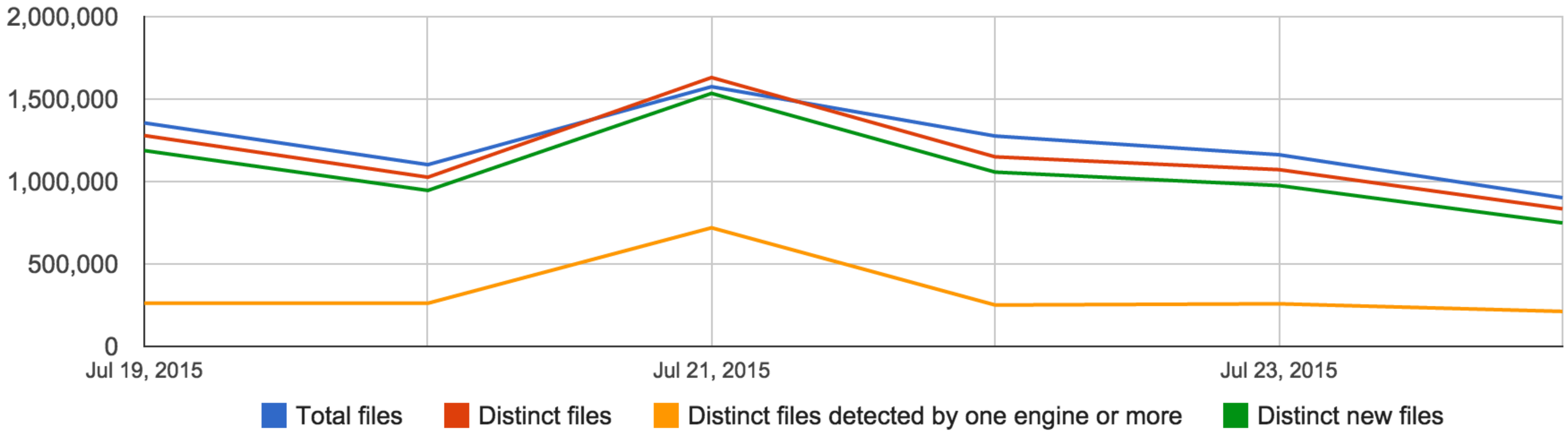
Experimental Study of Fuzzy Hashing in Malware Clustering Analysis

Yuping Li[1], Sathya Chandran Sundaramurthy[1]
Alexandru G. Bardas[1], Xinming Ou[1]
Doina Caragea[1], Xin Hu[2], Jiyong Jang[2]
1) Kansas State University 2) IBM Research

Motivation

- Huge volume of malware samples:

