Security Aspects of Piecewise Hashing in Computer Forensics

Harald Baier, Frank Breitinger

Hochschule Darmstadt, CASED

IMF, 2011-05-10

Harald Baier

1. Doctoral degree from TU Darmstadt in the area of elliptic curve cryptography.
2. Principal Investigator within Center for Advanced Security Research Darmstadt (CASED)
3. Establishment of forensic courses within Hochschule Darmstadt.
4. Current working fields:
   - Fuzzy Hashing (IT forensics, biometrics, malware detection).
   - Real-time and efficient detection of malware.
   - Anomaly detection in high-traffic environments.

Motivation

Motivation

Foundations of Hash Functions

Motivation

Foundations of Hash Functions

Piecewise Hashing

Motivation

Foundations of Hash Functions

Piecewise Hashing

Anti-Forensics on CTPH

Motivation

Foundations of Hash Functions

Piecewise Hashing

Anti-Forensics on CTPH

Summary

Motivation

Use Case: Prosecution

1. Police and prosecutors confronted with different storage media:
   - Hard disk drives, solid-state drives, USB sticks.
   - Mobile phones, SIM cards.
   - Digital cameras, digital camcorders, SD cards.
   - CDs, DVDs.
   - RAM (dumps).
   - ...
2. Amount of distrained data often exceeds 1 terabyte.

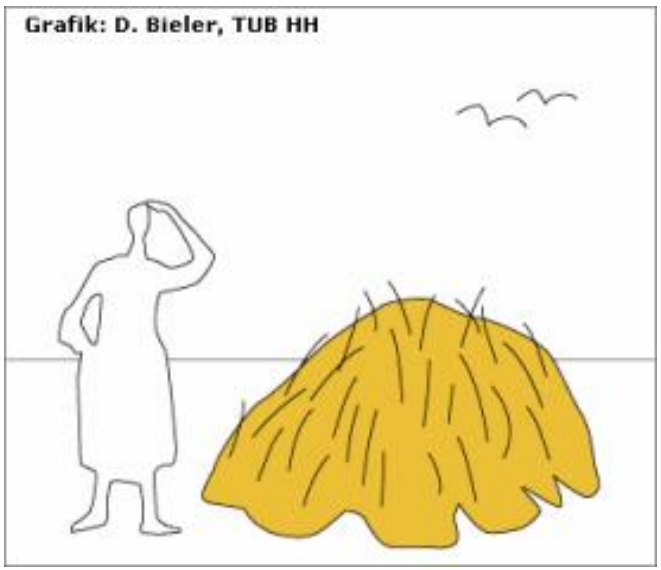
Different views of 1 terabyte (idea due to BKA)

1 terabyte of digital text is (approximately) equal to:

1. 1 trillion characters: 1 character = 1 byte.
2. 220 million pages: 1 page = 5000 characters.
3. 21 years of printing time: 20 sheets per minute.
4. 1 million kg of paper: onesided printed.
5. Paper stack of 22 km height: bulk of 0.1 mm.

Motivation

Finding relevant files resembles ...

