Privacy and Robust Hashes Privacy-Preserving Forensics for Image Re-Identification

Martin Steinebach*, Sebastian Lutz and Huajian Liu

Fraunhofer SIT, Darmstadt, Germany
E-mail: steinebach@sit.fraunhofer.de
**Corresponding Author*

Received 28 November 2019; Accepted 28 November 2019;
Publication 23 January 2020

Abstract

Within a forensic examination of a computer for illegal image content, robust hashing can be used to detect images even after they have been altered. Here the perceptible properties of an image are used to create the hash values. Whether an image has the same content is determined by a distance function. Cryptographic hash functions, on the other hand, create a unique bit-sensitive value. With these, no similarity measurement is possible, since only with exact agreement a picture is found. A minimal change in the image results in a completely different cryptographic hash value. However, the robust hashes have an big disadvantage: hash values can reveal something about the structure of the picture. This results in a data protection leak. The advantage of a cryptographic hash function is in turn that its values do not allow any conclusions about the structure of an image. The aim of this work is to develop a procedure for which combines the advantages of both hashing functions.

Keywords: Privacy, robust hashing, hashing, fingerprinting, forensics.

Journal of Cyber Security and Mobility, Vol. 9_1, 111–140.
doi: 10.13052/jcsm2245-1439.914

1 Introduction

Today one sees a rapid increase of digital images available on the Internet and also stored on devices like PCs or smartphones. This results in large collections to be sighted during forensic investigations, for example during an investigation regarding the possession of child pornography.

If there is a suspicion of illegal possession of such images, an investigation of storage devices is carried out. If large amounts of data are to be examined, a manual search of a suspect's computer or the classification of the stored images is very time-consuming and morally questionable. The privacy of the person concerned or other users sharing the same devices (primarily family) is no longer protected. This problem is avoided by using automated techniques that are efficient and precise. A common approach is cryptographic hashing. Programs such as PERKEO++ (Program for the Recognition of Relevant Child Pornographic Unique Objects) are used, which quickly and easily check files on a data carrier (similar to a virus scanner) using cryptographic hash values and provide information as to whether suspicious content has been found. However, cryptographic hash methods can only locate exact copies of a file. This is particularly problematic in the case of a digital image. Saving the file again under lossy JPEG compression will produce an image with a different cryptographic hash. The result is two images with contents identical to a human observer, which have a different hash value and are thus incorrectly classified as different images.

An alternative or an solution further variant is the robust hash, which does not compare bitwise for exact files, but saves an abstraction of the perceptible image content as a hash value in order to make it comparable. This provides a good robustness against changes to the image not altering its actual content. Figure 1 illustrates this process. But a drawback which is sometimes raised is that as the robust hashes store information describing the content of the image, the hashes may also tell something about the content of a hashed image. This could become a privacy risk in some scenarios.

In a forensic investigation, an investigator could be assigned to verify the possession of so called 'revenge-porn' on a computer of a suspect. The original content would come from the person shown on the images and also accusing the suspect of possession and distribution. This person would prefer to ensure minimal leakage of the compromising photos. In the case of a false accusation, the investigator should learn nothing about the content of the photos. If the accusation is true, the investigator will find the copies of the photos, therefore having access to the content. So from a security protocol point of view, the relevant challenge is to prevent the investigator from learning anything about the content of the images in the case of a false accusation.

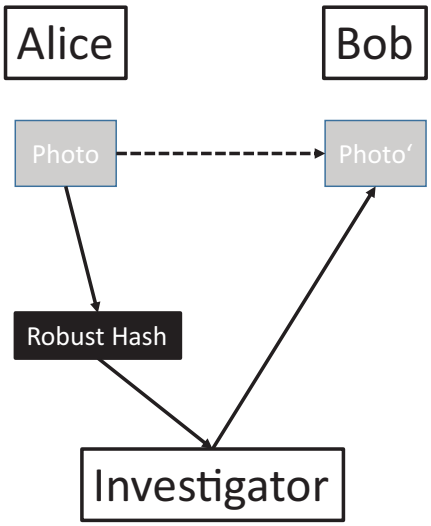

Figure 1 Bob steals a compromising photo of Alice e.g. from her camera. He stores a smaller copy of it on his smartphone. Alice accuses Bob of doing so and provides a robust hash of the original photo to an investigator who searches Bob's devices for a photo matching to the robust hash.