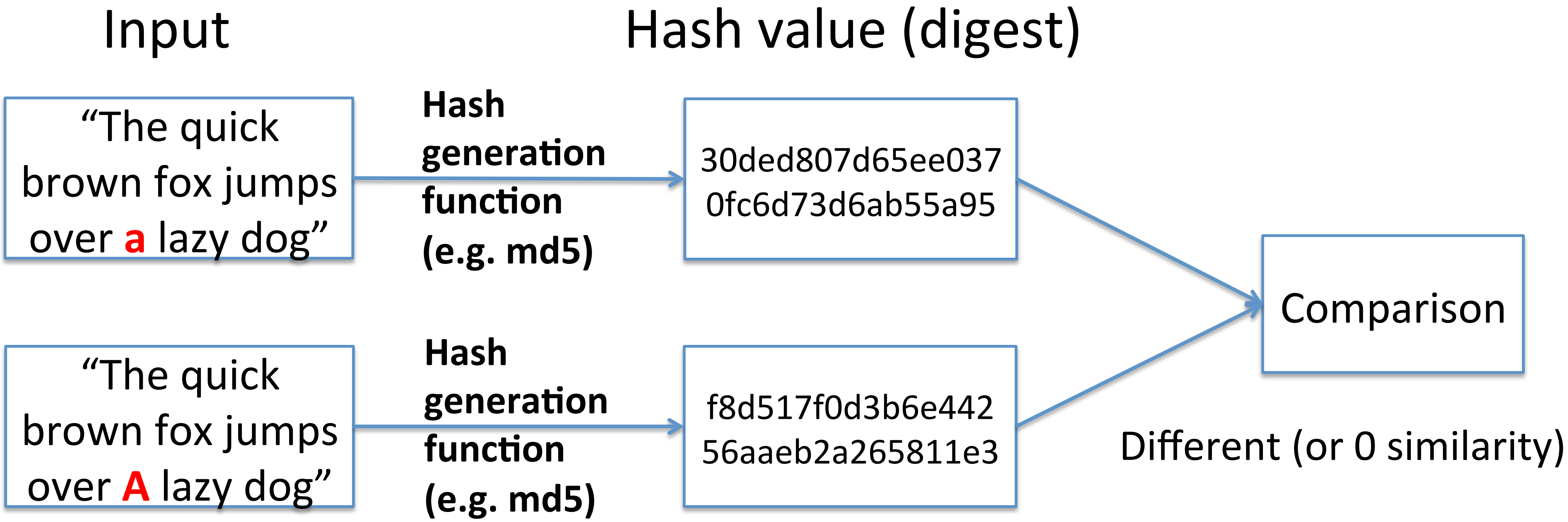


Source: Malware submission statistics from VirusTotal (Jul 19, 2015 – Jul 24, 2015)

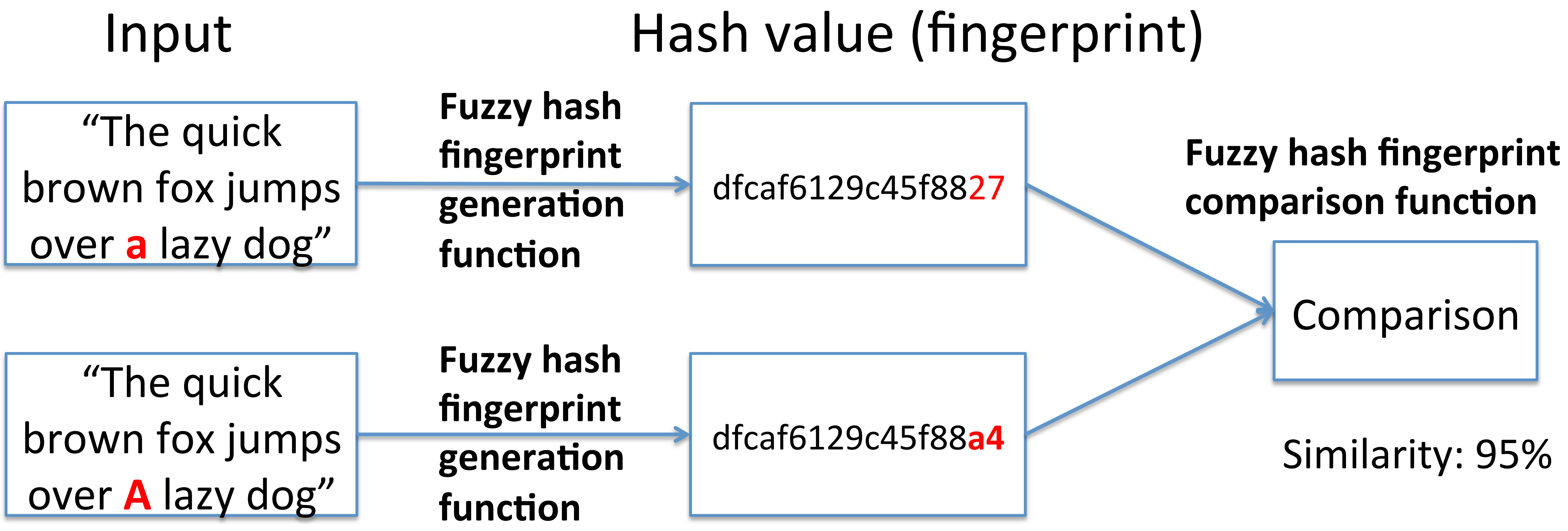
Why to use fuzzy hashing?

