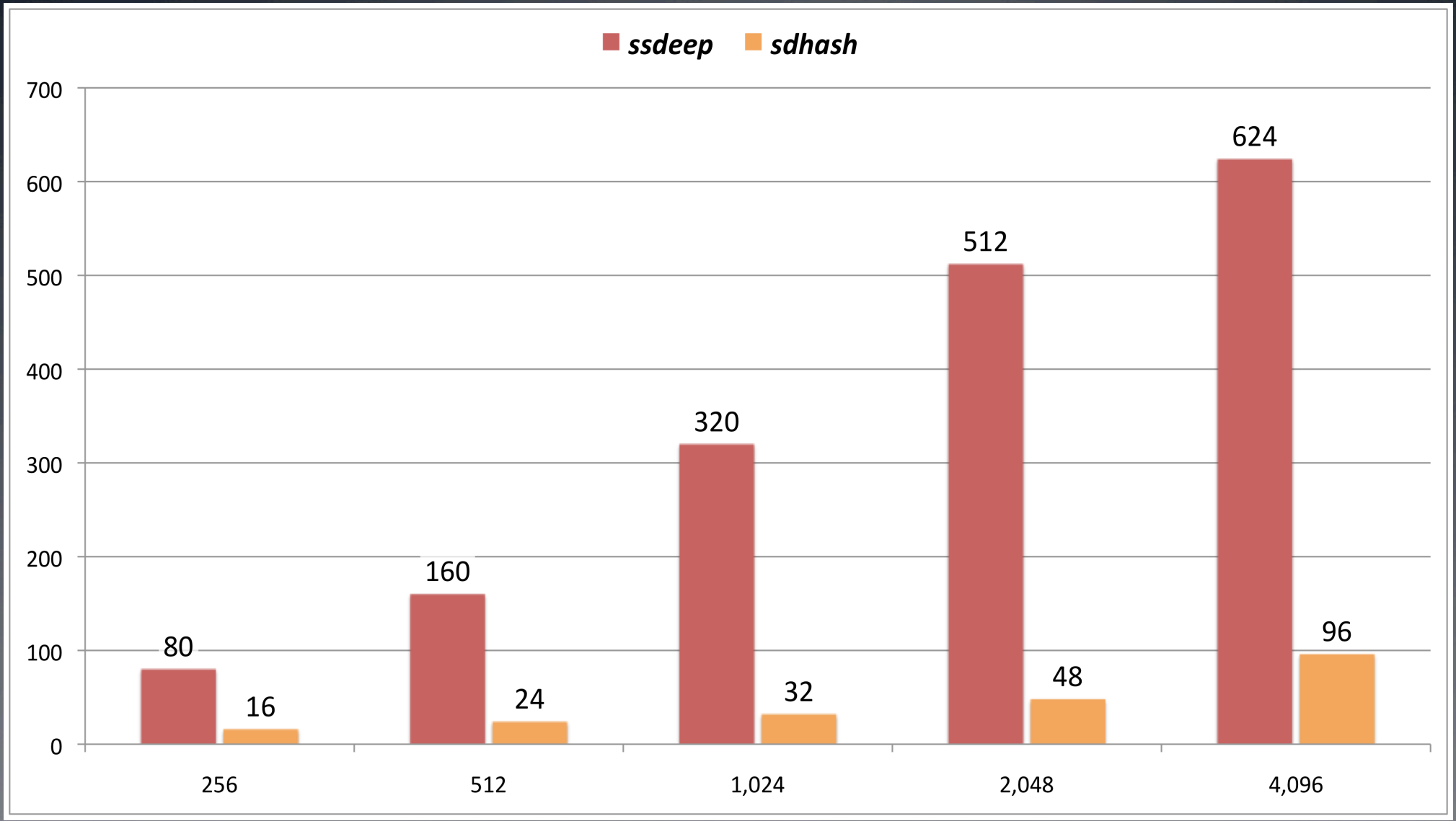
* max values tested

Single-Common-Block Correlation

- Scenario implementation
  - Generate targets & object
  - Place object randomly in both target
  - Run tools on <T_1, T_2>
  - Do 1,000 runs changing target, object, and placement
- Evaluation criterion
  - Given: targets of fixed size
  - Q: What is the smallest embedded object that can be ***reliably detected***?
    - Reliable detection == 95%$^+$ successful correlations

Min Common Block Correlation (KB)

(smaller **is better**)



Multiple-Common-Blocks Correlation

- Scenario implementation
  - Generate targets & object; split object in 4/8 pieces
  - Place pieces randomly in both target
  - Run tools on <T_1, T_2>
  - Do 1,000 runs changing target, object, and placement
- Evaluation criterion
  - Given: targets of fixed size, object size = ½ target size
  - Q: What is the probability that a tool will detect it?