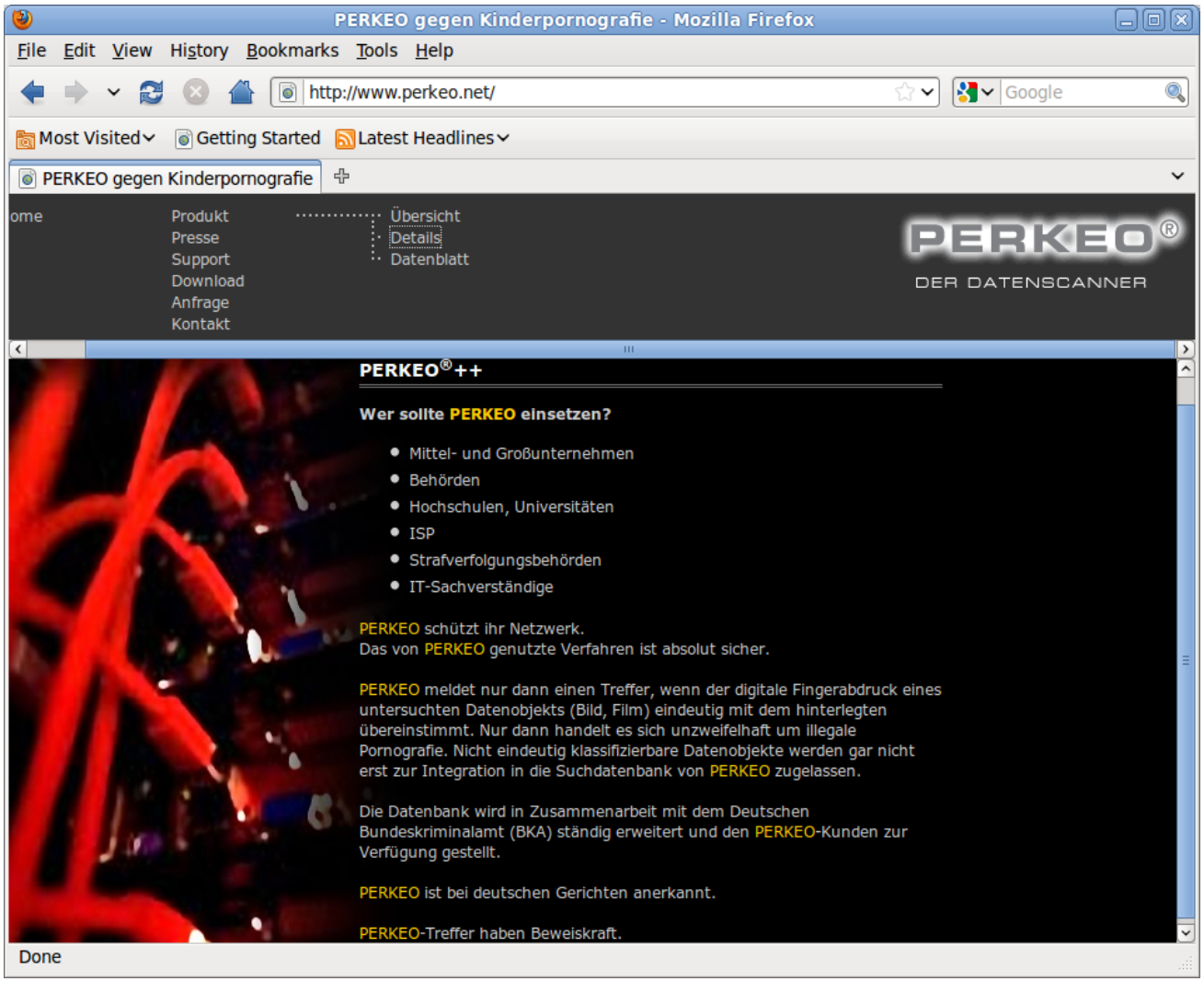


Source: tu-harburg.de

Source: beepworld.de

Motivation

... or is it solved for suspect files?



Aims of the Use of Hash Functions

Automated identification of known files.

Aims of the Use of Hash Functions

Automated identification of known files.

1. Accuracy: Find ...
   - ... identical files.
   - ... similar files.

Aims of the Use of Hash Functions

Automated identification of known files.

1. Accuracy: Find ...
   - ... identical files.
   - ... similar files.
2. Logical level: Decision based on the ...
   - ... semantic / perceptual level.
   - ... byte level.

Aims of the Use of Hash Functions

Automated identification of known files.

1. Accuracy: Find ...
   - ... identical files.
   - ... similar files.
2. Logical level: Decision based on the ...
   - ... semantic / perceptual level.
   - ... byte level.

Promise of piecewise hashing: Support automatic finding of files similar to known files on the byte level.

Foundations of Hash Functions

Definition and Security Applications

1. A hash function h is a function with two properties:
   - Compression: $h : \{0,1\}^* \longrightarrow \{0,1\}^n$.
   - Ease of computation: Computation of $h(m)$ is 'fast' in practice.
2. Notation:
   - m is a 'document' (e.g. a file, a device).
   - $h(m)$ its *hash value* or *digest*.

Definition and Security Applications

1. A hash function h is a function with two properties:
   - Compression: $h : \{0,1\}^* \longrightarrow \{0,1\}^n$.
   - Ease of computation: Computation of $h(m)$ is 'fast' in practice.
2. Notation:
   - m is a 'document' (e.g. a file, a device).
   - $h(m)$ its *hash value* or *digest*.
3. Cryptographic hash functions follow the avalanche effect:
   - If m is replaced by m', $h(m')$ behaves pseudo randomly.
   - Every bit in $h(m')$ changes with probability 50%, independently of the number of different bits in m'.
   - No control over the output, if the input is changed.

