Fuzzy hashing, или хэши наоборот

Тюрин Кай

О себе

- Аспирант Института математики, механики и информатики им. Воровича, где занимаюсь научной деятельностью в области ИБ

- Сотрудник научно-исследовательского института, где занимаюсь задачами практической ИБ

Почему я хочу об этом рассказать?

- Очень мало информации об этой теме
  (на русском нет совсем)

Почему я хочу об этом рассказать?

- Очень мало информации об этой теме

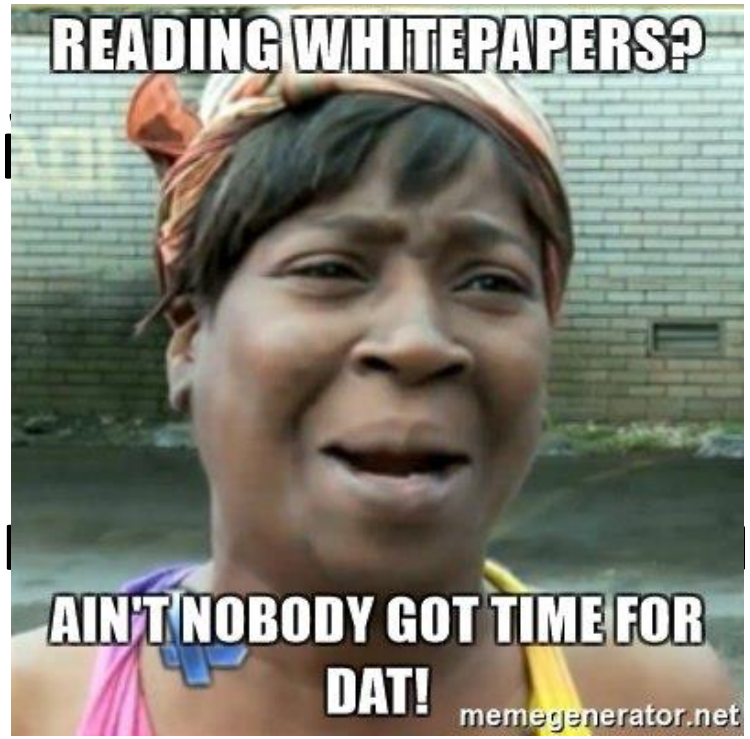
  (на русском нет совсем)

- Почти вся информация на английском содержится в научных статьях

Почему я хочу об этом рассказать?

- Очень мало информации об этом
  (на русском нет совсем)

- Почти вся информация на англи
  в научных статьях



Почему я хочу об этом рассказать?

- Очень мало информации об этой теме
  (на русском нет совсем)
- Почти вся информация на английском содержится в научных статьях
- Тема очень интересна даже для не-безопасников

Что такое fuzzy hash?

Что такое fuzzy hash?

- Это как обычный хэш, но наоборот

Что такое fuzzy hash?

- Это как обычный хэш, но наоборот

Обычный хэш

hash(C6 52 63 A0 A1 71 1C 69) = C0 90 6B 92

Что такое fuzzy hash?

- Это как обычный хэш, но наоборот

Обычный хэш

hash(C6 52 63 A0 A1 71 1C 69) = C0 90 6B 92

hash(C6 52 63 B0 A1 71 1C 69) = 9F AF A7 30

Что такое fuzzy hash?

- Это как обычный хэш, но наоборот

“Нечёткий” хэш

hash(C6 52 63 A0 A1 71 1C 69) = EC 07 35 1B

Что такое fuzzy hash?

- Это как обычный хэш, но наоборот

“Нечёткий” хэш

hash(C6 52 63 A0 A1 71 1C 69) = EC 07 35 1B

hash(C6 52 63 B0 A1 71 1C 69) = EC 09 35 1B

- Fuzzy hashing
- Locality-sensitive hash
- Similarity digest
- …

Самая простая идея

Давайте хэшировать не весь файл,
а куски по отдельности!

Как будем делить файл?

Как будем делить файл?

- На куски фиксированного размера!

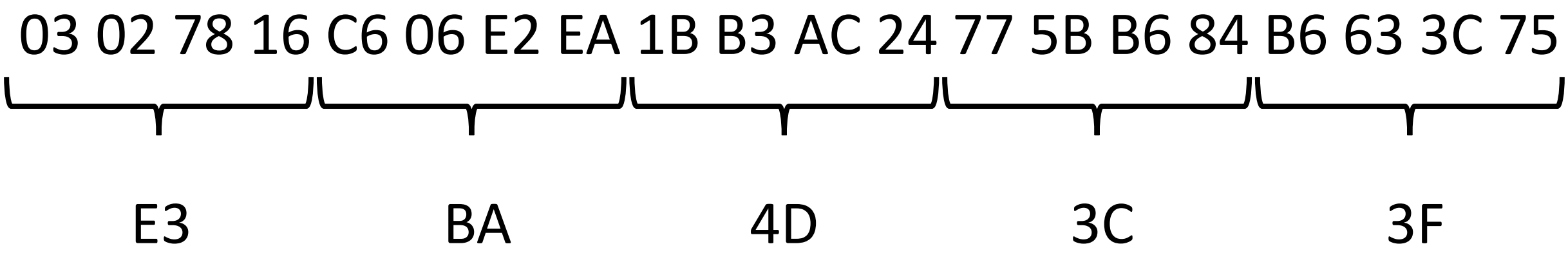
Как будем делить файл?

- На куски фиксированного размера!