Multiple Common Block Correlation (Fraction)
(BIGGER is better)

Targets (KB)	Common (KB)	4 pieces		8 pieces	
		ssdeep	**sdhash**	**ssdeep**	**sdhash**
256	128	0.35	1.00	0.04	1.00
512	256	0.48	1.00	0.04	1.00
1,024	512	0.39	1.00	0.07	1.00
2,048	1,024	0.59	1.00	0.17	1.00
4,096	2,048	0.96	1.00	0.54	1.00

ssdeep vs. *sdhash*
Round 2: *Real Data Study*

- Real files from the ***NPS GovDocs1*** corpus
  - Fundamentally, a user study
- Q: How does byte-level correlation map to human-perceived artifact correlation?
  - Not all commonality is reflected at the semantic level
- Related files defined:
  - Versions of the same file
  - Shared format/content (e.g. web layout, JPEG)
  - "Flash" evaluation: similarity obvious within 30sec

Real Data Study

doc	gif	html	jpg	pdf	ppt	text	xls
533	67	1093	362	1073	368	711	250

- The *T5* set
  - GovDocs1 sample: 000-004
  - 4,557 files, 1.8GB total
  - 4KB-16.4MB
- Evaluation
  - For all unique pairs (~10 mln.)
    - Run ***ssdeep***
    - Run ***sdhash***
    - Evaluate positive results manually