- Use fuzzy hashing to identify the new specie - “cat” for prioritized analysis

Cryptographic Hash Overview



Fuzzy Hash Overview



Previous Fuzzy Hashing Applications

- Using fuzzy hashing techniques to identify malicious code (ShadowServer, 2007)
- VirusTotal incorporated SSDeep hashes into their data set (VirusTotal, 2012)
- Fuzzy hashing techniques in applied malware analysis (CMU CERT, 2011)

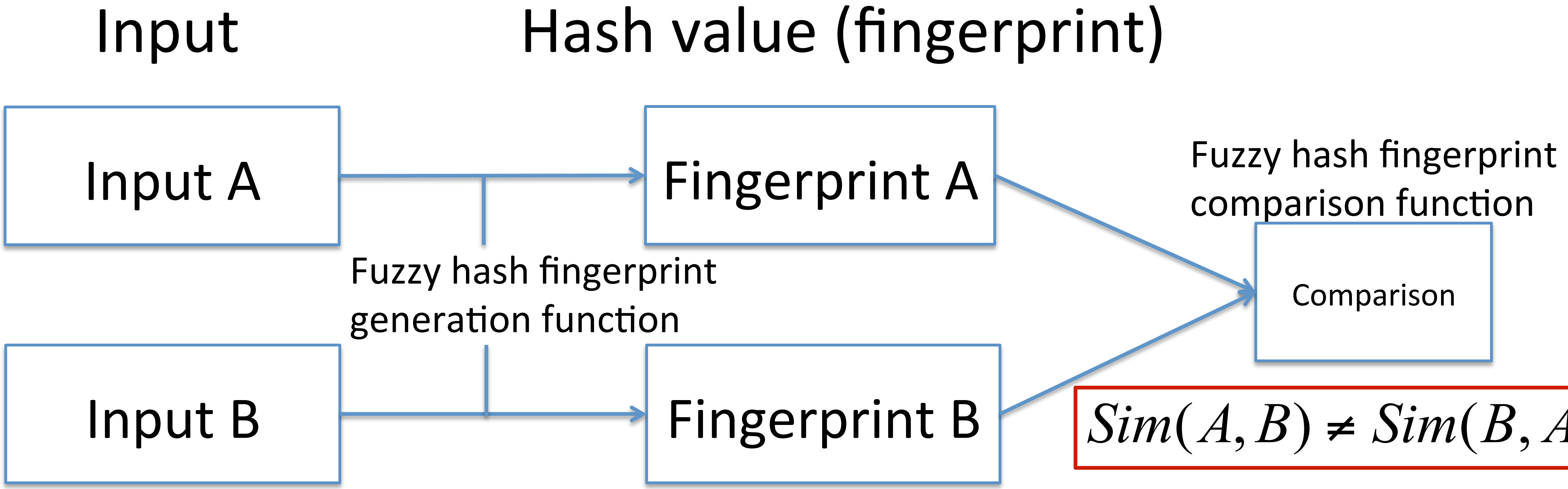
Our Contributions

- We identify two design flaws within some of existing fuzzy hashing algorithms.

- We design and implement a generic experimentation framework for evaluating the performances of different fuzzy hash functions.

- We show that current fuzzy hashing algorithms can be further improved and proposed a new fuzzy hash function.

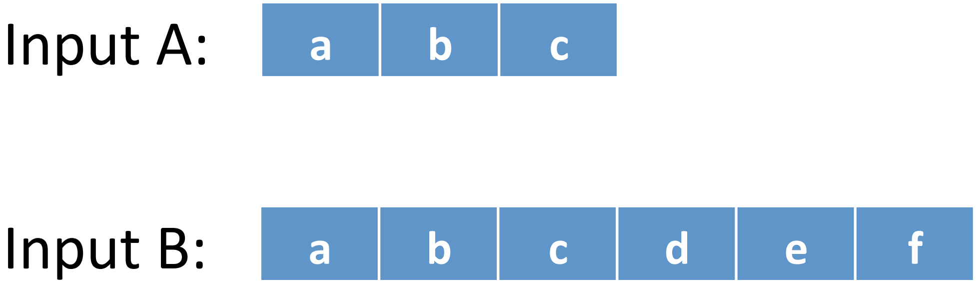
Design Flaw 1: Asymmetric Distance Computation