03 02 78 16 C6 06 E2 EA 1B B3 AC 24 77 5B B6 84 B6 63 3C 75

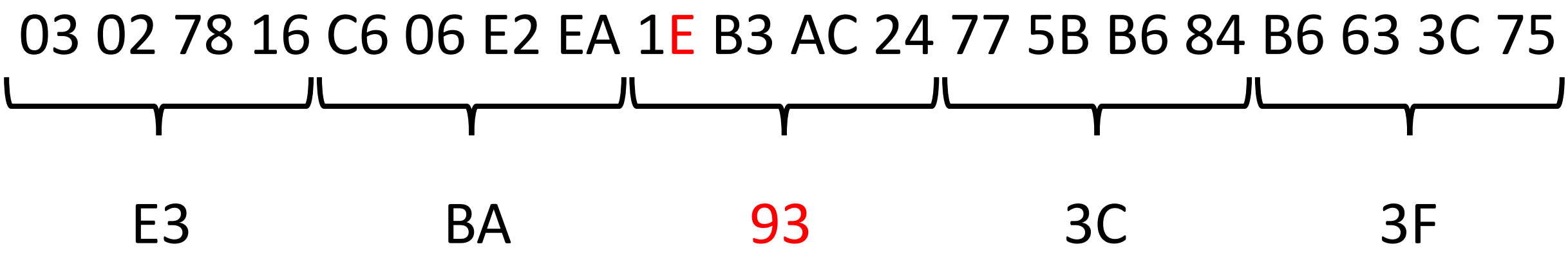
Как будем делить файл?

- На куски фиксированного размера!



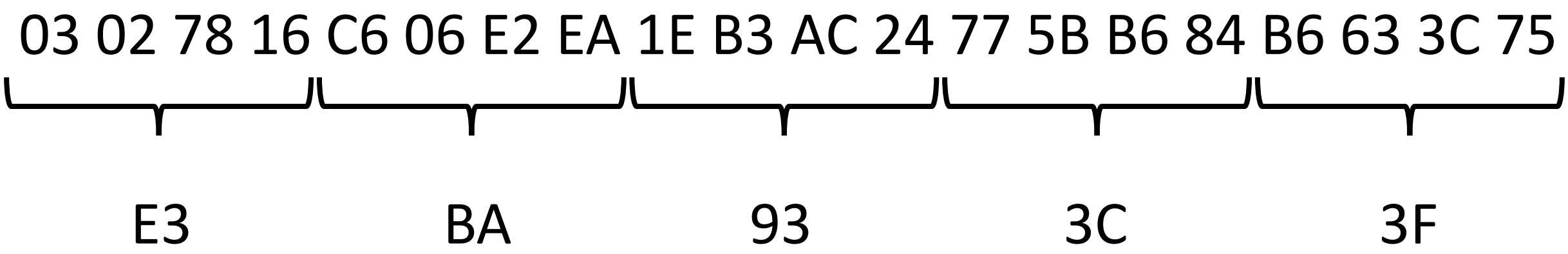
Как будем делить файл?

- На куски фиксированного размера!



Как будем делить файл?

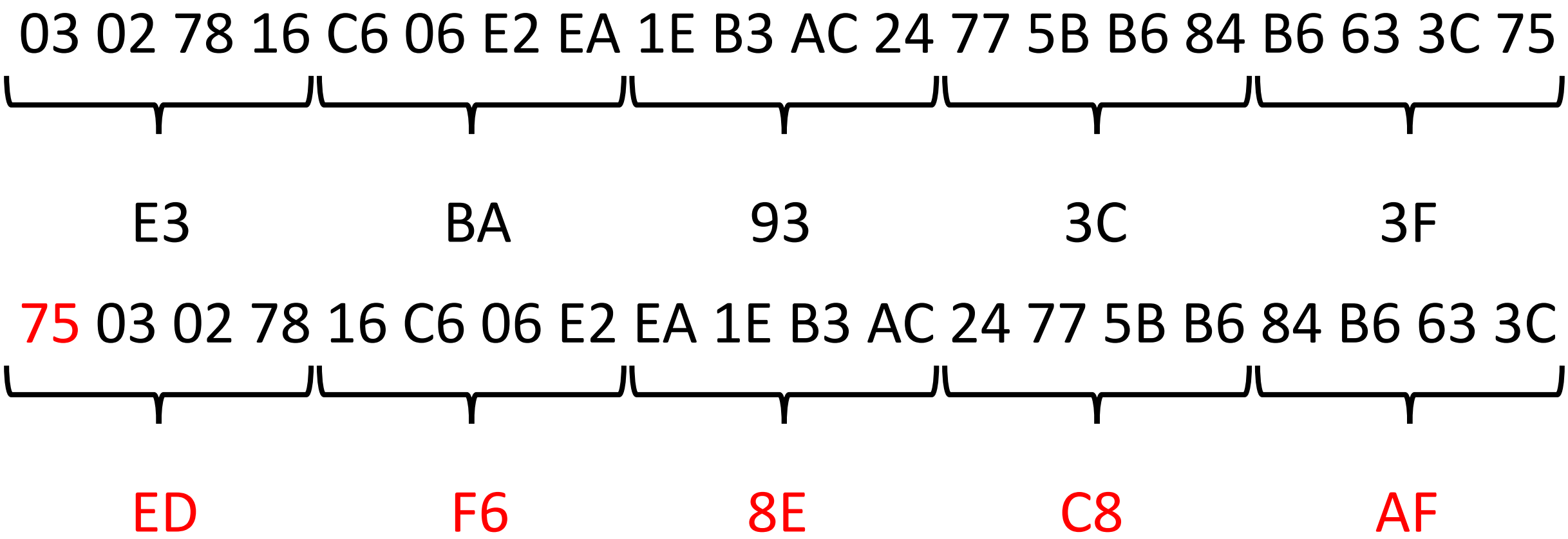
- На куски фиксированного размера!



Piecewise hashing – идея из dcfldd

Как будем делить файл?

- На куски фиксированного размера!