Sample Cryptographic Hash Functions

Name	MD5	SHA-1	SHA-256	SHA-512	RIPEMD-160
n	128	160	256	512	160

Foundations of Hash Functions

Sample Cryptographic Hash Functions

Name	MD5	SHA-1	SHA-256	SHA-512	RIPEMD-160
n	128	160	256	512	160

```
1  $ echo 'Dear Angela, I give you 1 million EUR. Wolfgang' | sha1sum
2  9bf13969f2c283cfe0ace585667fa22c7ab4f84a -
3
4  $ echo 'Dear Angela, I give you 1 billion EUR. Wolfgang' | sha1sum
5  60d0b09f8d18e75b3cd7ffb0406de84bbc459510 -
```

Piecewise Hashing

Piecewise Hashing

Goals

1. Overcome drawbacks of cryptographic hash functions in the context of computer forensics.

Piecewise Hashing

Goals

1. Overcome drawbacks of cryptographic hash functions in the context of computer forensics.
2. Main drawbacks are:
   - Data acquisition: Integrity of copy is destroyed, if some bits change.

Piecewise Hashing

Goals

1. Overcome drawbacks of cryptographic hash functions in the context of computer forensics.
2. Main drawbacks are:
   - Data acquisition: Integrity of copy is destroyed, if some bits change.
   - White-/Blacklisting:
     - Suspect files similar to known to be bad files are not detected.
     - Fragments are not detected (due to deletion, fragmentation).

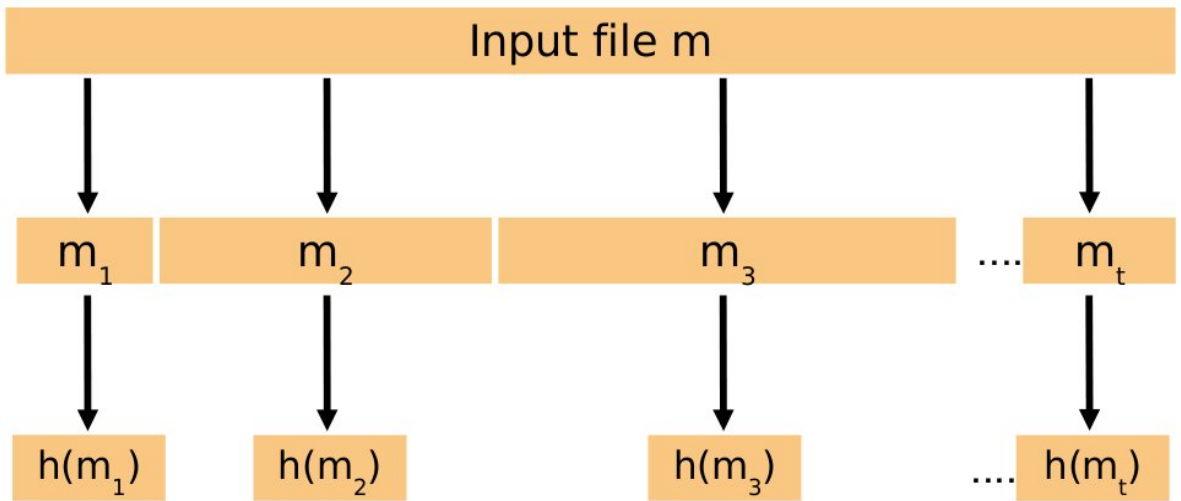
Piecewise Hashing

Goals

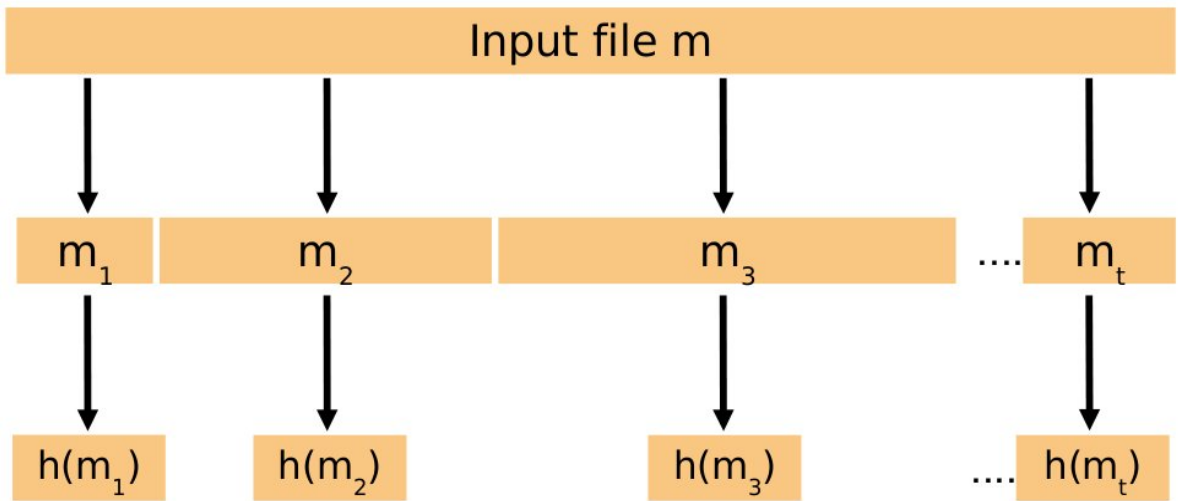
1. Overcome drawbacks of cryptographic hash functions in the context of computer forensics.
2. Main drawbacks are:
   - Data acquisition: Integrity of copy is destroyed, if some bits change.
   - White-/Blacklisting:
     - Suspect files similar to known to be bad files are not detected.
     - Fragments are not detected (due to deletion, fragmentation).
3. Currently known approaches:
   - Segment hashes (also called block hashes).
   - Context-triggered piecewise hashes.