2 Related work

In this section we address a number of areas relevant for our work. As the subject of our research is the handling of digital photos in forensic investigation, we provide a brief discussion of the different methods of relevance. We also mention the current state in privacy-preserving forensics. As robust hashing is the base of our approach, we give a short introduction and also briefly describe the block hash used in this work.

The security of robust or perceptual hashing which respect to aspects beyond privacy is not within the scope of this work. This has been discussed in the literature in e.g. [12] with respect to authentication by robust hashing. Also not within the scope of this work is the interaction of robust hashes and secret keys in protocols for integrity verification as in [22]. Using a robust hash as part of a digital signature by replacing the cryptographic hash in the protocol by a perceptual hash and signing it has already been suggested in 1996 [17]. All these approaches use a hash as a method of robust authentication but do not address the risk of leaking information about the hashed content.

2.1 Tasks of Image Handling

In forensics, there are several tasks which address the handling of photos or similar images. On a top level, this can be divided into two groups:

2.1.1 Re-Identification

Here the image is assumed to be already known, e.g. from a earlier investigation. It is to be re-identified and matched to some information about it stored in a database. This database can be a black or a white list. A black list will raise an alarm for an investigator telling him he has found relevant evidence. A white list will remove the image from any deeper investigation as the image is known to be of no relevance. As an example for this task, we have described a black list concept for identification of child porn based on a block hash in [20]. This strategy is used today by many users worldwide.

Images can be either identified by cryptographic or robust hashes. It needs to be stressed that this task is about re-identifying the images, not the persons shown on it as discussed in e.g. [6].

2.1.2 Classification

In this case [9] it is not assumed that the image is already known. To help the investigator with sighting a huge amount of images, meta-data is automatically generated, for example by matching the image to reference images with similar features. Thereby the investigator can browse through a selected set of images with relevant features and ignore the rest. For image classification, trained deep learning nets are known to provide the best results [4]. There are multiple general nets available allowing automated tagging or annotation of images. Our work based on deep learning [14] follows the concept of classification-based sorting of image sets. The network does not decide whether an user needs to view a given image but rather estimates its relevance and sorts the images by it. Therefore the user is more likely to quickly find relevant material.

In this work, we address the challenge of re-identification of images with the help of robust and cryptographic hashes.

2.2 Privacy and Forensics

The idea to execute a forensic investigation in a privacy-preserving manner is not new. In this section we provide a brief overview on approaches discussed in the literature.

Srinivasan et al. [18] describe various policies for preserving privacy during an investigation. They do not focus on technical solutions, but rather on correct behavior of investigators and acceptance of evidence by the court. Adams [1] discusses the requirements of a forensics tool to be compliant with the laws of the United States. One aspect addressed is logging the actions of investigators, allowing to trace privacy breaches.

Hou et al. [8] present a technique is presented to search encrypted data for multiple search words. In the scenario presented, there are two roles. An investigator who performs an investigation and only has access to relevant data and an administrator who manages the data. A similar approach is taken by Armknecht and Dewald [2]. Here a third party is investigating emails of a company. All emails are encrypted, and only if a sufficient number of keywords within the emails are found, the full text of these individual emails can be decrypted.

Various research also addresses risks to privacy and the role of technology as potential evidence. Stahlberg et al. [19] explore the role of databases in the context.

Peter et al. introduced the need for improved privacy in handling images during forensic investigations in their discussion [16] of a 'Privacy-Preserving Architecture for Forensic Image Recognition'. Their approach was to derive cryptographic keys from the images to be sighted and protect sensitive elements of the images by these keys.

We also contributed to the domain of privacy-preserving forensics by introducing a protocol based on partial encryption, face recognition and key exchange [13]. Here during an investigation faces in images are recognized and encrypted. Only if a scene is of relevance for an investigator, he can ask for decryption keys provided by a third party.

2.3 Cryptographic Hash Functions

Cryptographic hash functions (see e.g. [10]) are a common primitive of security protocols with many applications. We only describe them briefly as a reminder here. They calculate hash values of fixed length from information of arbitrary length. They must meet a number of requirements including:

- Efficiency: They must be efficient to calculate
- Collision-Resistant: It must be extremely unlikely to find two pieces of information that have the same hash value
- One-way function: It must be practically impossible to find the information associated with a hash value.