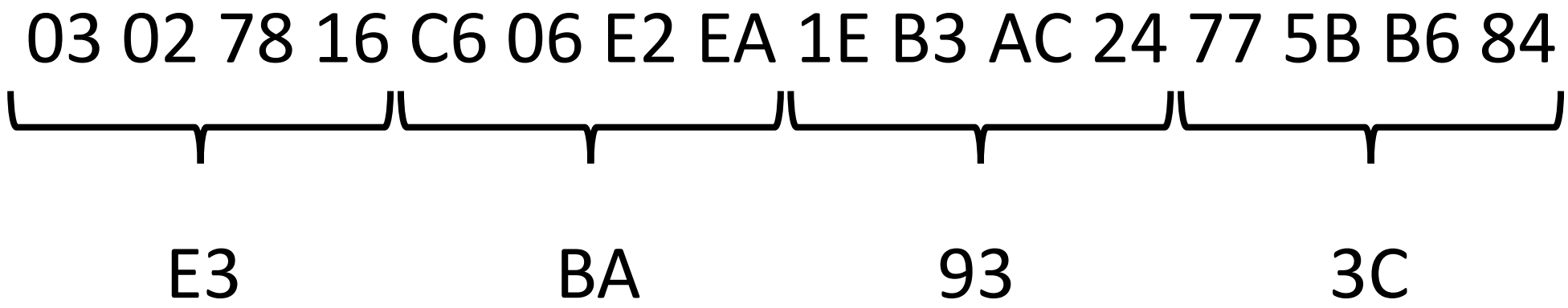


Как будем делить файл?

- ~~На куски фиксированного размера!~~

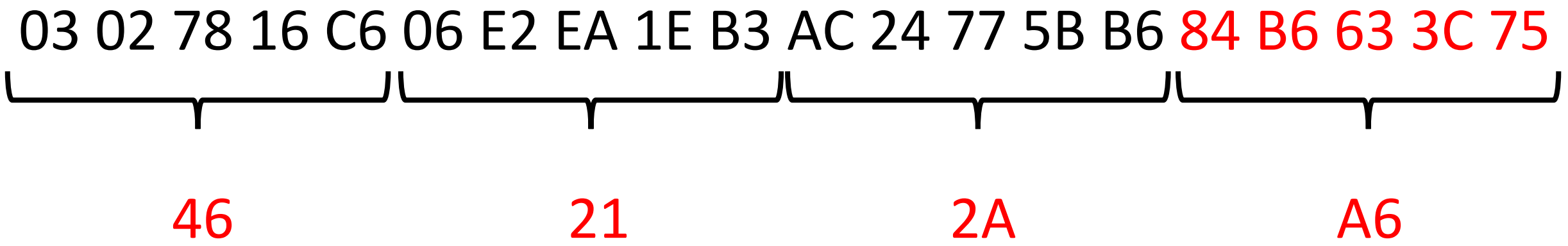
Как будем делить файл?

- ~~На куски фиксированного размера!~~
- На определенное количество кусков!



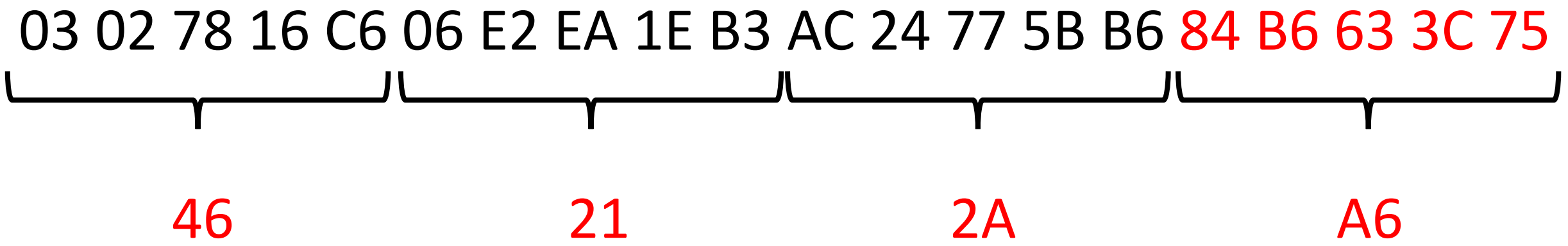
Как будем делить файл?

- ~~На куски фиксированного размера!~~
- На определенное количество кусков!



Как будем делить файл?

- ~~На куски фиксированного размера!~~
- На определенное количество кусков!



Совсем плохо

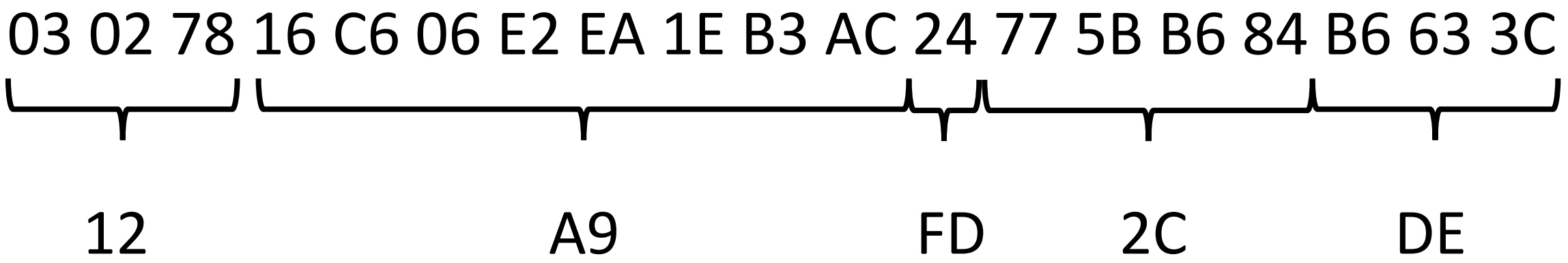
Как будем делить файл?