Piecewise Hashing

Context Triggered Piecewise Hashes



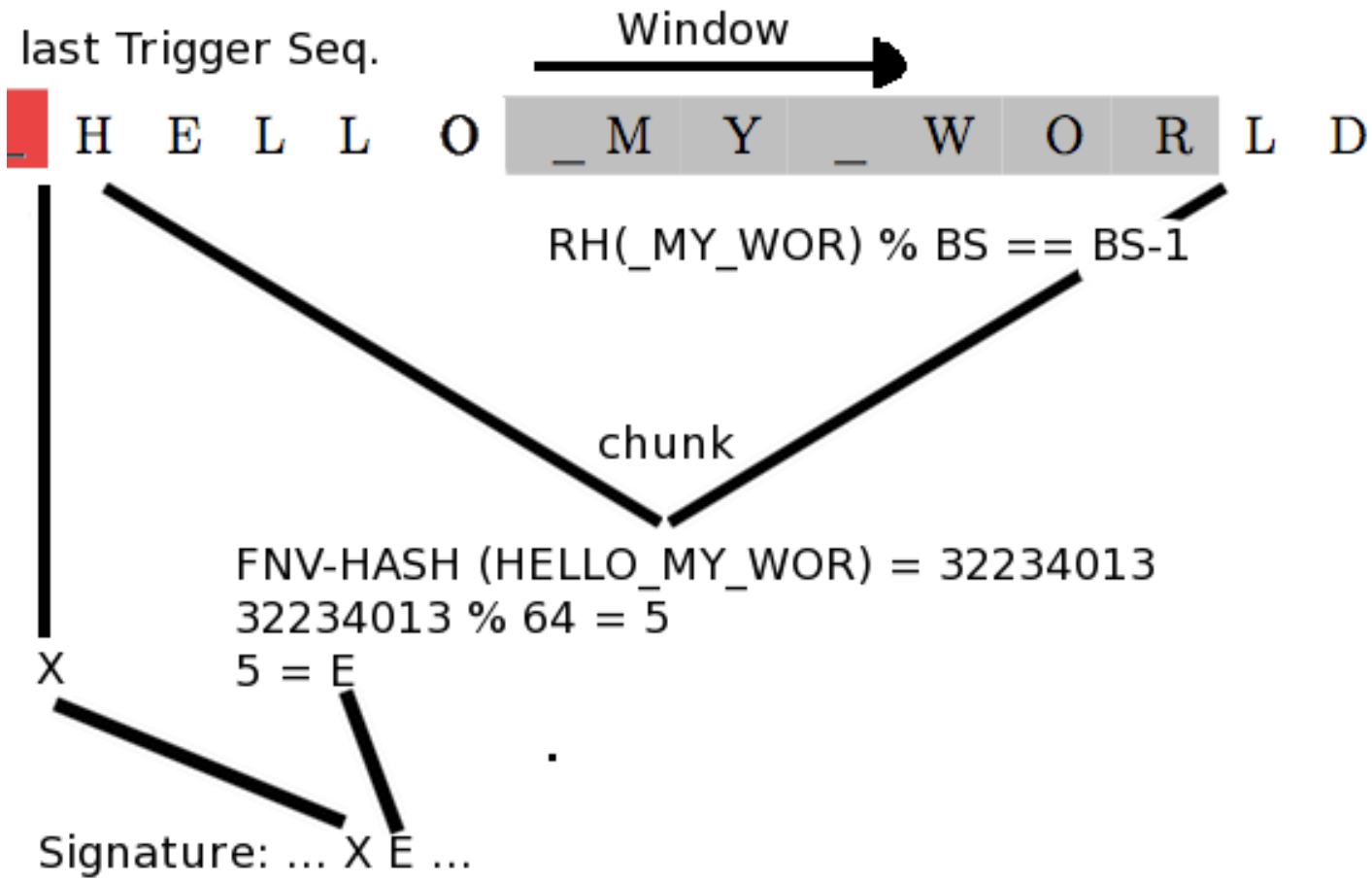
Context Triggered Piecewise Hashes



1. Originally proposed for spam detection (`spamsum` by Andrew Tridgell, 2002)
2. Ported to forensics by Jesse Kornblum, 2006: `ssdeep`.