$$Sim(A,B) \neq Sim(B,A)$$

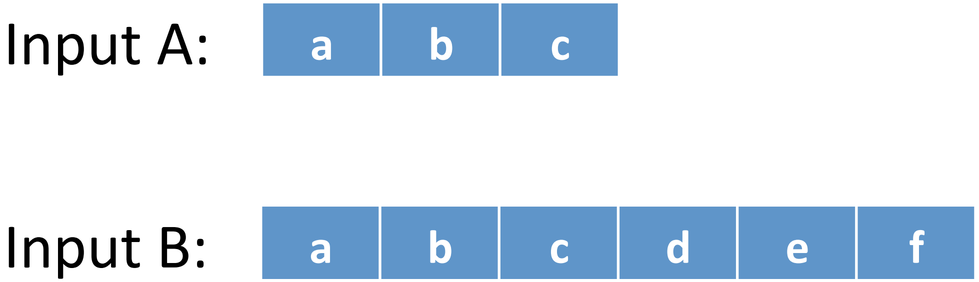
Design Flaw 2: Incompatible Interpretations

- Containment analysis, expected optimal score: **1.0**



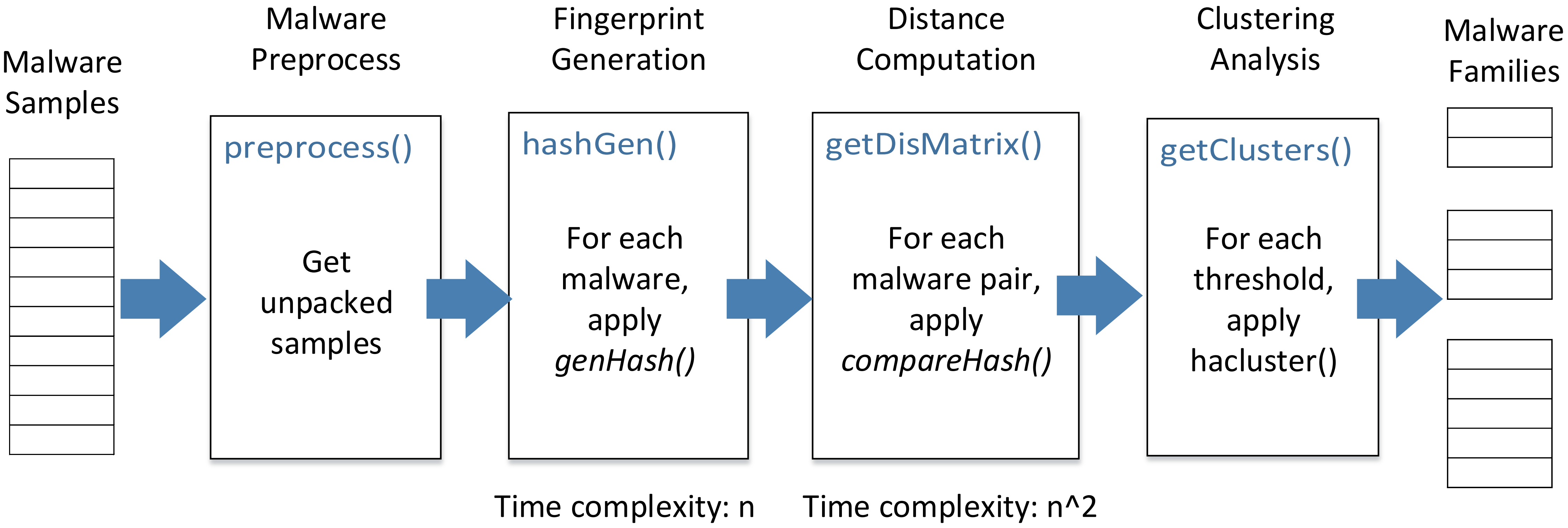
- Explanation: A is **100%** contained in B

- Similarity analysis, expected optimal score: **0.5**



- Explanation: A and B have an overall similarity of **50%**

Generic Experimentation Framework For Fuzzy Hash Evaluation



Malware Data Preparation

- Preparing malware dataset that are reliable:

  1) Prepare unpacked samples

  - Classification of packed and unpacked samples

  2) Prepare samples with accurate family name

  - Majority vote of VirusTotal labels

Dataset Statistics (from VirusShare)