- ~~На куски фиксированного размера!~~
- ~~На определенное количество кусков!~~

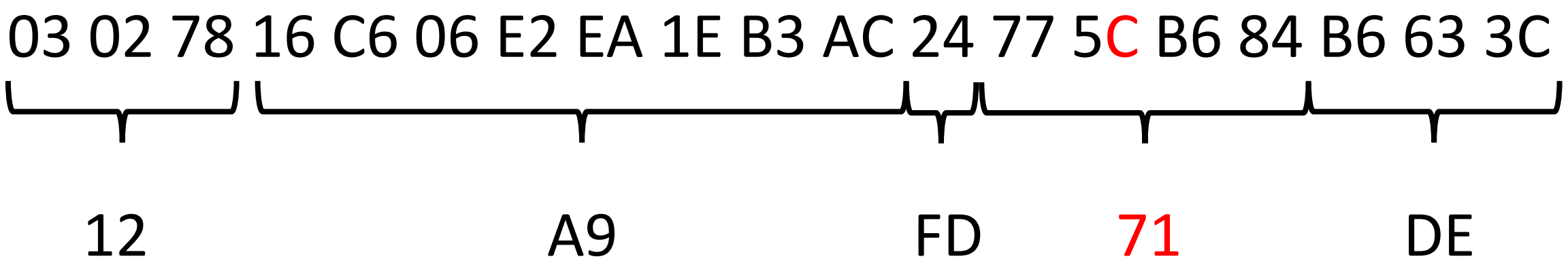
Как будем делить файл?

- ~~На куски фиксированного размера!~~
- ~~На определенное количество кусков!~~
- Попробуем разделять куски в зависимости от контента?

Пример: разделитель – байты с делимостью на 4

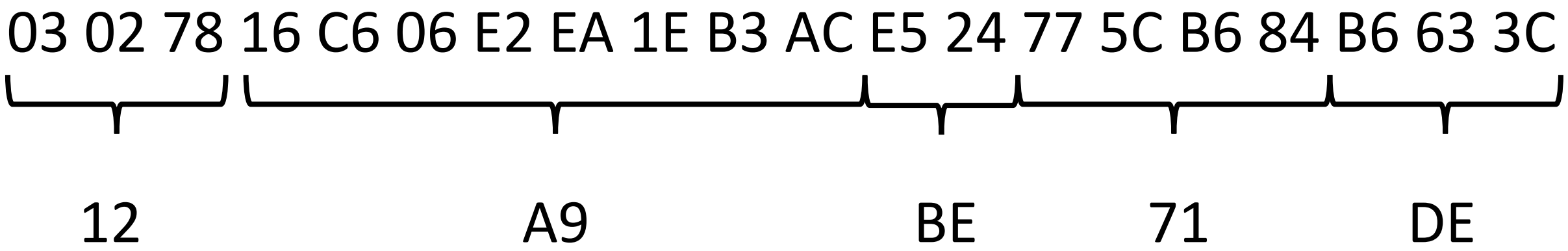


Пример: разделитель – байты с делимостью на 4



Пример: разделитель – байты с делимостью на 4

Пример: разделитель – байты с делимостью на 4



CTPH – Context triggered piecewise hashing

spamsum и SSDEEP – первый Fuzzy хэш

- Rolling hash (как в rsync)
- Куски хэшируются при помощи FNV, в хэш записывается base64 от младших 6 бит
- Окно подбирается так, чтобы хэш был 80 байт длиной
- Два окна: x и 2x (потому что сравнивать можно хэши только с одинаковым размером окна)