Piecewise Hashing

CTPH: Workflow



Piecewise Hashes: The algorithm

Algorithm 1 CTPH(BS, PF, TV, h)

Input: A string of bytes BS of length N.
A pseudo random function $PF : \{0,1\}^{8w} \rightarrow \{0,1\}^{l}$.
A trigger value TV.
A hash function $h : \{0,1\}^{*} \rightarrow \{0,1\}^{n}$.

Output: The corresponding context triggered piecewise hash of the byte string BS.

$i \leftarrow 0; c \leftarrow 0;$
$CTPH \leftarrow$ ' '; // initialise CTPH as empty string
$B_{c-w+1} \leftarrow 0; B_{c-w+2} \leftarrow 0; \cdots; B_{c-1} \leftarrow 0;$ // initialise padding bytes
while $i < N$ **do**
  if $PF(B_{i-w+1}B_{i-w+2} \cdots B_i) = TV$ **then**
    $CTPH \leftarrow CTPH \parallel h(B_cB_{c+1} \cdots B_i);$
    $c \leftarrow i+1; B_{c-w+1} \leftarrow 0; B_{c-w+2} \leftarrow 0; \cdots; B_{c-1} \leftarrow 0;$
  $i \leftarrow i+1$
$CTPH \leftarrow CTPH \| h(B_c \| B_{c+1} \| \cdots \| B_{N-1});$
return $(CTPH);$

Kornblum's ssdeep tool

1. Window size: $w = 7$.
2. CTPH is a sequence of printable characters:
   - Least significant 6 bits (LS6B) of a segment hash.
   - LS6B are encoded Base64.

Kornblum's ssdeep tool

1. Window size: $w = 7$.
2. CTPH is a sequence of printable characters:
   - Least significant 6 bits (LS6B) of a segment hash.
   - LS6B are encoded Base64.
3. Number of segments S: $32 \leq S \leq 64$.

Kornblum's ssdeep tool

1. Window size: $w = 7$.
2. CTPH is a sequence of printable characters:
   - Least significant 6 bits (LS6B) of a segment hash.
   - LS6B are encoded Base64.
3. Number of segments S: $32 \leq S \leq 64$.
4. Block size b: Used to trigger the rolling hash PF
   - $PF(B_{i-w+1}B_{i-w+2}\cdots B_i) \equiv b - 1 \bmod b$.
   - $b \approx \frac{N}{S}$ (medium segment size)
   - $b_{init} = b_{min} \cdot 2^{\lceil \log_2(\frac{N}{S \cdot b_{min}}) \rceil}$
   - If S is too small for b_{init}, set $b \leftarrow \frac{b}{2}$.

Kornblum's ssdeep tool

1. Window size: $w = 7$.
2. CTPH is a sequence of printable characters:
   - Least significant 6 bits (LS6B) of a segment hash.
   - LS6B are encoded Base64.
3. Number of segments S: $32 \leq S \leq 64$.
4. Block size b: Used to trigger the rolling hash PF
   - $PF(B_{i-w+1}B_{i-w+2}\cdots B_i) \equiv b - 1 \bmod b$.
   - $b \approx \frac{N}{S}$ (medium segment size)
   - $b_{init} = b_{min} \cdot 2^{\lceil \log_2(\frac{N}{S \cdot b_{min}}) \rceil}$
   - If S is too small for b_{init}, set $b \leftarrow \frac{b}{2}$.
5. ssdeep outputs CTPH for both, b and $2b$.

Piecewise Hashing

Sample ssdeep output


```
1  $ ls -l hacker-siedlung.jpg
2  -rw------- 1 baier baier 78831 2011-05-09 10:16 hacker-siedlung.jpg
3
4  $ ssdeep -l hacker-siedlung.jpg
5  ssdeep,1.0--blocksize:hash:hash,filename
6  1536:ZfICsORJt2PazD7Z2xqHmqL36uuXtrHTXkkknIKB+W2pDHviF4eYySb:
       ZfICNRf2CD7YwGqL36FXVTXQnIWgDvi2,"hacker-siedlung.jpg"
```

