mvHash-B – similarity hashing

CASED

IMF, Nürnberg, 12th – 14th of March, 2013

Authors:

- Frank Breitinger
- *Knut Petter Åstebøl*
- Harald Baier
- Christoph Busch

Knut Petter Åstebøl

- **Education**
  - Bachelor degree at The Norwegian Defense University College of Engineering, 2008
  - Master degree at Gjøvik University College, 2012
    - Master thesis at CASED

- **Work experience**
  - 2008 – 2011
    - Norwegian Armed Forces, network security analyst
  - 2012 - present
    - Deloitte, senior information security consultant

Outline

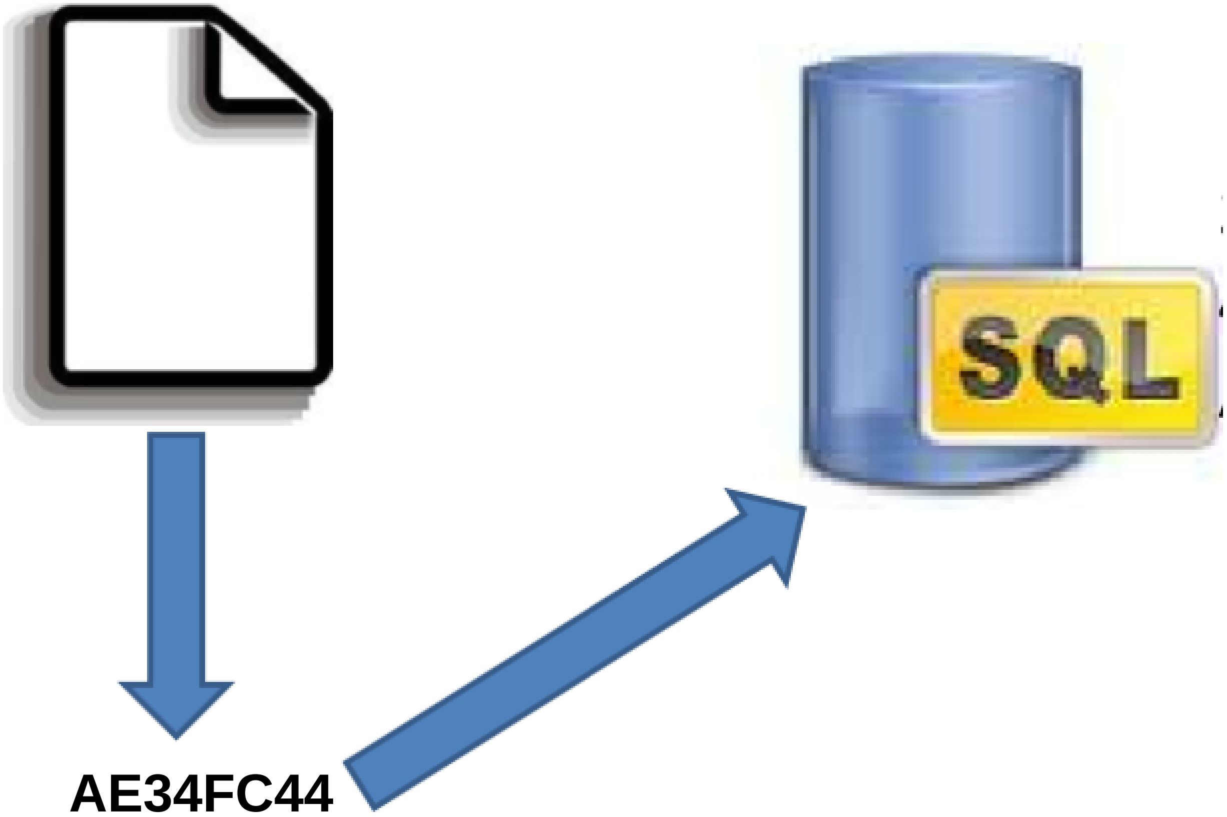
1. Motivation
2. Foundations
3. The algorithm mvHash-B
4. Experimental results
5. Conclusion and future work

1. Motivation

Digital forensics

- Criminal investigation
- Huge amount of data
- Identify known files

Digital forensics



AE34FC44

AE34FC44
EFF3421A
DAD299DD
A2E3DD44

2. Foundations

Hash

- **A hash function is a function with two properties:**
  - Compression
  - Ease of computation

- **Cryptographic hash function**
  - Avalanche effect
  - If there is a small change in the input, the output will be entirely different
  - Similar inputs will get different outputs