Более нетривиальные идеи

1. Искать какие-то особенно характерные куски для файла и сравнивать их

SDHash

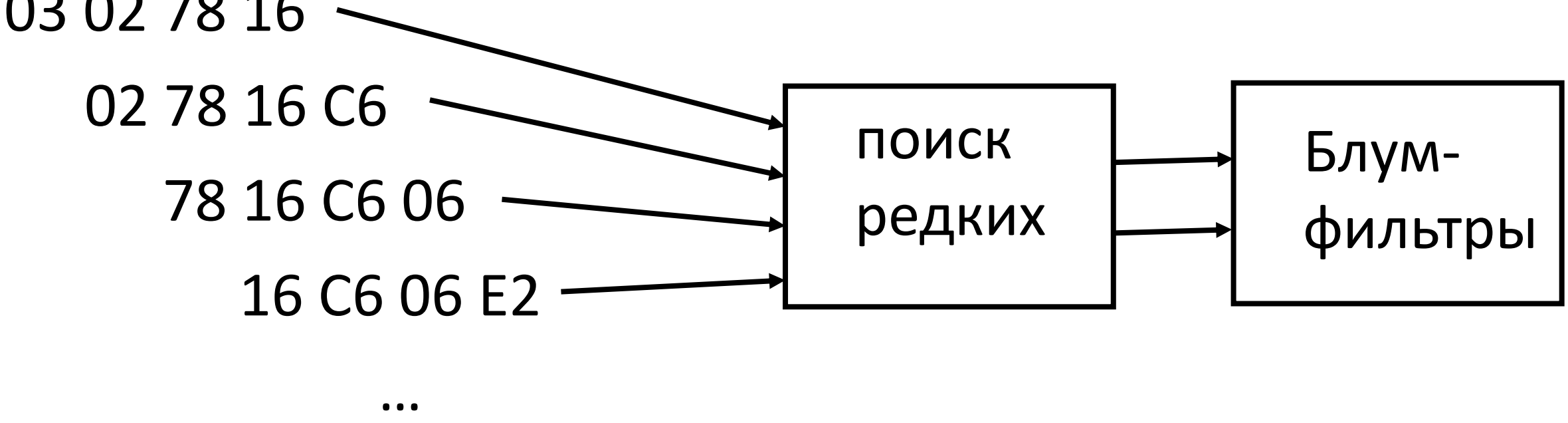
Считает энтропию 64-байтных участков

Определяет статистически маловероятные куски

Хэширует их и загоняет в блум-фитры

SDHash

03 02 78 16 C6 06 E2 EA 1E B3 AC E5 24 77 5C B6 84 B6 63 3C



Более нетривиальные идеи

1. Искать какие-то особенно характерные куски для файла и сравнивать их

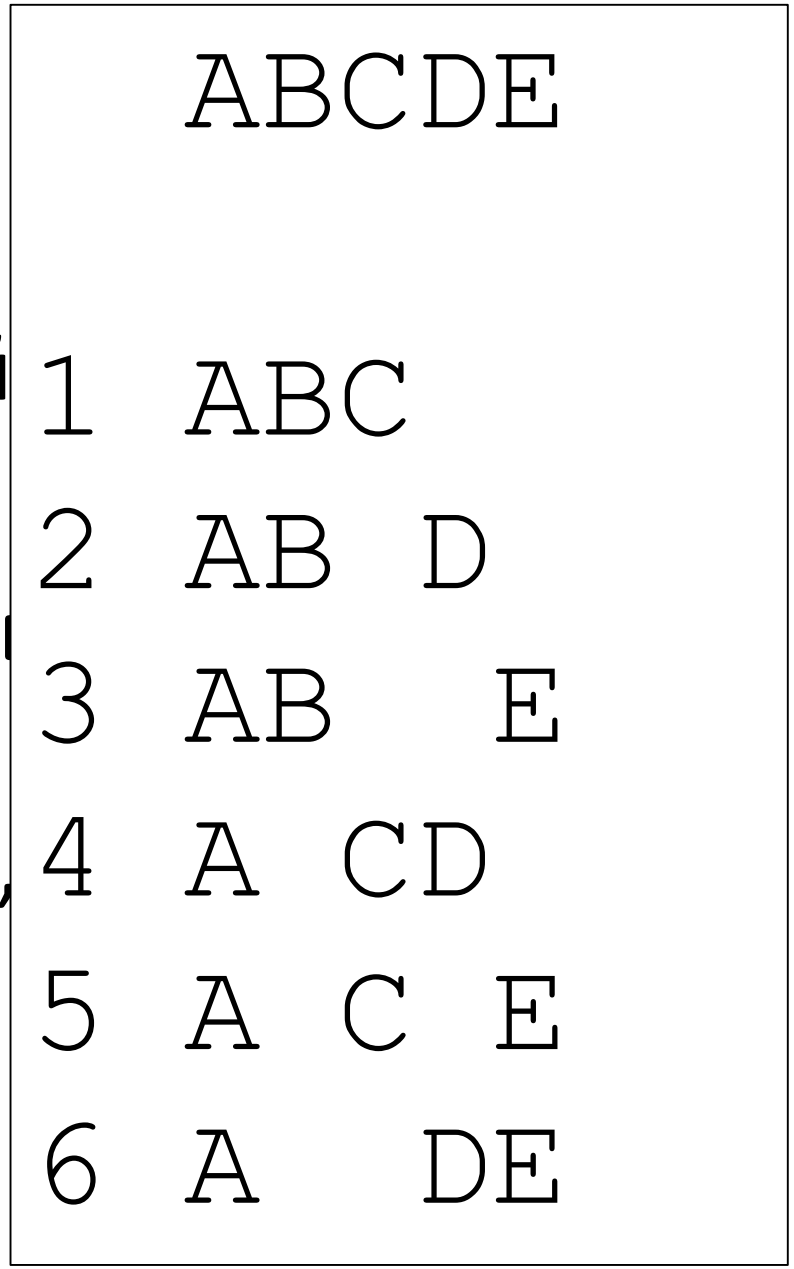
2. Строить некоторое вероятностное распределение байтовых последовательностей в файле

Nilsimsa, TLSH

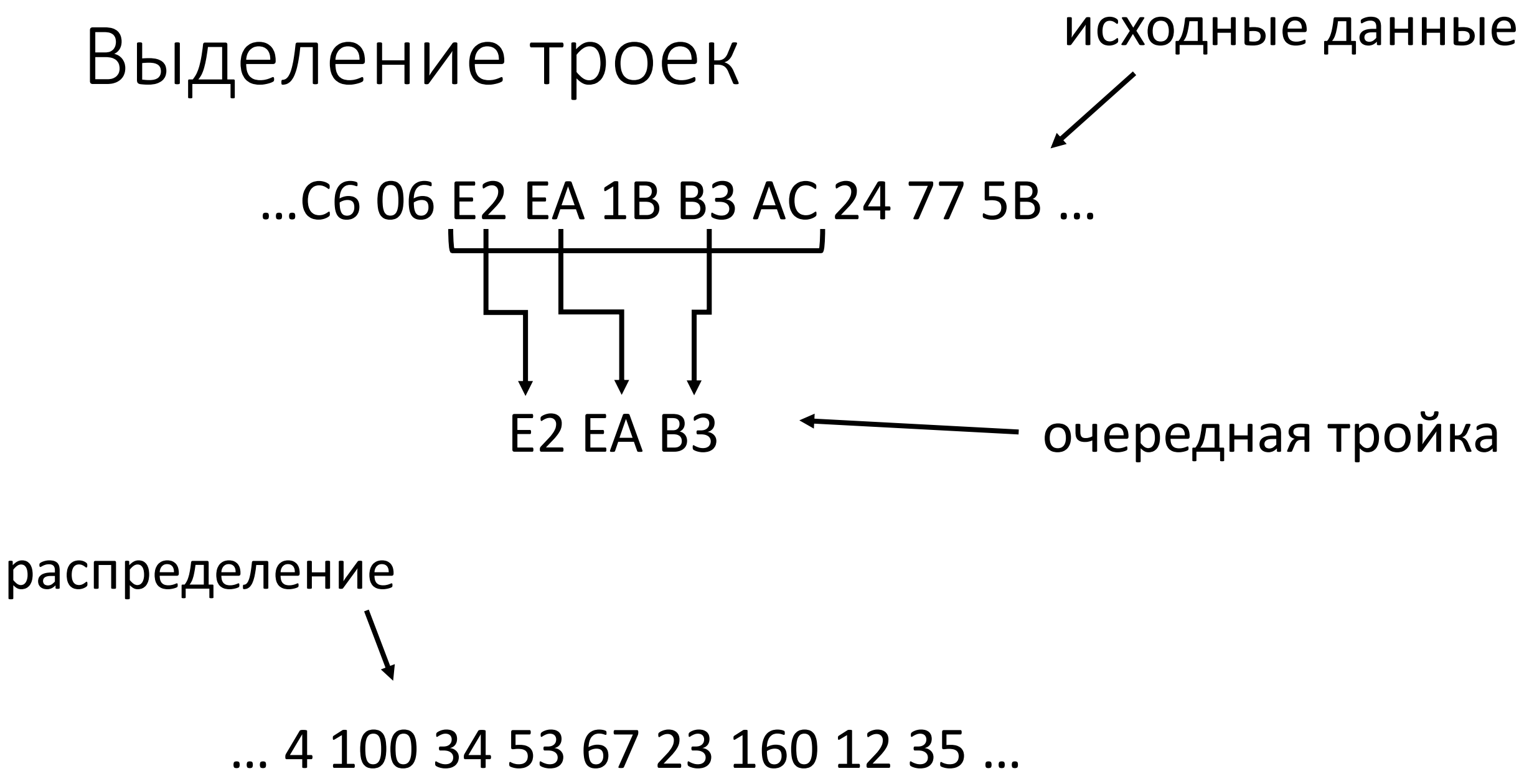
- Пробегает по файлу окном в 5 байт длиной
- Из 5 байт генерируется 6 троек
- Каждая тройка хэшируется и добавляется к гистограмме
- Гистограммы делятся на квартили, результат – принадлежность квартилям