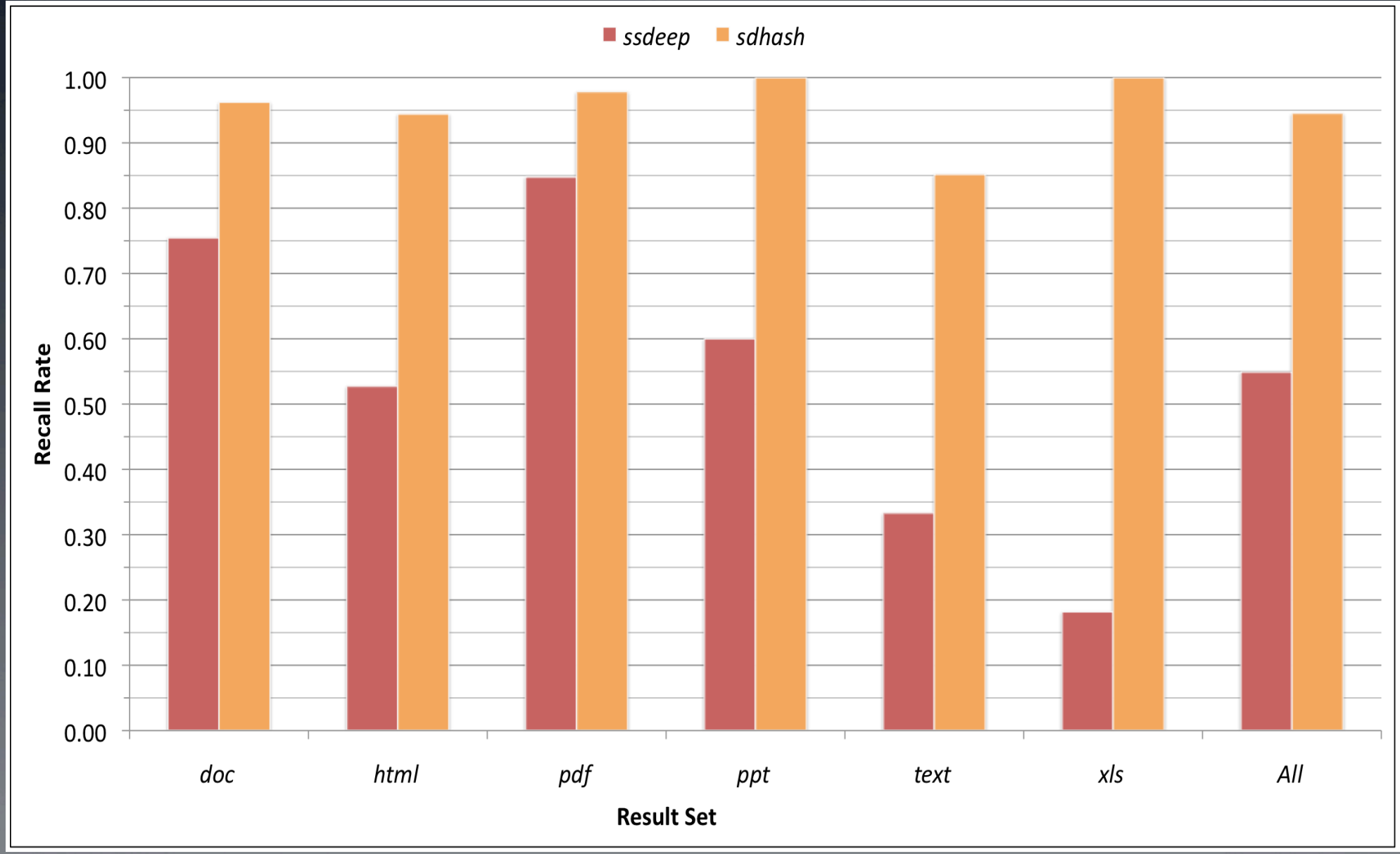
Evaluation Statistics

Total correlations examined	1,699
Total unique files examined	820
True positives	1,203
Common true positives	588
False positives	496
Common false positives	65