Comparing CTPH

- Based on the weighted edit distance of both letter sequences.
- Count the number of insertions, deletions, changes, swaps.
- $e(s_1, s_2) = i + d + 3c + 5s$.
- Edit distance is scaled to a similarity percentage match M, $0\% \leq M \leq 100\%$ (e.g. $e(s_1, s_2) = 0 \mapsto 100\%$)

Comparing CTPH

- Based on the weighted edit distance of both letter sequences.
- Count the number of insertions, deletions, changes, swaps.
- $e(s_1, s_2) = i + d + 3c + 5s$.
- Edit distance is scaled to a similarity percentage match M, $0\% \leq M \leq 100\%$ (e.g. $e(s_1, s_2) = 0 \mapsto 100\%$)

```
1  $ dd if=hacker-siedlung.jpg of=start bs=1 count=50000
2
3  $ ssdeep -l -p hacker-siedlung.jpg start
4  hacker-siedlung.jpg matches start (82)
5
6
7  $ dd if=hacker-siedlung.jpg of=tail bs=1 skip=40000
8
9  $ ssdeep -l -p hacker-siedlung.jpg tail
10 hacker-siedlung.jpg matches tail (72)
```

Anti-Forensics on CTPH

Characteristics of Kornblum's Implementation

1. We exploit basic properties of `ssdeep`:
   - Only `ssdeep` signatures with the same block size or within a factor of 2 can be compared.

Characteristics of Kornblum's Implementation

1. We exploit basic properties of `ssdeep`:
   - Only `ssdeep` signatures with the same block size or within a factor of 2 can be compared.
   - A successful match needs at least one common substring in the signature of length 7.

Characteristics of Kornblum's Implementation

1. We exploit basic properties of `ssdeep`:
   - Only `ssdeep` signatures with the same block size or within a factor of 2 can be compared.
   - A successful match needs at least one common substring in the signature of length 7.
   - An `ssdeep` signature has at most 64 characters:
     - If 63 trigger points are already found, all further ones until `EOF` are ignored, i.e. the final trigger point is `EOF`.

Anti-Forensic Ideas for Anti-Blacklisting

1. Our anti-forensic aim is to circumvent a blacklist:
   - Modified incriminated files of the blacklist are not detected although perceptual identical to the original known-to-be-bad file.
   - Generate false negatives.

Anti-Forensic Ideas for Anti-Blacklisting

1. Our anti-forensic aim is to circumvent a blacklist:
   - Modified incriminated files of the blacklist are not detected although perceptual identical to the original known-to-be-bad file.
   - Generate false negatives.
2. Ideas:
   - Inflating a file.
   - Edit between trigger points.
   - Adding trigger sequences.

Anti-Forensic Ideas for Anti-Blacklisting

1. Our anti-forensic aim is to circumvent a blacklist:
   - Modified incriminated files of the blacklist are not detected although perceptual identical to the original known-to-be-bad file.
   - Generate false negatives.
2. Ideas:
   - Inflating a file.
   - Edit between trigger points.
   - Adding trigger sequences.
3. No semantic attacks like rotations, colour changes, ...

Inflating a File (1/2)

- Exploited property: *Only `ssdeep` signatures with the same block size or within a factor of 2 can be compared.*
- Attack idea: Increase block size by increasing the file size.

Inflating a File (1/2)

- Exploited property: *Only ssdeep signatures with the same block size or within a factor of 2 can be compared.*
- Attack idea: Increase block size by increasing the file size.

```
1  $ ls -l hacker-siedlung.jpg
2  -rw------- 1 baier baier 78831 2011-05-09 10:16 hacker-siedlung.jpg
3
4  $ dd if=/dev/urandom of=hacker-siedlung-inflated.jpg bs=1 count=300000
5
6  $ dd if=hacker-siedlung.jpg of=hacker-siedlung-inflated.jpg bs=1 conv=
       notrunc
7
8  $ ssdeep -l hacker-siedlung.jpg hacker-siedlung-inflated.jpg
9  ssdeep,1.0--blocksize:hash:hash,filename
10 1536:ZfICsORJt2PazD7Z2xqHmqL36uuXtrHTXkkknIKB+W2pDHviF4eYySb:
       ZfICNRf2CD7YwGqL36FXVTXQnIWgDvi2,"hacker-siedlung.jpg"
11 6144:ZACNp2CD7Yw3L3e4IWMmDJm29krRbUrfLjhf3zrfaiEj1dXJrgpiF57IjRdseF:
       VQC37jezmDg29kr1O3J3vfBiH77mX,"hacker-siedlung-inflated.jpg"
```

Inflating a File (2/2)

`hacker-siedlung.jpg`

`hacker-siedlung-inflated.jpg`

Edit between Trigger Points (1/4)

- Exploited property: *A successful match needs at least one common substring in the signature of length 7*.
- Attack idea: Change one (irrelevant) byte in every 7^{th} chunk.