Nilsimsa, TLSH

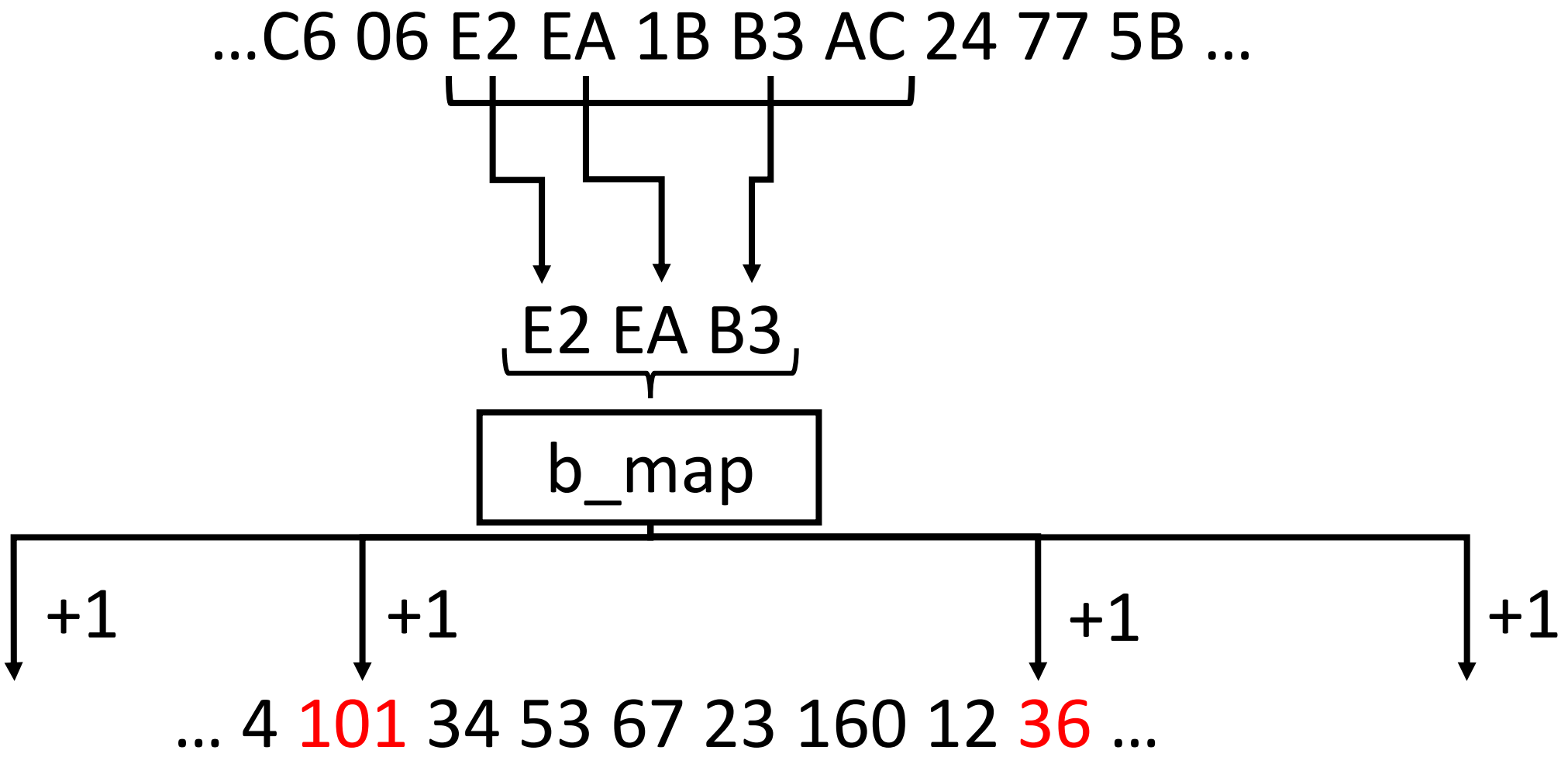
- Пробегает по файлу окном в 5 бай
- Из 5 байт генерируется 6 троек
- Каждая тройка хэшируется и доба гистограмме
- Гистограммы делятся на квартили, принадлежность квартилям