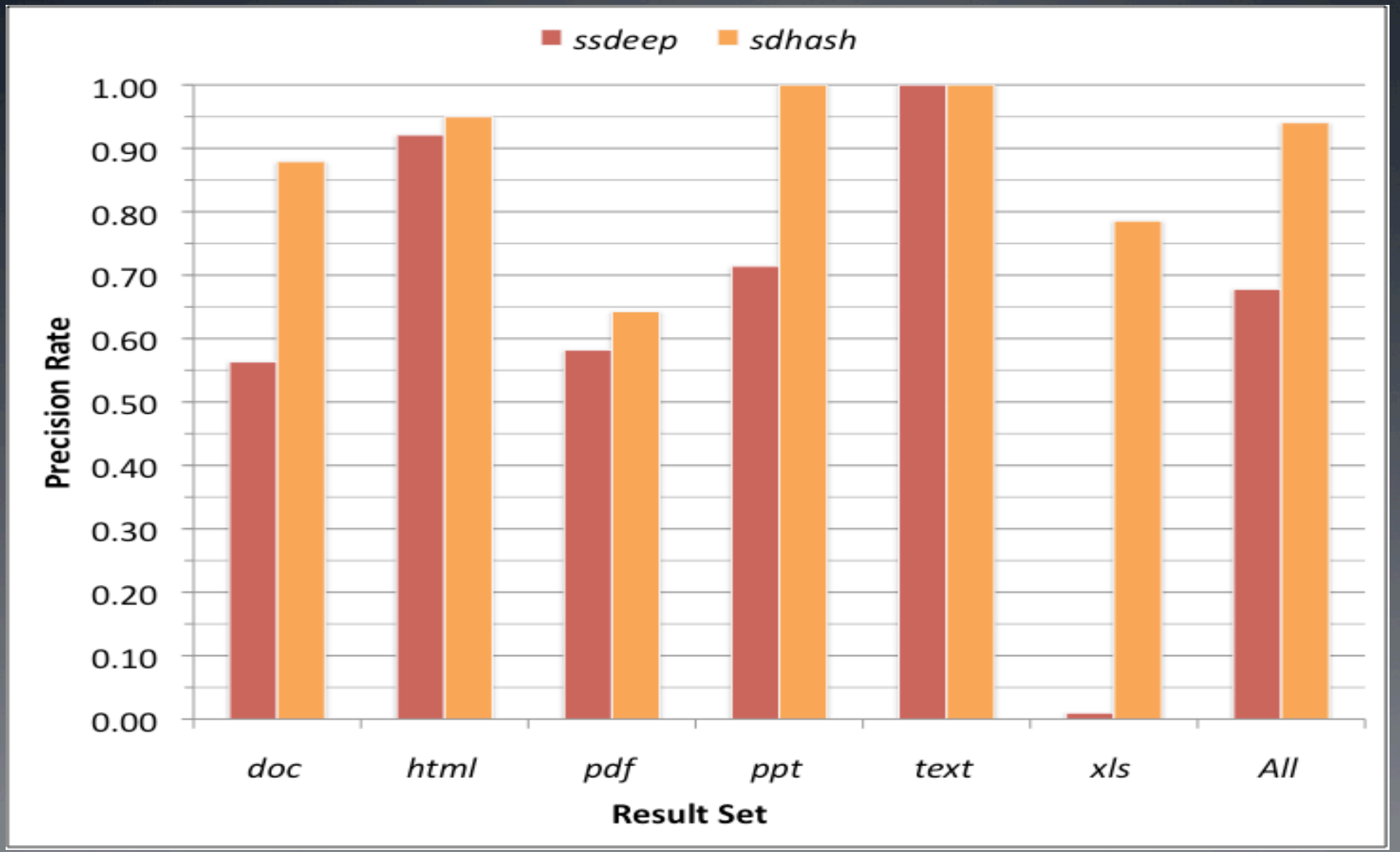
The Raw Numbers

Set	*ssdeep*		*sdhash*		Total
	TP	FP	TP	FP	
doc	40	31	51	7	53
html	550	47	985	52	1043
jpg	0	23	0	2	0
pdf	39	28	45	25	46
ppt	15	6	25	0	25
text	9	0	23	0	27
xls	2	199	11	3	11
Mixed	*2*	*24*	*16*	*18*	~~*58*~~ *16*
All	**653**	**310**	**1124**	**71**	**1189**

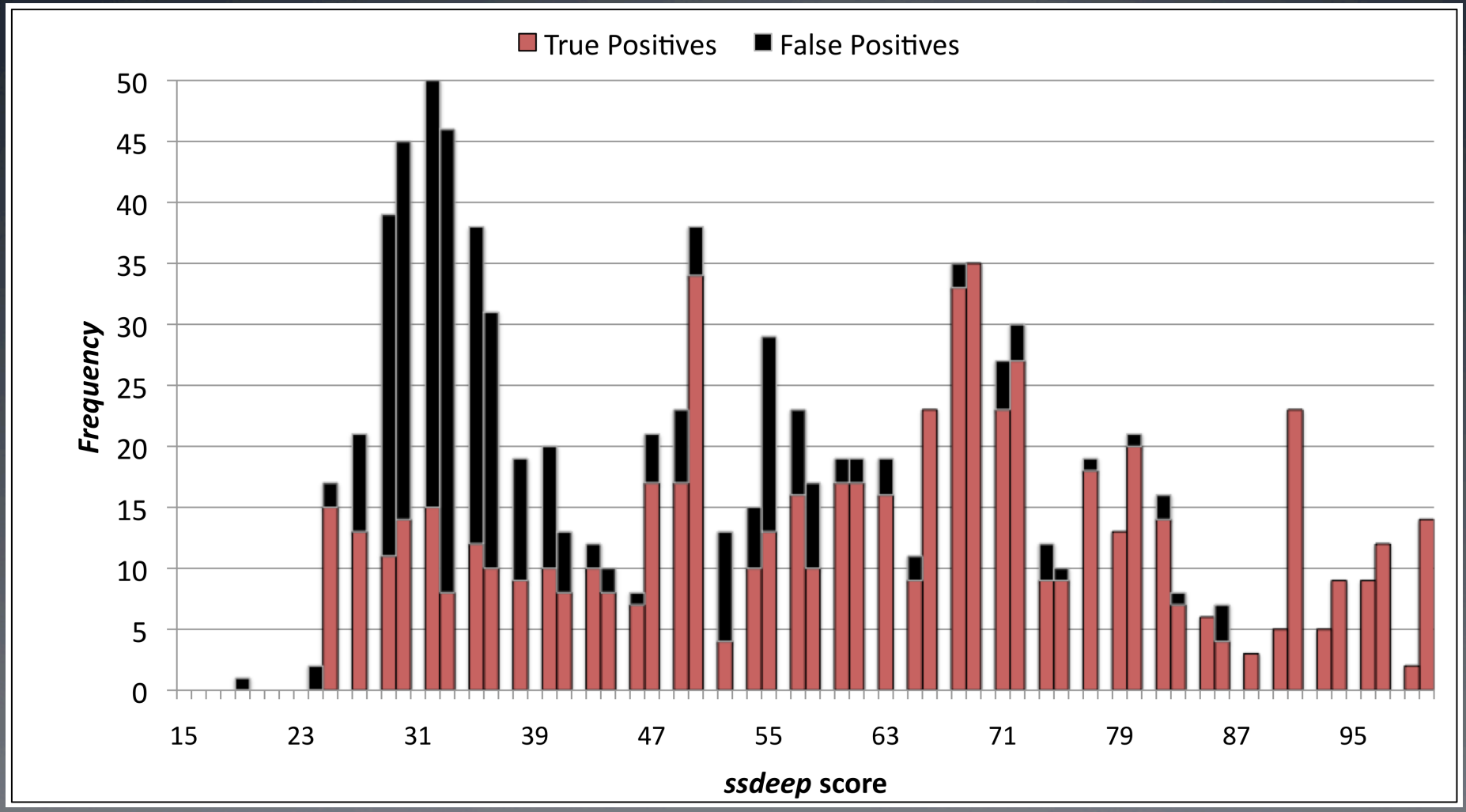
Recall Rates: TP/Total



Precision Rates: TP/(TP+FP)