Edit between Trigger Points (1/4)

- Exploited property: *A successful match needs at least one common substring in the signature of length 7*.
- Attack idea: Change one (irrelevant) byte in every 7^{th} chunk.
- Typical consequences:
  - File size remains.
  - Offset of trigger points does not change.
  - Every 7^{th} segment hash changes.
  - No match is computed using `ssdeep`.

Edit between Trigger Points (1/4)

- Exploited property: *A successful match needs at least one common substring in the signature of length 7*.
- Attack idea: Change one (irrelevant) byte in every 7^{th} chunk.
- Typical consequences:
  - File size remains.
  - Offset of trigger points does not change.
  - Every 7^{th} segment hash changes.
  - No match is computed using `ssdeep`.
- Easily applicable to `txt` or `bmp` files.
- Approach for `bmp` files: $B \leftarrow B \oplus 01$.

Edit between Trigger Points (2/4)

```
$ ./anti-bl-ebtp hacker_siedlung.bmp h 7 hacker_siedlung-manipulated.bmp
block size: 12288
7 -> TS: ->|0E 0E 11 0F 0F 0F 0F | -> change: 10 to 11
   chunk-size: 244fc  modify character offset: 244f5
14 -> TS: ->|25 21 20 23 1E 1D 24 | -> change: 20 to 21
   chunk-size: 9245  modify character offset: 2d73a
21 -> TS: ->|7A 78 78 7D 78 79 7B | -> change: 78 to 79
   chunk-size: 28dbe  modify character offset: 564f8
28 -> TS: ->|9C 9F 8E 9A 9E 93 9F | -> change: 8D to 8C
   chunk-size: 188b0  modify character offset: 6eda8
35 -> TS: ->|1F 23 1E 1F 22 1D 1E | -> change: 1E to 1F
   chunk-size: 230cb  modify character offset: 91e73
42 -> TS: ->|24 24 32 30 30 46 41 | -> change: 26 to 27
   chunk-size: 18236  modify character offset: aa0a9
49 -> TS: ->|25 18 21 24 1A 1F 20 | -> change: 23 to 22
   chunk-size: 24232  modify character offset: ce2db
56 -> TS: ->|DC DF EA E8 CA D7 D5 | -> change: DF to DE
   chunk-size: 15d47  modify character offset: e4022
58 Trigger Sequences
done
```

Edit between Trigger Points (3/4)

```
1  $ ls -l hacker_siedlung.bmp hacker_siedlung-manipulated.bmp
2  -rw-r--r-- 1 baier baier 959754 2011-01-12 13:29 hacker_siedlung.bmp
3  -rw------- 1 baier baier 959754 2011-05-09 11:59 hacker_siedlung-
       manipulated.bmp
4
5  $ ssdeep -l hacker_siedlung.bmp hacker_siedlung-manipulated.bmp
6  ssdeep,1.0--blocksize:hash:hash,filename
7  12288:GrxpWErqprJp3415jQd+PS9Q5oCitcoMNaBpeEvNEB7cQ0c6ePmbLikD+Jd:23
       qZYLQd+PMQKlioXvNEVEiJJd,"hacker_siedlung.bmp"
8  12288:GrxpWEYqprJp3715jQd+wS9Q5oCttcoMNa6peEvNEO7cQ0c61PmbLik8+Jd:2
       kqZbLQd+wMQKaiowvNEY5icJd,"hacker_siedlung-manipulated.bmp"
9
10 $ ssdeep -p hacker_siedlung.bmp hacker_siedlung-manipulated.bmp
```

Edit between Trigger Points (4/4)

`hacker-siedlung.bmp`

`hacker-siedlung-manipulated.bmp`

Adding Trigger Points (1/4)

- Exploited property: *An `ssdeep` signature has at most* 64 *characters:*
- Attack idea: Insert 63 trigger sequences in the file header.
  - Trigger sequences may be pre-computed (even *global trigger sequences*).
  - Trigger sequences may be computed on the fly.

Adding Trigger Points (1/4)

- Exploited property: *An `ssdeep` signature has at most* 64 *characters:*
- Attack idea: Insert 63 trigger sequences in the file header.
  - Trigger sequences may be pre-computed (even *global trigger sequences*).
  - Trigger sequences may be computed on the fly.
- Easily applicable for `jpg`, `pdf`, `doc`.
  - Make use of dedicated software to insert trigger sequences.
  - E.g. `jhead` for `jpg` files.