Family	Size	Family	Size
Viking	31	Vilsel	185
Fesber	57	Jeefo	36
Neshta	39	Turkojan	22
Skintrim	41	Bettersurf	300
Ramnit	38	Koutodoor	30
Zenosearch	99	Zbot	22
Hupigon	28	Fosniw	22
Domaiq	147	Wabot	27
Xpaj	22	Total	**1146**

Clustering Accuracy Measurement

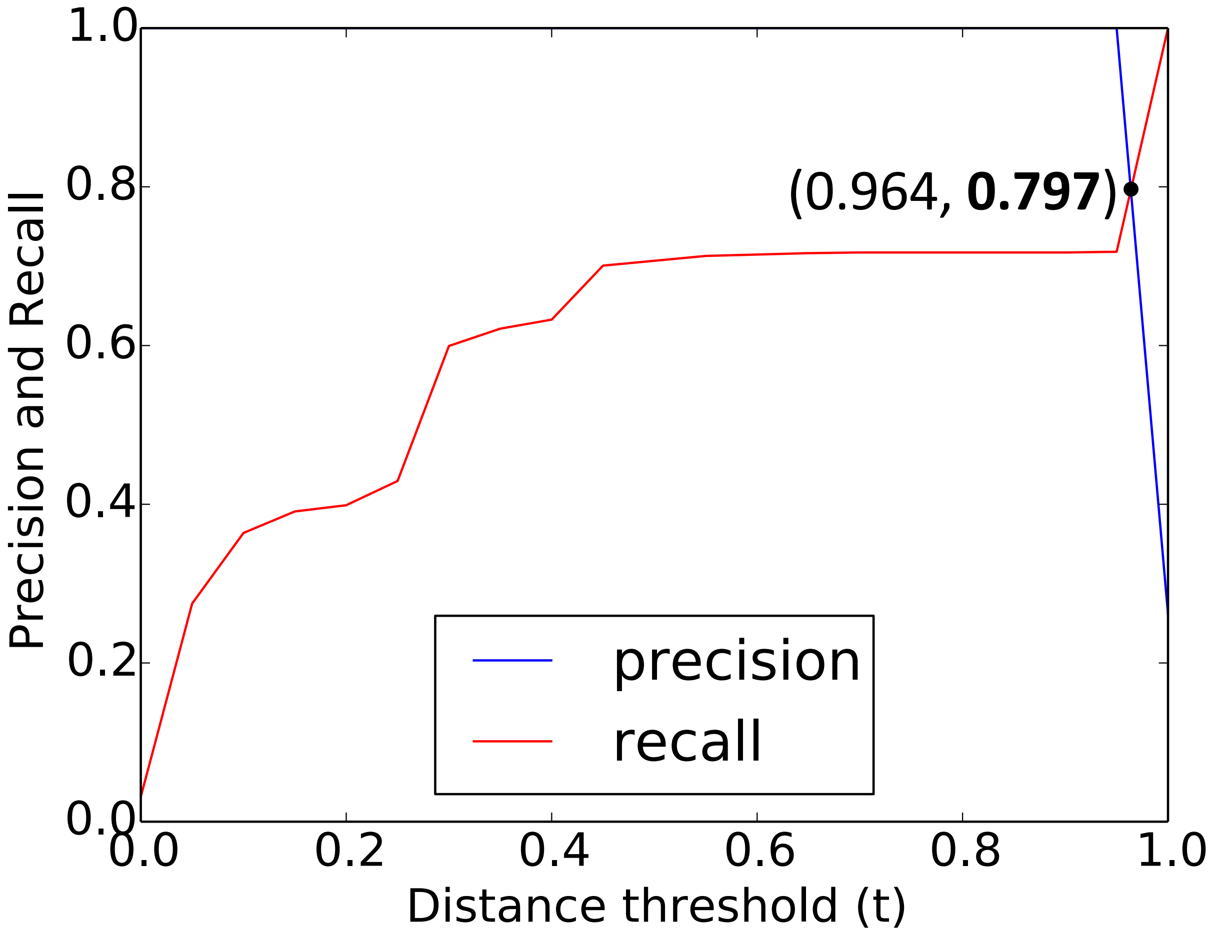
- **Precision** is to capture how well the clustering algorithm separates samples of different families to different clusters