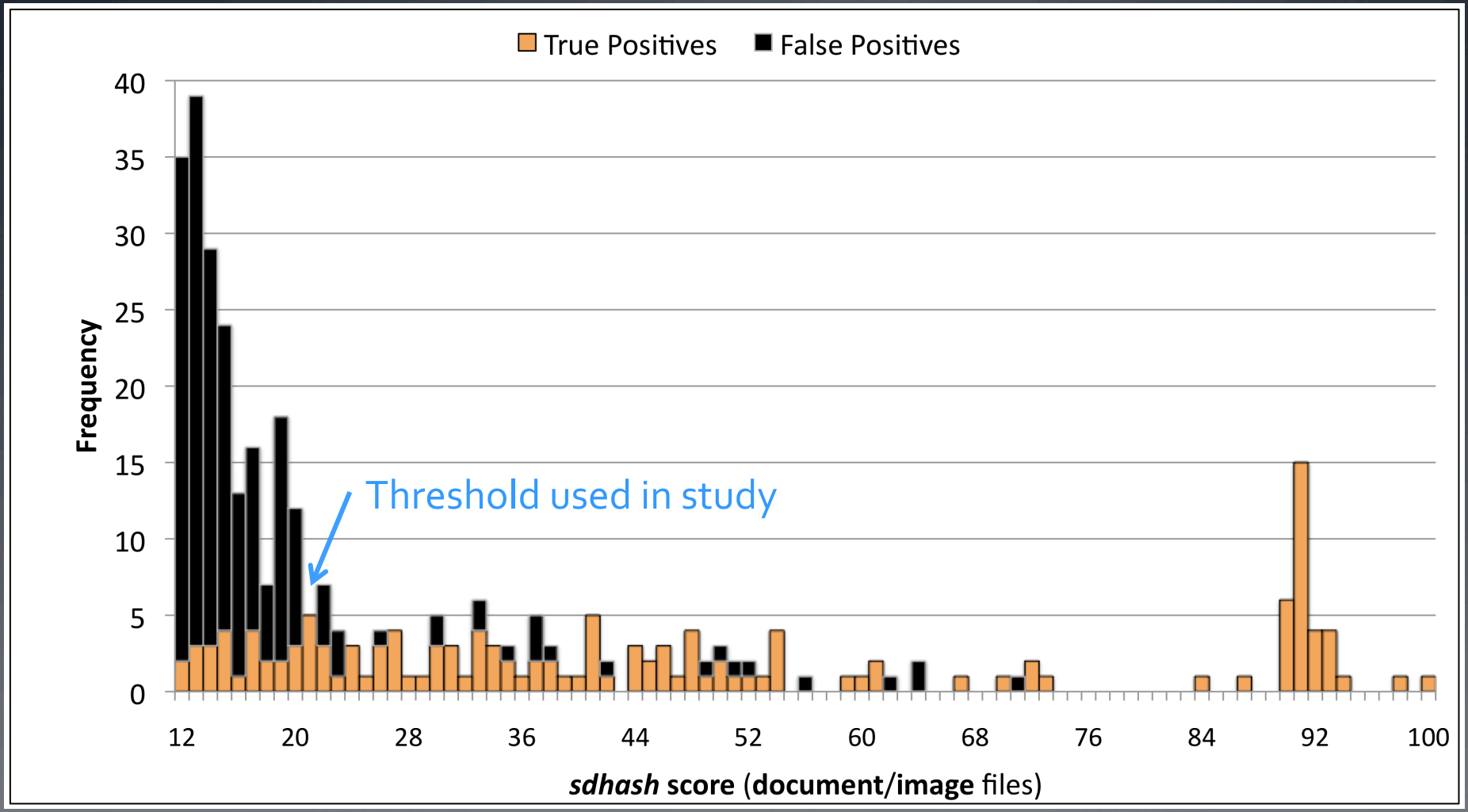


ssdeep: FP & TP substantially scores overlap



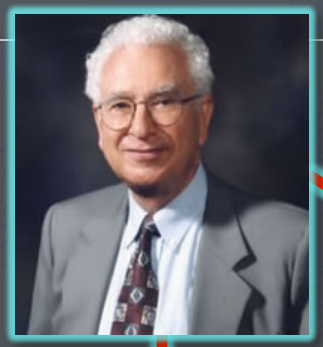
- Cannot use thresholds for ROC trade off