Adding Trigger Points (2/4)

```
$ ./generateTriggerSeqForFile hacker-siedlung.jpg j
1. Read file and generate Blocksize
2. Process file..
Modified: fake.jpg

$ ls -l hacker-siedlung.jpg fake.jpg
-rw------- 1 baier baier 79241 2011-05-09 14:17 fake.jpg
-rw------- 1 baier baier 78831 2011-05-09 14:16 hacker-siedlung.jpg

$ ssdeep -l hacker-siedlung.jpg fake.jpg
ssdeep,1.0--blocksize:hash:hash,filename
1536:ZfICsORJt2PazD7Z2xqHmqL36uuXtrHTXkkknIKB+W2pDHviF4eYySb:
     ZfICNRf2CD7YwGqL36FXVTXQnIWgDvi2,"hacker-siedlung.jpg"
1536:FG+OGOmu1VIAET7HSBZJhMfLbuuVV1wssHaq66y6pwwk0o7HDGlddwrHr/SfICsv:FG
     +OGEVGHNVV1mHaq66yewLHDvCfICNS,"fake.jpg"

$ ssdeep -p hacker-siedlung.jpg fake.jpg
```

Adding Trigger Points (3/4)

```
$ xxd hacker-siedlung.jpg | less

0000000: ffd8 ffe0 0010 4a46 4946 0001 0102 001c ......JFIF......
0000010: 001c 0000 ffdb 0043 0003 0202 0202 0203 .......C........
0000020: 0202 0203 0303 0304 0604 0404 0404 0806 ................
0000030: 0605 0609 080a 0a09 0809 090a 0c0f 0c0a ................
0000040: 0b0e 0b09 090d 110d 0e0f 1010 1110 0a0c ................
0000050: 1213 1210 130f 1010 10ff db00 4301 0303 ............C...
0000060: 0304 0304 0804 0408 100b 090b 1010 1010 ................
0000070: 1010 1010 1010 1010 1010 1010 1010 1010 ................
0000080: 1010 1010 1010 1010 1010 1010 1010 1010 ................
0000090: 1010 1010 1010 1010 1010 1010 1010 ffc0 ................
00000a0: 0011 0801 c902 bc03 0122 0002 1101 0311 ........."......
```

Adding Trigger Points (4/4)

```
$ xxd fake.jpg | less

0000000: ffd8 ffe0 0010 4a46 4946 0001 0101 012c ......JFIF.....,
0000010: 012c 0000 ffdb 0043 0003 0202 0202 0203 .,.....C........
0000020: 0202 0203 0303 0304 0604 0404 0404 0806 ................
[REMOVED]
0000090: 1010 1010 1010 1010 1010 1010 1010 fffe ................
00000a0: 0198 4141 4141 4158 2b41 4141 4142 4b78 ..AAAAAX+AAAABKx
00000b0: 4141 4141 424d 5a41 4141 4142 504d 4141 AAAABMZAAAABPMAA
00000c0: 4141 4256 3341 4141 4142 706d 4141 4141 AABV3AAAABpmAAAA
00000d0: 4277 6441 4141 4142 7946 4141 4141 4366 BwdAAAAByFAAAACf
00000e0: 6d41 4141 4143 6f46 4141 4141 4473 7441 mAAAACoFAAAADstA
00000f0: 4141 4144 7556 4141 4141 4478 4941 4141 AAADuVAAAADxIAAA
[REMOVED]
0000210: 4141 5375 4c41 4141 4154 797a 4141 4141 AASuLAAAATyzAAAA
0000220: 5543 4e41 4141 4155 4641 4141 4141 556f UCNAAAAUFAAAAAUo
0000230: 7a41 4141 4155 744f ffc0 0011 0801 c902 zAAAAUtO........
0000240: bc03 0122 0002 1101 0311 01ff c400 1d00 ..."............
```

Summary

Summary

- CTPH from Kornblum does not withstand an active adversary with respect to
  - blacklisting.
  - whitelisting.

Summary

- CTPH from Kornblum does not withstand an active adversary with respect to
  - blacklisting.
  - whitelisting.
- Doubtful if piecewise hashing can fulfill the expectations of fuzzy hashing:
  - Typically it is possible to flip one bit in each chunk.

Summary

- CTPH from Kornblum does not withstand an active adversary with respect to
  - blacklisting.
  - whitelisting.
- Doubtful if piecewise hashing can fulfill the expectations of fuzzy hashing:
  - Typically it is possible to flip one bit in each chunk.
- In order to create a viable new fuzzy hash function, it will be necessary to find different approaches.

Summary

Questions?

"Sorry about the odor. I have all my passwords tattooed between my toes."