Among the best known one-way hash functions are MD5 and the Secure Hash Algorithm (SHA).

2.4 Robust Hash Functions

Several robust or perceptual hashes for various media types are known, which provide different levels of robustness. For example, Roover et al. [3] provide an image hash algorithm which is robust against geometrical operations like scaling and rotation; the hash draws its robustness from the use of the Radon transform. Friedrich and Goljan propose an approach based on random noise similarity in [5]. As there are too many algorithms to mention here, we recommend surveys like the one by Haouzia et al. [7] or Neemila and Singh [15]. There are also methods for audio and video streams as well as text data.

Robust hashing extracts perceptually relevant features from multimedia content for identification purposes. They must meet a number of requirements. The most important are:

- Distinction: Perceptually different pieces of media data shall have different hash values.
- Robustness: The robust hash values shall show a certain degree of perceptual invariance, i.e. two pieces of media data that are perceptually similar for an average viewer/listener shall be similar, too.
- Security: The features must survive attacks that are directly aimed at the feature extraction and consecutive processing steps. Similar to cryptographic hash functions, the robust hash values shall be equally distributed among all possible pieces of media data and pairwise statistically independent for two pieces of media data that are perceptually different.

The robustness brings the risk of leakage. When two images are very similar, their hashes are also similar. Distinction goes only so far that two similar images will not have an identical hash, but both hashes will be more similar than the hashes of two images with different content. As an example: Portrait photos with a human face in the center and a light plain background all share a similar robust hash structure. This leads to false positives in robust hash function higher than expected given the theoretic number space spanned by a hash. As an example, Steinebach et al. show this in [21] with two landscape photos and motivate their own countermeasure of using sub-hashes of image areas.

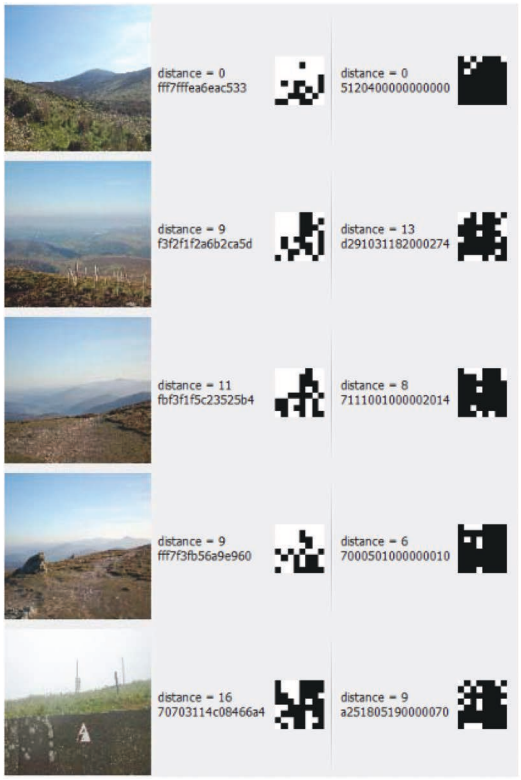

Figure 2 Similar images have similar robust block-based hashes. The mid column shows a differential block hash, the right column the block hash from the phash library.

We have verified this by comparing a landscape photo with other photos and different robust block hash algorithms. Figure 2 shows that the hamming distances between the top image and the other images are small enough to be confused for the original image by common thresholds between 16 and 32 bits for re-identification. If we have the robust hash of the first image but do not know the image, we can compare the hash to hashes of images we know and will learn which of our images are likely to be similar.

Figure 3 shows four other examples created with a simple blockhash implementation to further illustrate this. Here the images are placed next to their low res 16×16 grey scale representation and the resulting binary block hash. The similar structures lead to similar hashes.

