Zum Einsatz von Hash-Funktionen in der Computer-Forensik:

Status Quo und Herausforderungen

Harald Baier

Hochschule Darmstadt, CASED

Fhg-SIT, 2011-04-12

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

Foundations of Hash Functions

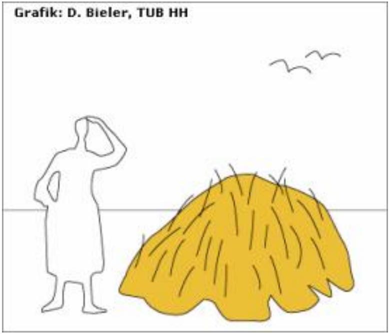
Use Cases of Hash Functions

Piecewise Hashing

Outlook

Motivation

Finding relevant files resembles ...

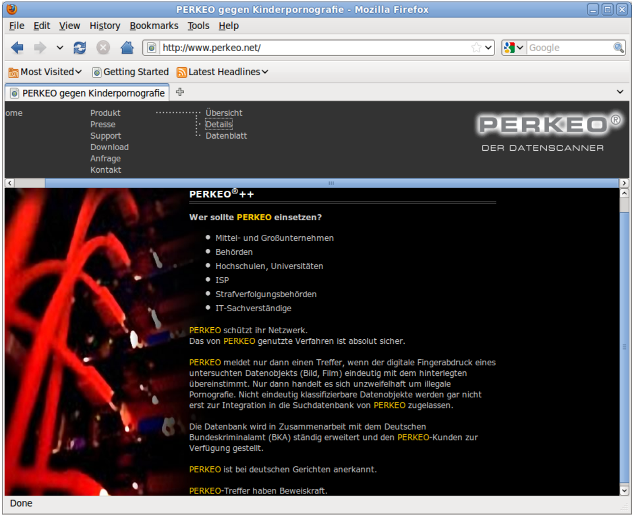


Source: tu-harburg.de

Source: beepworld.de

Motivation

... or is it solved for suspect files?



Definition and Avalanche Effect

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 volume, 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.
   - No control over the output, if the input is changed.
   - If only one bit in m is changed to get m', the two outputs $h(m)$ and $h(m')$ look 'very' different.

Sample Cryptographic Hash Functions

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

```
watson $ sha1sum vortrag_hash-in-forensics.pdf
83393d77d6f03de998c5ee1c2c9a2ad08f0901d2 vortrag_hash-in-forensics.pdf

watson $ sha1sum /dev/hda1
fba81604531ff5a26f1b2ab3a4674ab1d9dbf113 /dev/hda1

watson $ sha256sum /dev/hda
80ba7ddb431798591c1a6254de059e5734e5e4ab03e8a5185749fce6fde2de41 /dev/
    hda
```

Use Cases of Hash Functions

Use Cases of Hash Functions

Use Cases

1. Ensure authenticity and integrity during data acquisition.
   - Relevant for both dead and live analysis.
   - Hash values must be protected:
     - Written down by hand in investigation notebook.
     - Compute a digital signature over it.
2. Automatically identify known files:
   - Whitelisting: Known to be good files.
   - Blacklisting: Known to be bad files.

Relevant security property of the hash function:
Second-preimage resistance.

Whitelisting

1. Underlying Idea:
   - Generate a database G of known to be good files and their corresponding hash values.
   - Identify automatically an unsuspicious file on base of its hash value, which matches a fingerprint of a file in G.
   - Exclude a known to be good file from further investigation.
   - Significant reduction of irrelevant data.
2. Examples of unsuspicious files:
   - System files of operating systems.
   - Well-known benign applications like browsers, editors, ...
3. Widespread database:
   - Reference Data Set (RDS) of the National Software Reference Library (NSRL), maintained by NIST

NSRL-RDS: Sample Entries

```
watson $ less NSRLFile.txt

"SHA-1","MD5","CRC32","FileName","FileSize","ProductCode","OpSystemCode","SpecialCode"
"000000206738748EDD92C4E3D2E823896700F849","392126E756571EBF112CB1C1CDEDF926","EBD105A0","I05002T2.PFB",98
"0000004DA6391F7F5D2F7FCCF36CEBDA60C6EA02","0E53C14A3E48D94FF596A2824307B492","AA6A7B16","00br2026.gif",22
"000000A9E47BD385A0A3685AA12C2DB6FD727A20","176308F27DD52890F013A3FD80F92E51","D749B562","femvo523.wav",42
"00000142988AFA836117B1B572FAE4713F200567","9B3702B0E788C6D62996392FE3C9786A","05E566DF","J0180794.JPG",32
"00000142988AFA836117B1B572FAE4713F200567","9B3702B0E788C6D62996392FE3C9786A","05E566DF","J0180794.JPG",32
"00000142988AFA836117B1B572FAE4713F200567","9B3702B0E788C6D62996392FE3C9786A","05E566DF","J0180794.JPG",32
"00000142988AFA836117B1B572FAE4713F200567","9B3702B0E788C6D62996392FE3C9786A","05E566DF","J0180794.JPG",32
"00000142988AFA836117B1B572FAE4713F200567","9B3702B0E788C6D62996392FE3C9786A","05E566DF","J0180794.JPG",32
"00000142988AFA836117B1B572FAE4713F200567","9B3702B0E788C6D62996392FE3C9786A","05E566DF","J0180794.JPG",32
```

```
"SHA-1","MD5","CRC32","FileName","FileSize","ProductCode","OpSystemCode"
    ,"SpecialCode"

"00000142988AFA836117B1B572FAE4713F200567","9
    B3702B0E788C6D62996392FE3C9786A","05E566DF","J0180794.JPG"
    ,32768,2322,"WIN",""
```