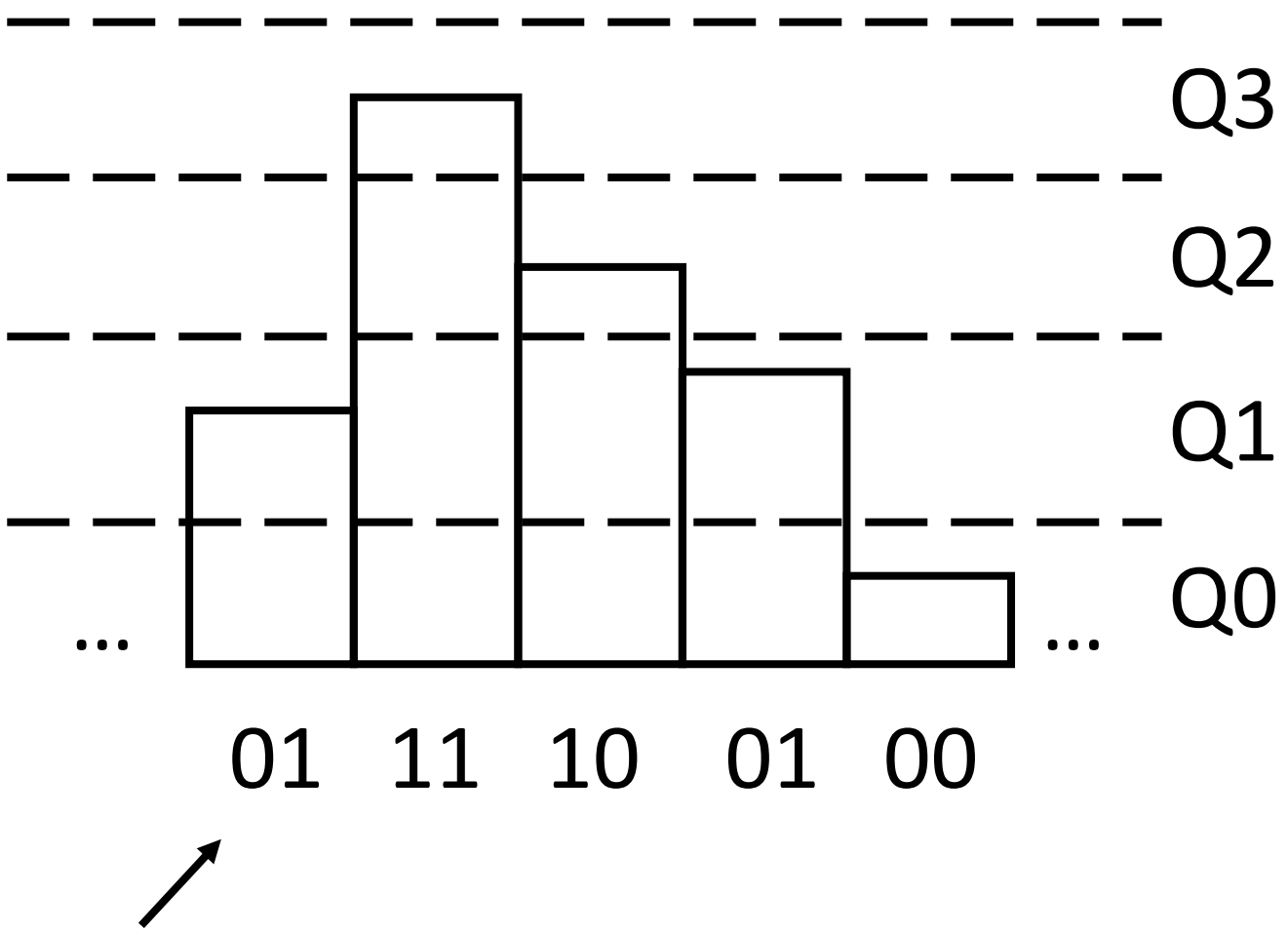
Выделение троек



Увеличение гистограммы

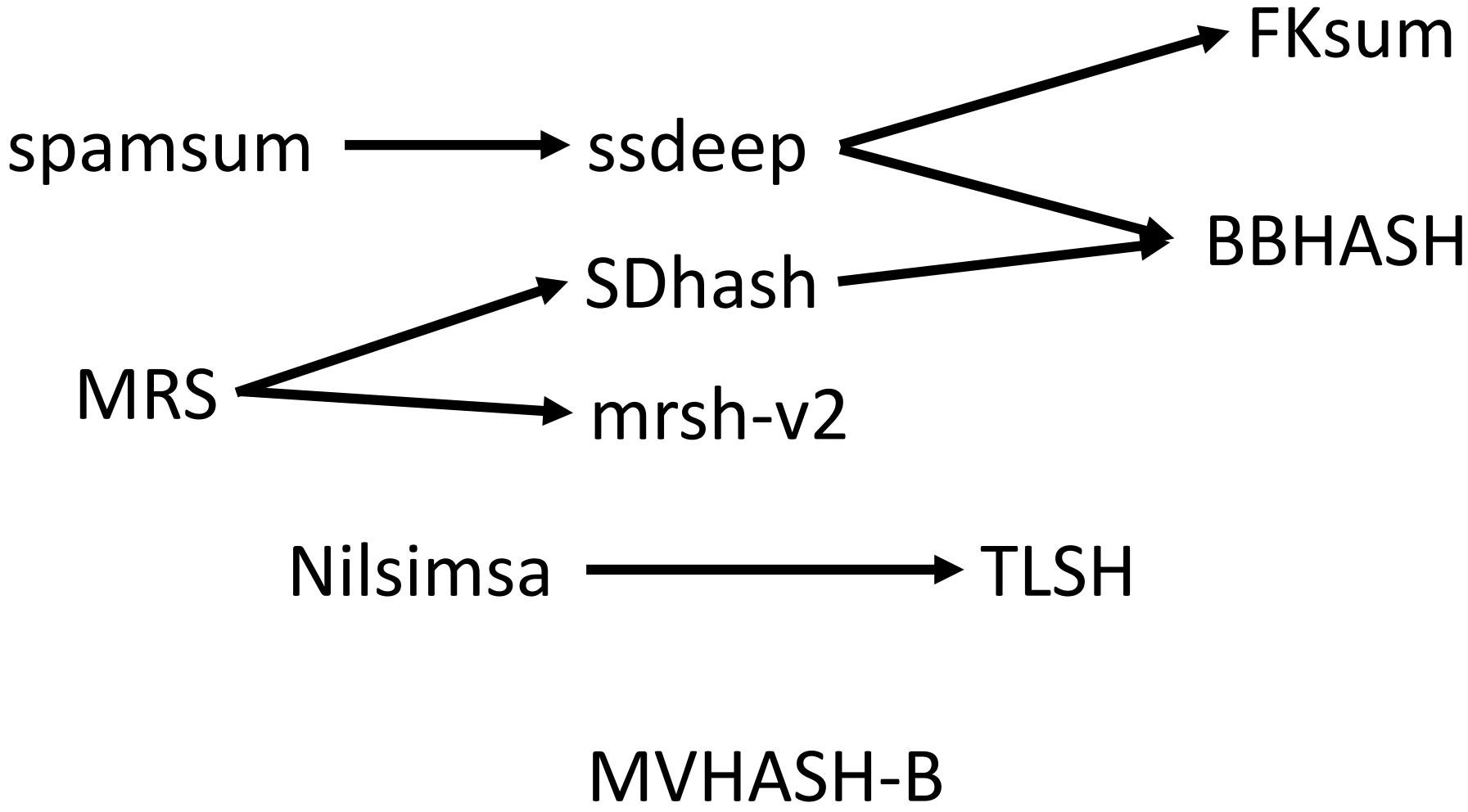


Вычисление конечного хэша



биты результирующего хэша

А что там есть еще?



И где это применяется?

- Спам-фильтры
- Обнаружение вредоносного ПО
- Форензика
- Антиплагиат
- Определение сходства изображений

Мало кто знает, но…

Анализ | Сведения о файле | Родство | Дополнительные сведения | Комментарии 0

File identification

MD5	a14efbfb2e56ad51b6a9d3190b01ce98
SHA1	ef45c5ae6472761850894c4712d2f0c2a52da5db
SHA256	a2194923b6277db953489c884bf6e4f1976ff50ad159a34b756bef6fc47b7128
ssdeep	24576:T0z9ZUBfc5U7wrIWH+kEmVKm5Ba6463RAfA1L:/fc5lb+kEgKABxN1L
authentihash	6cf06f5658c59a571c1a6aba45453382d6e9f58c70932f357d2a18d39c3b8ba0
imphash	3b471bcd644c5120f87ea09425e064ba
Размер файла	2.2 МБ (2284024 bytes)
Тип файла	Win32 EXE
Описание	PE32 executable for MS Windows (GUI) Intel 80386 32-bit

Еще меньше людей знает, что...

National Software Reference Library (NSRL)

- Curated Kaspersky Hash Set - September 2017
- About the NSRL +
- NSRL Download −
  - Current RDS Hash Sets
  - Non-RDS Hash Sets
  - RDS Query Tools
  - NSRL Legacy Tool Downloads

Non-RDS Hash Sets

On this page, we will make links available to data sets that use hashing or digest algorithms not contained in the RDS release. If there is an algorithm or process that you think would be interesting to run on the NSRL file corpus files found in the RDS (or on a subset of files), please contact nsrl@nist.gov .

NOTE: These downloads are being served from the amazon cloud. You may receive a notice that you are leaving the NSRL website.

- Data sets using the ssdeep algorithm (aka "fuzzy hashes")
- Data sets using the sdhash algorithm (aka "similarity digest")
- Data sets using the bulk_extractor tool