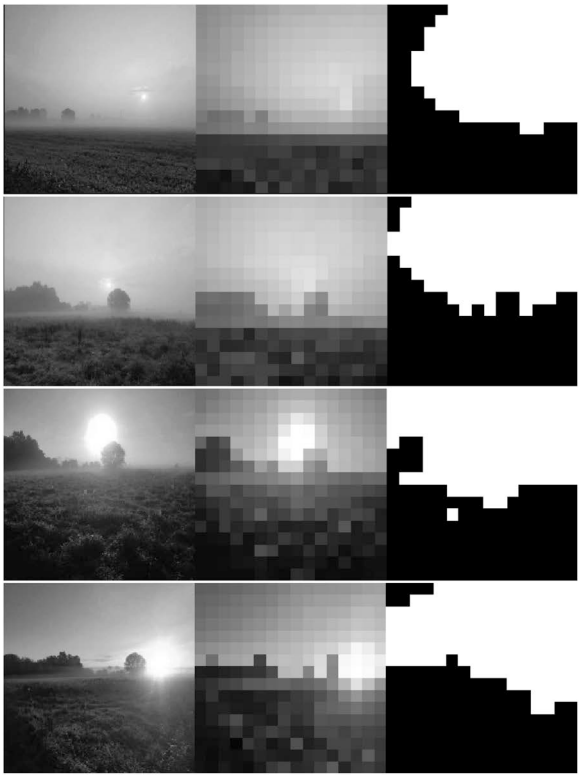

Figure 3 Similar images have similar robust block-based hashes. The mid column shows the small grey scale version of the image, the right column the block hash.

2.5 Block Hash

In 2006, Bian Yang et al. proposed a block mean value based perceptual image hash function [23]. Four slightly different methods are proposed. The latter two additionally incorporate an image rotation operation to enhance robustness against rotation attacks. This increases the computational complexity of the latter two methods. Here we focus on the simplest method:

- Convert the image to grey scale and normalize the original image into a preset size.
- Let N denote the bit length (usually 256 bit) of the final hash value. Divide the pixels of the image I into non-overlapped blocks $I_1, I_2, \ldots, I_N$.

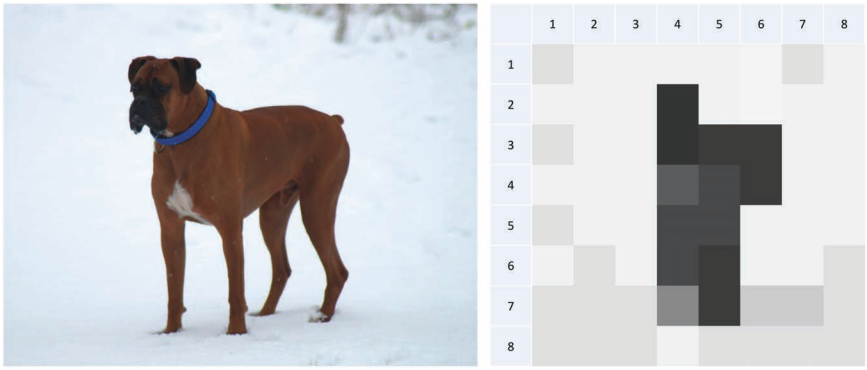

Figure 4 Left: Image. Right: Gray scale 8×8 pixel version.

- Calculate the mean of the pixel values of each block. That is, calculate the mean value sequence $M_1, M_2, \ldots, M_N$ from the corresponding block sequence. Finally obtain the median value M_d of the mean value sequence.
- Normalize the mean value sequence into a binary form and obtain the hash value h(i) as 0 if $M_i < M_d$ or 1 if $M_i >= M_d$.

In 2012, Steinebach et al. suggested a number of improvements to the approach [21]. One is based on the observation that the individual hash bits are not equally robust or stable. Depending on their distance to the median value, they are more likely to skip. Hash bits with a small difference between M_x and M_d are less robust than those with a large difference.

As this finding is the base for our approach to combine robust and cryptographic hashes, we illustrate this with a simplified example. In Figure 4 we show an example of a photo and the downscaled gray scale version of it. We only use 8×8 pixel in the example for a more simple visualization while the actual size is 16×16 pixel. In Figure 5 the values (we use a range from 1 (white) to 100 (black) here) of the gray scale pixels are shown. From these values the median is calculated, which is 15. We can now calculate the difference of the values from the median and see a high variance between 0 and 80. For calculating a binary hash, one would map all values above or equal the median to a bit value of 1 and all smaller to bit value 0.