- **Recall** is to capture how well the clustering algorithm assigns samples of same family to the same cluster

- **Intersection point** between precision and recall can be seen as good balance between the two measurements

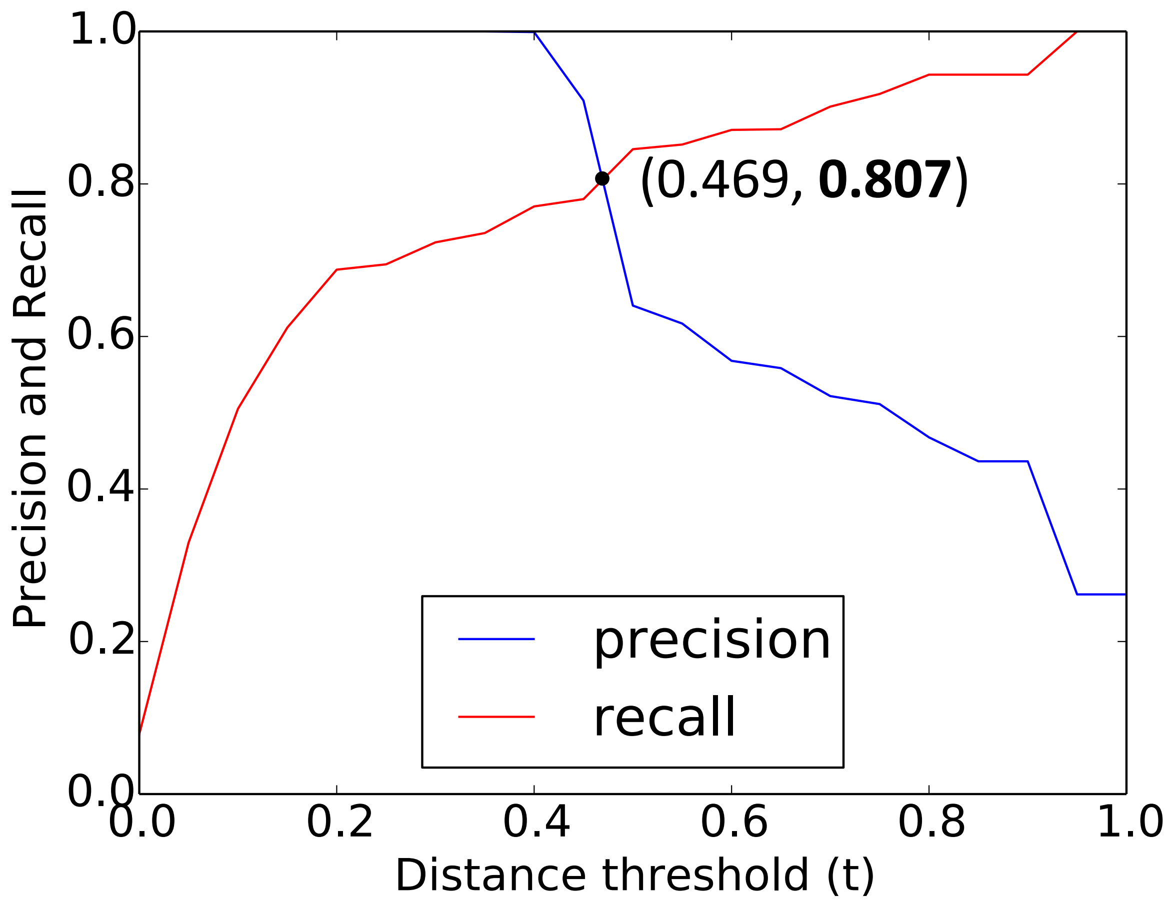
Precision & Recall of SSDeep

- Modification of SSDeep:
  - Let similarity equals 0 if two SSDeep fingerprints can not be meaningfully compared