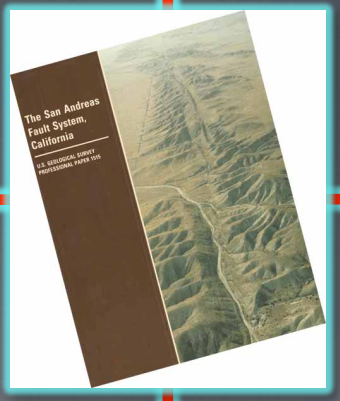
sdhash: FP & TP scores are separable

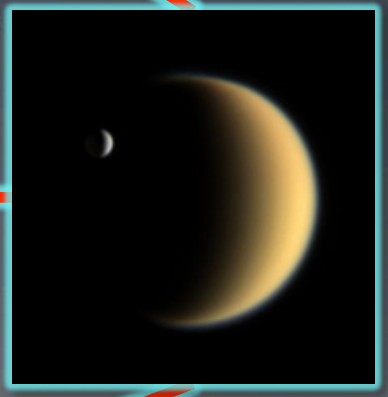


- Thresholding is effective in cheaply eliminating FPs

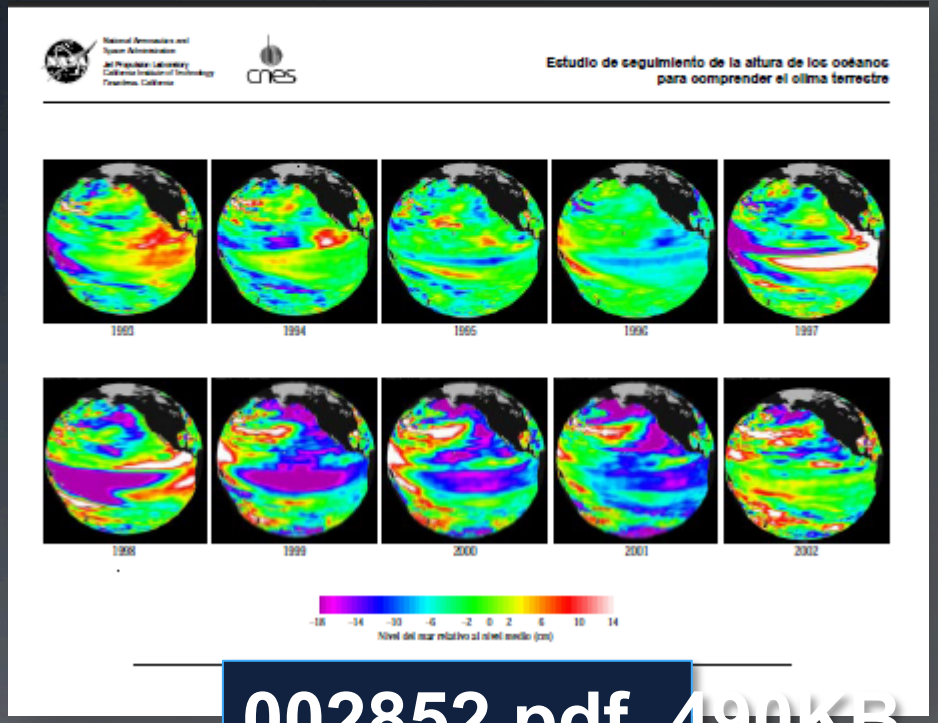
Example *ssdeep* false positives
(score: 54-86)







Example *sdhash* false positive (score: 63)



002852.pdf 490KB

003189.pdf 660KB

- ***NOT*** really a false positive!
  - The two files share a of 381 KB (73% of 2852.pdf):

```
… /DeviceCMYK /Length 389758
/Filter /FlateDecode >> …
```