3 Concept

Our aim is to combine the robustness of robust hashes with the privacy given by the one-way nature of cryptographic hashes. This could be done straightforward: a robust hash strategy usually allows a given number of bits

	1	2	3	4	5	6	7	8
1	20	10	10	10	10	10	20	10
2	15	10	10	95	10	8	10	10
3	20	10	10	80	75	80	10	10
4	15	10	8	45	75	78	10	10
5	20	10	8	75	75	8	10	10
6	15	20	10	75	80	8	10	10
7	20	20	20	40	70	30	30	20
8	20	20	20	10	20	20	20	20

	1	2	3	4	5	6	7	8
1	5	-5	-5	-5	-5	-5	5	-5
2	0	-5	-5	80	-5	-7	-5	-5
3	5	-5	-5	65	60	65	-5	-5
4	0	-5	-7	30	60	63	-5	-5
5	5	-5	-7	60	60	-7	-5	-5
6	0	5	-5	60	65	-7	-5	-5
7	5	5	5	25	55	15	15	5
8	5	5	5	-5	5	5	5	5

Figure 5 Left: Values from Figure 4 (right). Right: Distance from median 15.

to be different from the original hash and still identify both hashes as equal. Commonly this is calculated by a hamming distance between both hashes. A threshold n is used as the maximal hamming distance accepted. So in theory we could derive all versions of a robust hash with a size of m bit where up to n bit differ and calculate a cryptographic hash of each version. Thereby we would have a set of cryptographic hashes which represent each potential copy of the robust hash deemed equal to the original hash. But this would result in a vast amount of hashes: a common robust block hash has a size of 256 bit and the typical accepted hamming distance is 32. So the number of cryptographic hashes would be $\binom{256}{32}$ or 5.824459×10^{40}.

But from research on the block hash [21] we know that we can make assumptions about the positions in the robust hashes where changes are most likely. Table 1 shows a small robust hash of 10 bit calculated from values between 0 and 1. In the row 'Dist. M.' we see that the differences of the hash bits from the median vary strongly. Those bits with the smallest difference are the least stable bits most likely to change. In the example these are bits 4 and 7. Figure 7 illustrates the behavior of the hash bits with respect to value and median-based decision.

Now if we need to guess which bits will change, we would guess bit 4 and bit 7. So if we wanted to generate a set of cryptographic hashes from the robust hash which represent a robustness equal to an allowed hamming distance of 2 from the original hash, we would create versions of the robust hashes where these 2 would flip. The rows Com_1 to Com_4 show the 4 variations resulting of this idea. So instead of $\binom{10}{2}$ or 45 hashes we only calculate 4 hashes. Of course this comes with a price: while the bits 4 and 7 are the most

Table 1 Conceptual example of strategies for a set of 10 value with median 0.37. HashBit shows the regular hash. Dist. M. is the difference of the actual value to the median. 0-neu. is the hash calculated by the 0-neutral strategy. X-neu. shows the X-neutral strategy. The lines Com_1 to Com_4 show the first 4 variations of the hash starting from the lowest median difference

Bit#	1	2	3	4	5	6	7	8	9	10
Value	0.61	0.28	0.25	0.33	0.51	0.26	0.41	0.31	0.77	0.68
HashBit	1	0	0	0	1	0	1	0	1	1
Dist.M.	0.24	0.10	0.12	0.04	0.13	0.12	0.04	0.07	0.40	0.30
0-neu.	1	0	0	0	1	0	0	0	1	1
X-neu.	1	0	0	X	1	0	X	0	1	1
Com_1	1	0	0	0	1	0	0	0	1	1
Com_2	1	0	0	0	1	0	1	0	1	1
Com_3	1	0	0	1	1	0	0	0	1	1
Com_4	1	0	0	1	1	0	1	0	1	1

likely to change if the image the hash is calculated from is changed, there is no guarantee that in some cases e.g. bit 1 would change but bit 4 and 7 stay unchanged. This would result in a hamming distance of 1 to the original hash. The original robust hash would correctly see this as an equal copy. The set of 4 cryptographic hashes would not include this new hash. Robustness with this strategy is therefore robustness with a specified threshold. This threshold also controls the number of cryptographic hashes to be calculated. It therefore rules the compromise between robustness and hash quantity.