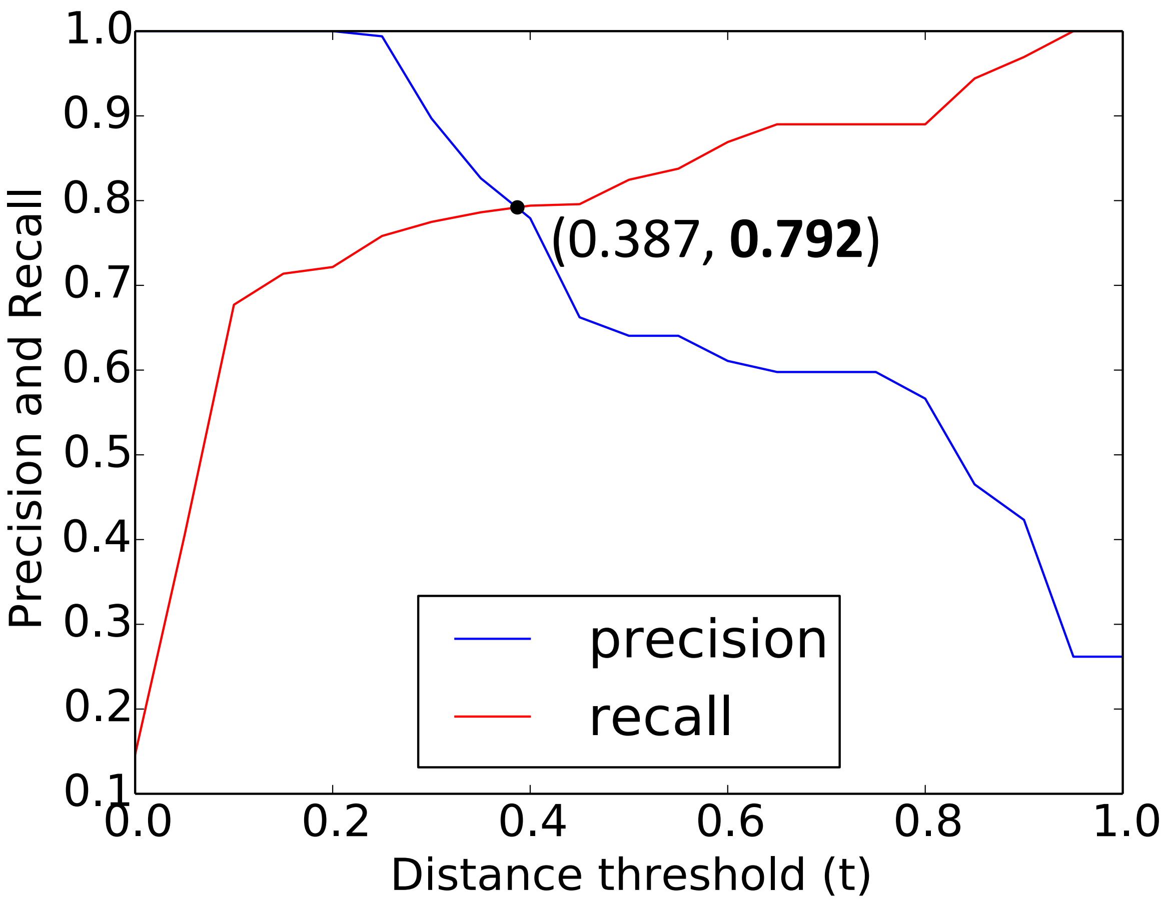
Precision & Recall of sdHash

- Modification of sdHash:
  - Fix the asymmetric distance computation problem



Precision & Recall of mvHash-B

- Modification of mvHash-B:
  - Fix the asymmetric distance computation problem



Key Elements of Fuzzy Hashing Algorithms

- Features (characteristics of input):
  - hash values of substrings, entropy values of substrings, **ngrams**

- Fingerprint (representation of features):
  - dynamic length strings, multiple bit-vectors, **single bit-vector**

- Distance function (comparison of fingerprints):
  - edit distance, customized distance, jaccard distance, etc