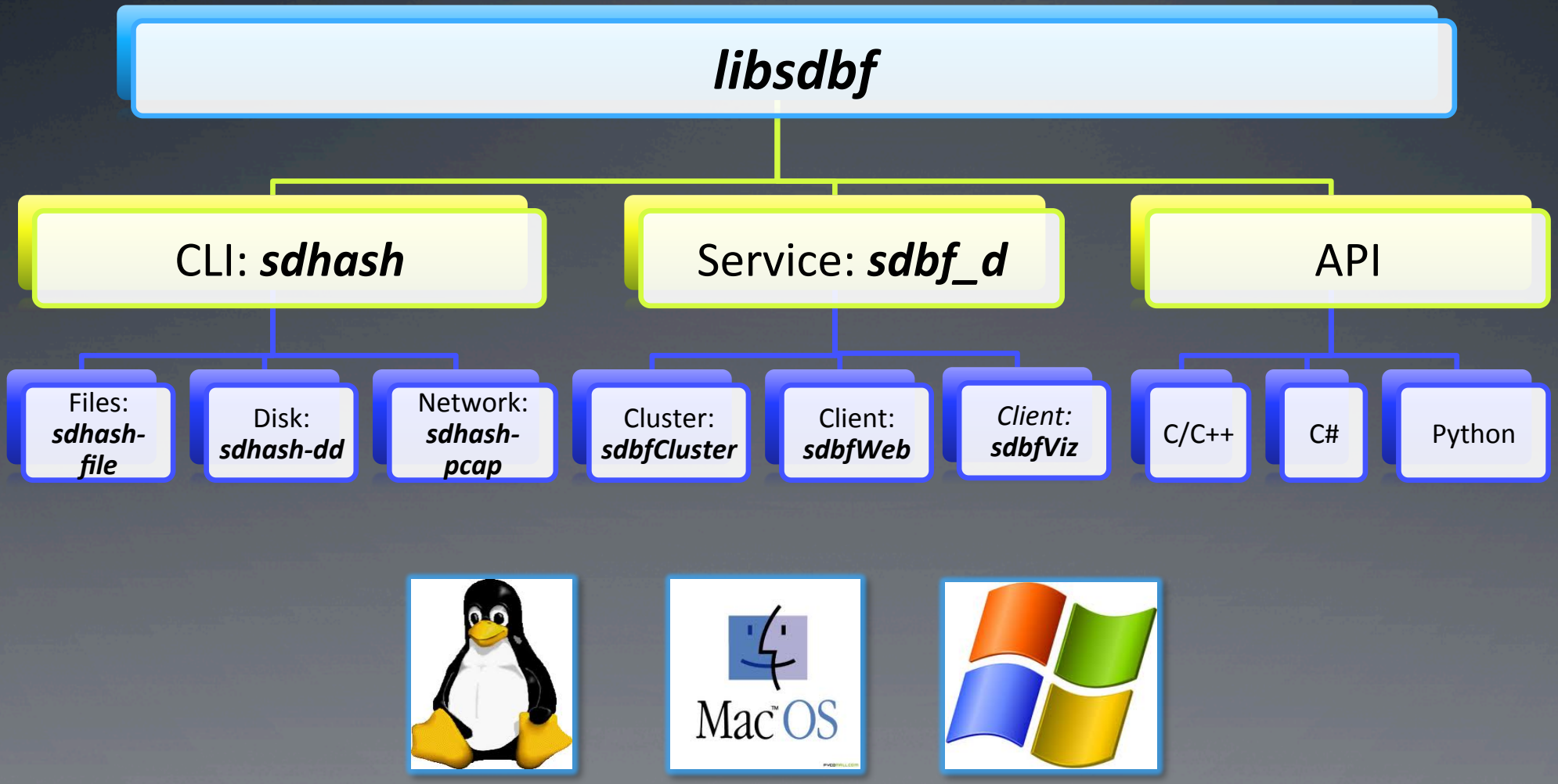
Evaluation Summary

- New hashing scheme based on ***similarity digests***
  - Scalable, robust, parallelizable
  - Evaluated under controlled & realistic conditions
  - Outperforms existing work by a wide margin
    - Recall: 95% vs. 55%
    - Precision: 94% vs. 68%
  - Graceful behavior at the margin
    - Intuitive behavior of the similarity score
    - Scores drop gradually as detection limits are approached
  - Meets at least three requirements
    - More evaluation needed for disk/network & disk/RAM