Пример работы

pepe1.bmp

pepe2.bmp

pepe3.bmp

Пример работы (ssdeep)

```
>ssdeep.exe pepe1.bmp
ssdeep,1.1--blocksize:hash:hash,filename
6144:VJszgFOrgMV8y4HZJLaCnDFXXXXXX1I33KvYcIhgbfbbb1:
8HzVl45JBzNN,"D:\tlsh-test\pepe1.bmp"

>ssdeep.exe pepe2.bmp
ssdeep,1.1--blocksize:hash:hash,filename
6144:VJszgFOrgMV8y4HZJLaCnDZXXXXXXBD33JjNdUugbfbbb1:
8HzVl45JBzt6,"D:\tlsh-test\pepe2.bmp"

>ssdeep.exe pepe3.bmp
ssdeep,1.1--blocksize:hash:hash,filename
12288:o5l7Zi4Qry0bNxNwMfALsO30qhHdR9wr79D5oLr4QWJL1yW1zoXG92y
2Jn/RxYwk:474hd2MILs1ZrAz0X,"D:\tlsh-test\pepe3.bmp"
```

Пример работы (tlsh)

```
>tlsh.exe -f pepe1.bmp
c5350a3ba5851817cb1eb13b40b98715f8f0db8b0a19679e
5168f3fa3fab1703a461d4  pepe1.bmp

>tlsh.exe -f pepe2.bmp
02350b3ba5841817db5eb13740b98715f8f0db4b0a19679e
6268f3fa3fab1303a461d4  pepe2.bmp

>tlsh.exe -f pepe3.bmp
da35e9950d48f50bd7a0ea5ce7c9293f7525a9a4309b863d
8cca70bcd864ce3d32b6d4  pepe3.bmp
```

Оригинальные пейперы

- Nilsimsa: Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. 2004. An Open Digestbased Technique for Spam Detection. ISCA PDCS 2004 (2004), 559–564.
- SSDeep: Jesse Kornblum. 2006. Identifying almost identical files using context triggered piecewise hashing. Digital investigation 3 (2006), 91–97.
- TLSH: Jonathan Oliver, Chun Cheng, and Yanggui Chen. 2013. TLSH–A Locality Sensitive Hash. In Cybercrime and Trustworthy Computing Workshop (CTC), 2013 Fourth. IEEE, 7–13.
- SDHash: Vassil Roussev. 2010. Data fingerprinting with similarity digests. In IFIP International Conference on Digital Forensics. Springer, 207–226.

Где меня найти

Twitter: https://twitter.com/turinkay

Telegram: https://t.me/kayvflu

VK: https://vk.com/kayvflu

Почта: kayvflu@gmail.com

А вот тут будут слайды и экспериментальные данные:

https://github.com/turinkay/presentations