Whitelisting: Assessment

1. General assessment:
   - Well-known and established process in computer forensics.
   - If database is trusted, no false positives (positive = benign).
2. Possible bottleneck: Size of database.
   - Size of database is increasing.
   - Currently RDS is about 6 gigabyte.

Blacklisting

1. Underlying idea:
   - Generate a database of known to be bad files and their corresponding hash values.
   - Let B denote this set.
   - Find automatically a suspicious file on base of its fingerprint, which matches a fingerprint of a file in B.
2. Sample suspect files:
   - Malware.
   - Encryption or steganographic software.
   - Corporate secrets.
   - IPR protected files.
   - Child pornography.

Blacklisting: Evaluation

1. Anti-detection approach:
   - Let a suspicious file $b \in B$ be given.
   - Change some (irrelevant) bit of b to get b'.
   - Consequence:
     - $h(b')$ is very different from $h(b)$.
     - b' is not detected automatically.
2. Core problem:
   - Cryptographic requirements of a hash function and forensic goals are complementary.
   - A suspicious file similar to an element of B is not detected.
3. Fragments of elements of B are not identified, too.

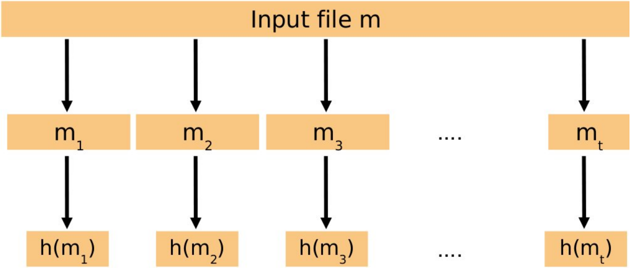
Piecewise Hashing

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.

Segment Hashes

1. Underlying idea:
   - Split input data (volume, file) in blocks of fixed length.
   - Compute for each segment its cryptographic hash.
   - Lookup in hash database for matches.

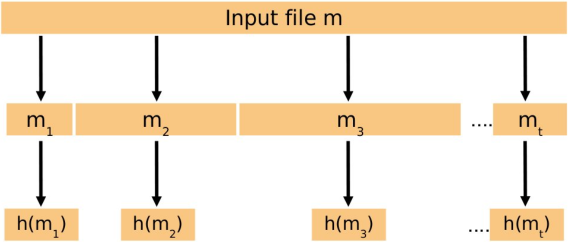


2. Original aim: Improve integrity of storage media.

Segment Hashes: Evaluation

1. Anti-Blacklisting is very easy:
   - Introduce / Delete an irrelevant byte in the first sector.
   - All segment hashes differ from the stored segment hashes.
   - Modified suspect file is not detected.
2. A good technique for whitelisting (see NIST results).
3. Size of segment hash database is large:
   - 4096 byte block size, SHA-1.
   - $\frac{\text{size of hash database}}{\text{size of raw data}} = \frac{20}{4096} = 0.00488$
     $\Longrightarrow$ 1 terabyte of raw data yields a 5 gigabyte hash database.
4. Hash database depends on the hashwindow size.

Context Triggered Piecewise Hashes



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

CTPH: A sample tool

1. ssdeep (based on spamsum).
2. CTPH is a sequence of printable characters:
   - Only the least significant 6 bits (LS6B) of a segment hash are considered.
   - LS6B are encoded base64.

```
watson $ ssdeep -l vortrag_hash-in-forensics_sit-110412.pdf

ssdeep,1.0--blocksize:hash:hash,filename
12288:UweC9h947a4LMqsMSO/6tzDEPU6P8Ohu7B9N9Fi:HD9/0MjI6aPU6kk69i,"
    vortrag_hash-in-forensics_sit-110412.pdf"
```

3. A good tool in absence of an active adversary.
4. FTK implements CTPH.

Outlook

Central Challenges

1. In the short term:
   - Determine a 'compression' ratio for whitelisting.
   - How successful is block hashing?
   - Process model of using CTPH and semantic layer similarity tools.
2. In the long term: Find a similarity preserving hash function.
   - Fuzzy hash function, denoted by f.
   - m and m' are 'similar' $\Longrightarrow f(m)$ and $f(m')$ are 'similar', too.
   - m shall be of any type: txt, doc, odt, jpg, bmp, devices, ...

Thank you for your attention!

1. Harald Baier
2. E-Mail:
    - harald.baier@h-da.de
    - harald.baier@cased.de

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