- **Similarity hash function**
  - The output will change proportionally to the change in the input
  - Similar inputs will get similar outputs

sdhash

- Developed by Vassil Roussev

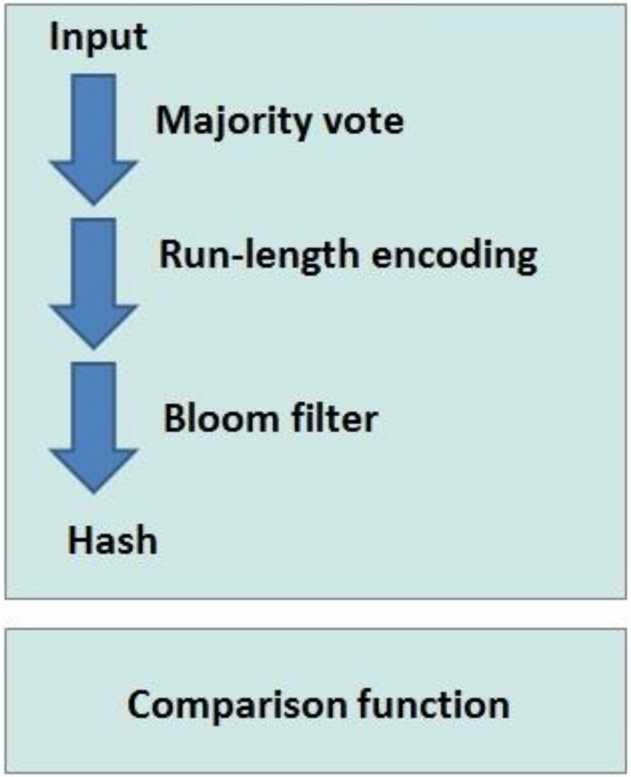
- A well-known similarity hashing algorithm

- Identifies “statistically-improbable features”

- Files are similar if they share identical features

3. The algorithm mvHash-B

Overview of mvHash-B



Input

Majority vote

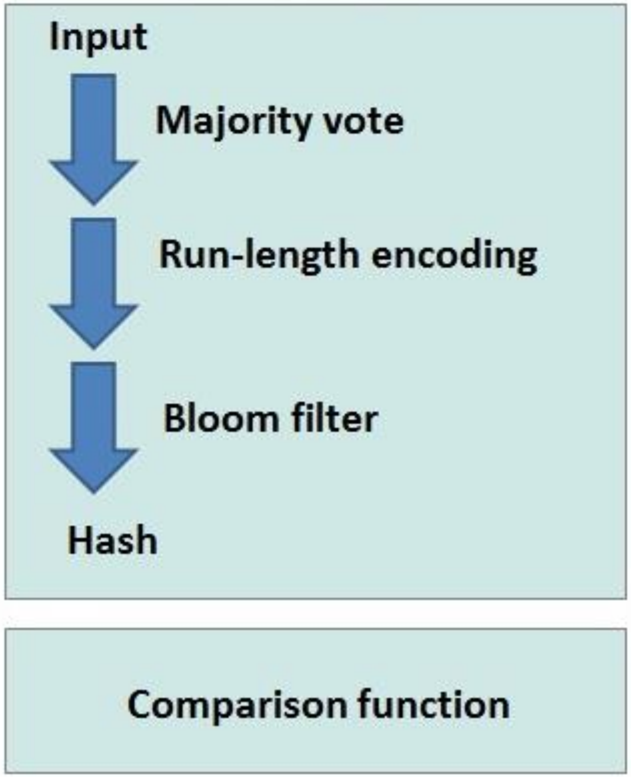
Run-length encoding

Bloom filter

Hash

Comparison function

Overview of mvHash-B



Input

Majority vote

Run-length encoding

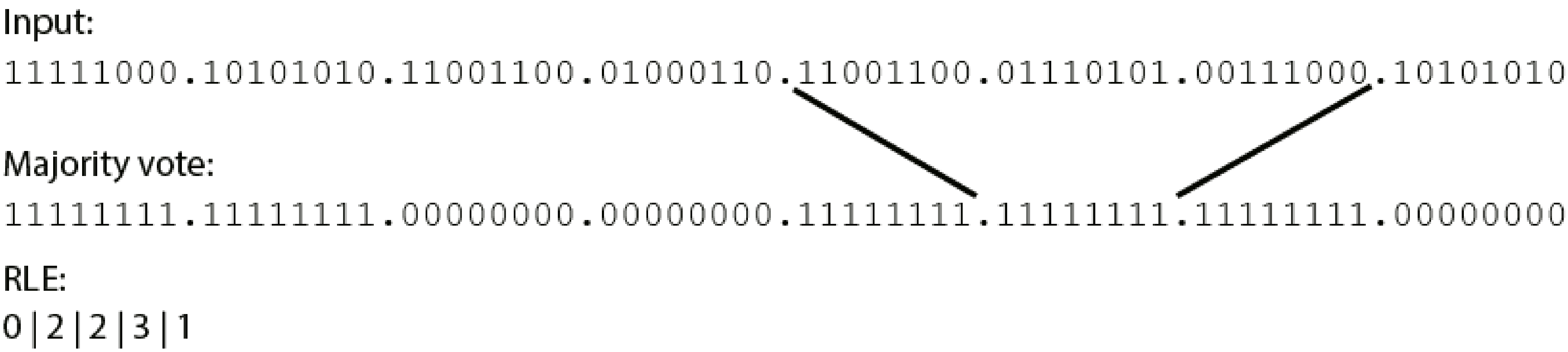
Bloom filter

Hash

Comparison function

- Enable compression
- Compression
- Enable fast comparison

Phase 1 & 2



Input:

11111000.10101010.11001100.01000110.11001100.01110101.00111000.10101010

Majority vote:

11111111.11111111.00000000.00000000.11111111.11111111.11111111.00000000

RLE:

0|2|2|3|1

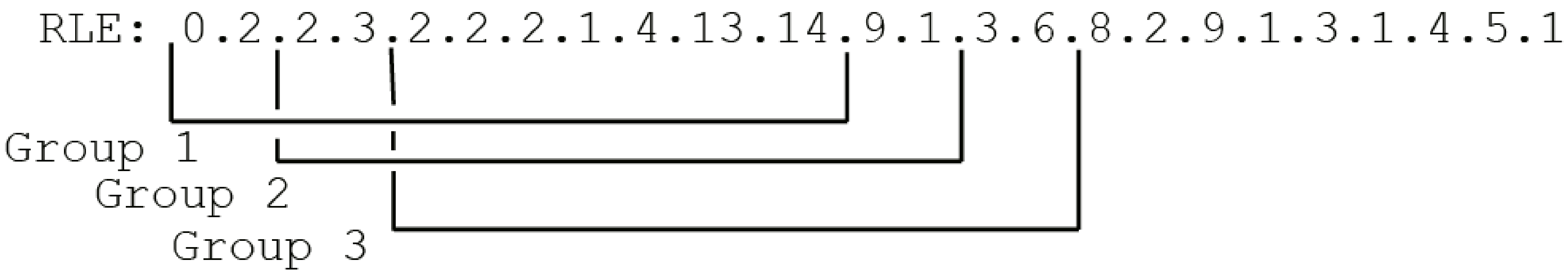
Phase 3 – Bloom filter

RLE:

```
0.2.2.3.2.2.1.4.13.14.9.1.3.6.8.2.9
```

Hash:

Phase 3 – Bloom filter



RLE: 0.2.2.3.2.2.2.1.4.13.14.9.1.3.6.8.2.9.1.3.1.4.5.1

Group 1

Group 2

Group 3

Phase 3 – Bloom filter

```
Group: 0.2.2.3.2.2.2.1.4.13.14
Entry: 0.0.0.1.0.0.0.1.0. 1. 0
```

Configurable parameters

- **Majority vote**
  - Neighbourhood size
  - Influencing bits

- **Bloom filter**
  - Entries per Bloom filter

- **Different parameters for different file types!**

Comparison algorithm

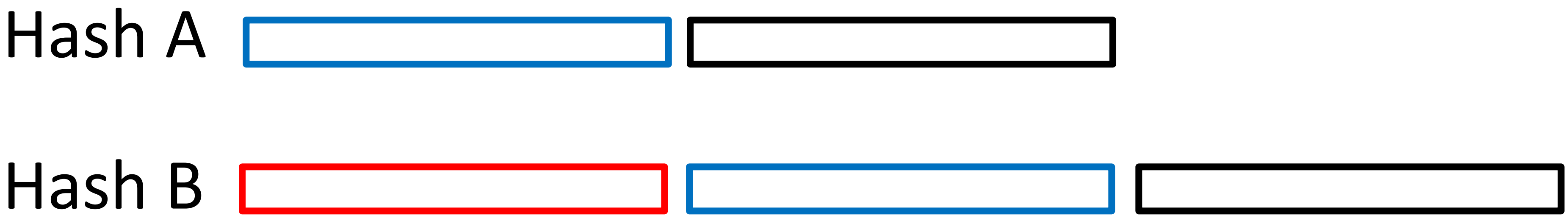
- Compares two hashes
- Outputs a value between 0 and 100

Comparison algorithm

Hash A

Hash B

Comparison algorithm



Hash A

Hash B

4. Experimental results

Experimental results

- **Platform**
  - Ubuntu 11.10 Desktop

- **Programming language**
  - C

- **Test corpuses**
  - C-corpus
  - t5-corpus

- **File types**
  - jpg and doc

Experimental results

- **Efficiency**

- **Run time efficiency**

SHA-1	**mvHash-B**	**Sdhash**
1.0	1.48	14.48

- **Compression**

mvHash-B	**sdhash**
0.59%	2.60%

Experimental results

- **Accuracy**
    - Ability to detect similar files with low cost in terms of false positive and false negative results
    - Almost no false positive

Experimental results

- **Robustness**
  - Ability to detect similar files
  - How many bytes in the file may be changed and the file will still be recognized as similar to the original file?

mvHash-B	**sdhash**
0.50%	0.92%

5. Conclusion & future work

Conclusion & future work

- **Conclusion**
  - mvHash-B is a new approach which uses three trivial phases
  - It is able to distinguish between similar and non-similar files
  - Great run time efficiency and compression

- **Future work**
  - Automatic detect file type and configure the parameters accordingly

Contact, discussion

- **Thank you for your attention!**

- **Knut Petter Åstebøl**
  - kaasteboel@Deloitte.no

- **Questions?**