New Fuzzy Hash Function: nextGen-hash

- Input: 2f 2a 0a 20 2a 20 43 6f 70 79 72 69 67 68 74 20

- 4-gram Features: 2f2a0a20, 2a0a202a, 0a202a20, 202a2043, …

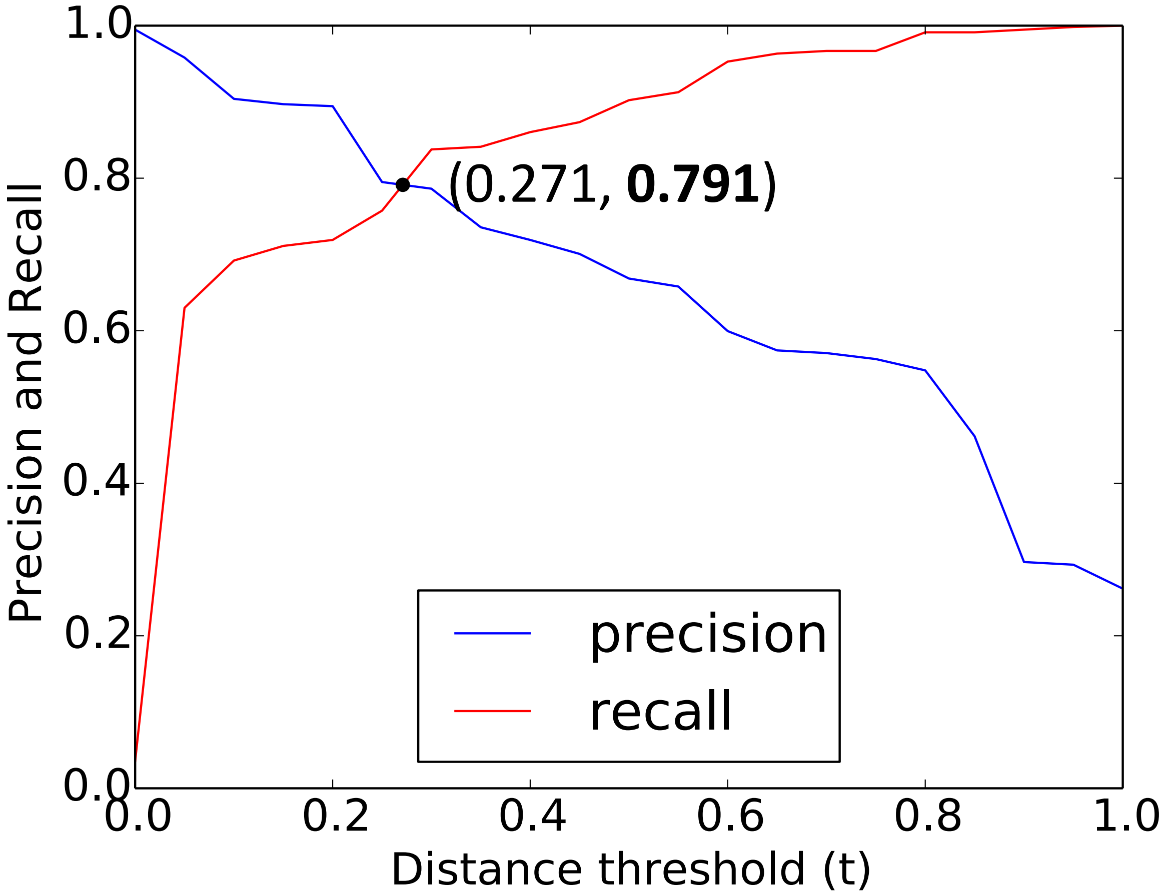
- Fingerprint: 1010100**1**0100000010100**1**01001010100**1**010010010**1**00010100

- Comparison:

$$similarity = \frac{bitcount(f_a \wedge f_b)}{bitcount(f_a \vee f_b)}$$

Precision & Recall of nextGen-hash

- Highlights of nextGen-hash:
  - Use **ngrams** as features
  - Use **bit-vector** as final fingerprint



Further Improve Fuzzy Hashing Algorithms

Algorithm Name	Whole Sample as Input (f-score)	Code Section as Input (f-score)
ssdeep	0.797	0.872
sdHash	0.807	0.877
mvHash-B	0.792	0.893
nextGen-hash	0.791	**0.919**

Summary

- We identify several **design flaws** within existing fuzzy hashing algorithms and provide algorithm and suggestion to fix the problems
- We design an **evaluation framework** and demonstrate that it can be used to compare the effectiveness of different fuzzy hash functions
- We propose a **new fuzzy hashing algorithm** and show that performances of existing fuzzy hash functions can be further improved