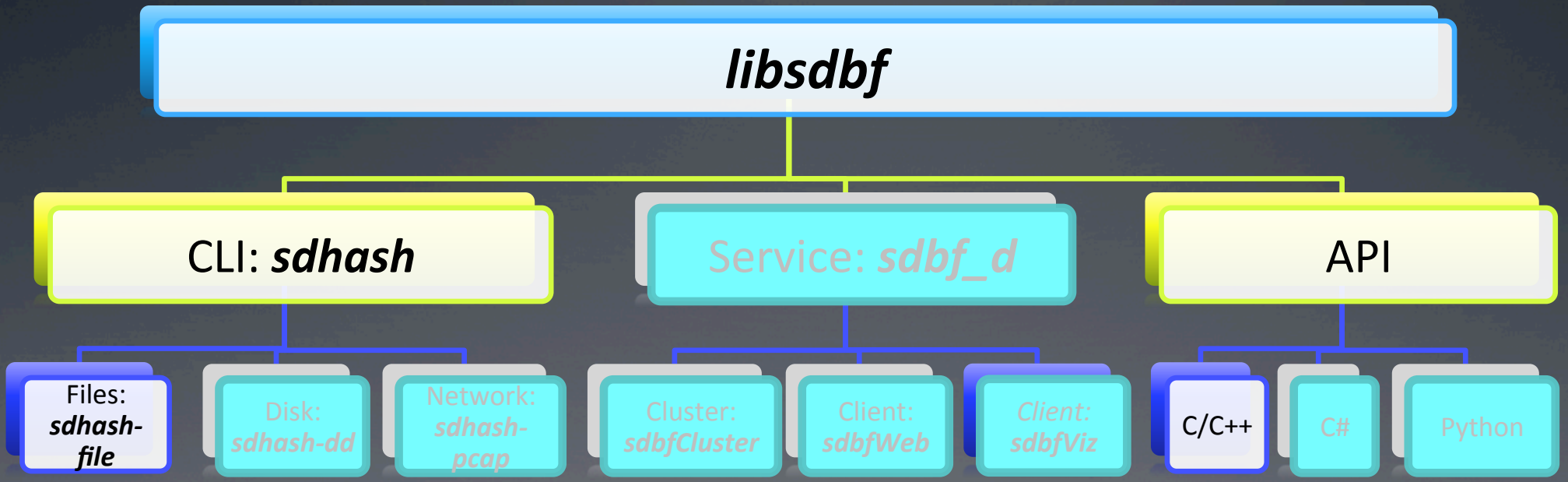
Current Throughput (ver 1.3)

- Hash generation rate
  - Six-core Intel Xeon X5670 @ 2.93GHz
    ~27MB/s per core
  - Quad-Core Intel Xeon @ 2.8 GHz
    ~20MB/s per core
- Hash comparison
  - 1MB vs. 1MB: 0.5ms
- T5 corpus (4,457 files, all pairs)
  - 10mln file comparisons in ~ 15min
    - 667K file comps per second
    - Single core

The Envisioned Architecture



The Current State



Todo List (1)

- ***libsdbf***
  - Ver 2.0 rewrite
  - Full parallelization (TBB?)
  - Compression (?)
- ***sdhash-file***
  - More command line options/compatibility w/ssdeep
  - Parallel processing
  - Service-based processing (w/ ***sdbf_d***)
- ***sdhash-pcap***
  - Pcap-aware processing:
    - payload extraction, file discovery, timelining

Todo List (2)

- ***sdhash-dd***
  - Block-aware processing, compression
- ***sdbf_d***
  - Persistance: XML
  - Service interface: JSON
  - Server clustering
- ***sdbfWeb***
  - Browser-based management/query
- ***sdbfViz***
  - Large-scale visualization & clustering

Further Development

- Integration w/ RDS
  - ***sdhash-set***: construct *SDBF*s from existing SHA1 sets
    - Compare/identify whole folders, distributions, etc.
- Structural feature selection
  - E.g., exe/dll, pdf, zip, …
- Optimizations
  - Sampling
  - Skipping
    - Under **min** continuous block assumption
  - Cluster "core" extraction/comparison
  - GPU acceleration
- Representation
  - Multi-resolution digests
  - New crypto hashes
  - Data offsets

Thank you!

- http://roussev.net/sdhash
  - wget http://roussev.net/sdhash/sdhash-1.3.zip
  - make
  - ./sdhash
- References
  - V. Roussev, "Data Fingerprinting with Similarity Digests", in K.-P. Chow, S. Shenoi (Eds.): Advances in Digital Forensics VI, IFIP AICT 337, pp. 207-225, 2010
  - V. Roussev, "An Evaluation of Forensic Similarity Hashes", in DFRWS 2011
- Contact:

Vassil Roussev
vassil@roussev.net

- Q & A