There is also an alternative to building a set of hashes based on a threshold: bits likely to flip can be set to a 'neutral' position. A distance lower than the threshold would change a hash bit to a neutral bit. This neutral bit can either be one of the two values '0' and '1' or it can be an additional symbol like 'X'. The lines '0-neu.' and 'x-neu.' illustrate this strategy. The advantage is limiting the number of cryptographic hashes to one single hash. The drawback is that the distance of the values to the threshold can change during operations on the hash. This is an additional source of errors leading to a cryptographic hash not matching the hash of the original image. In Figure 6 we illustrate the steps of our approach.

3.1 Phases

From a more formal perspective, the suggested process can be structures as follows based on the block hash from Section 2.5:

1. Computation of hash block values $M_1, M2, \ldots, M_N$ and median M_d
2. Computation of robust block hash $H_1, H_2, \ldots, H_N$

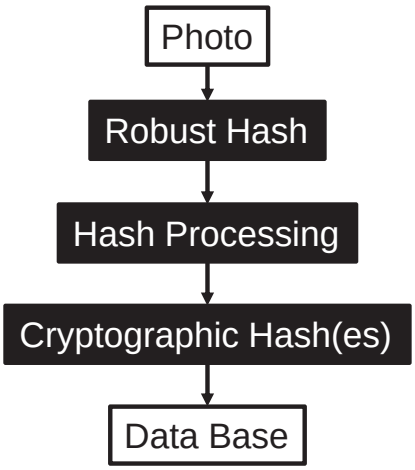

Figure 6 The hybrid concept: From a photo, a robust hash is derived. This hash is then processed and one or more cryptographic hashes are calculated from it.

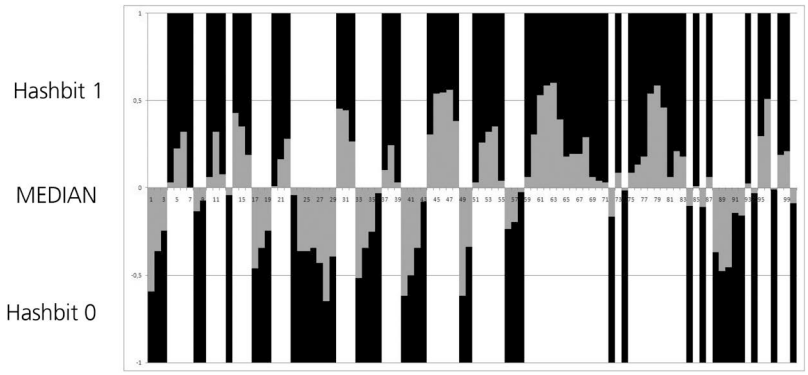

Figure 7 The robust hash is calculated by a threshold based on the median of the pixel values. Depending on the distance between median and values, decisions are more or less stable.

3. Computation of differences $D_1, D_2, \ldots, D_N$ by $D_i = |M_i - M_d|$
4. Identification of weak bits $W_1, W_2, \ldots, W_N$ based on a threshold t by $W_i = 1$ if $D_i <= t$ else $W_i = 0$
5. Creation of robust hash variant(s) based on (2) and (4)
6. Computation of cryptographic hash(es)

Figure 8 Original image.

The phases only differ in (5) and (6) depending on the strategies.

- Combinations: Here in (4) all variations of the original hash H are calculated based on the weak bits W and one cryptographic hash is computed for each version.
- 0-neutral: Here in (4) hash bits of H_i are set to '0' when $W_i = 1$. Only one cryptographic hash is computed.
- 0-neutral: Here in (4) hash bits of H_i are set to 'X' when $W_i = 1$. Here also only one cryptographic hash is computed.

3.2 Blockhash Behavior

For a deeper analysis of the behavior of the block hash we discuss its changes due to lossy compression in detail in this section. As an example image, we use the image of a horse as shown in Figure 8. The Figures 9 and 10 show the result of the blockhash calculation on the original image and a copy which has been stored as a quality factor 50 JPEG file, therefore subject to medium lossy compression. While the overall structure of the hashes seems to be identical, some positions differ from each other. Figure 11 gives the actual positions where both hashes differ: L2, K6, G9, J10 are the first of twelve such positions.

Now we look at the pixel values of the downsized 16×16 version of image 8 shown in Figure 12. The median of these 256 values is 135.

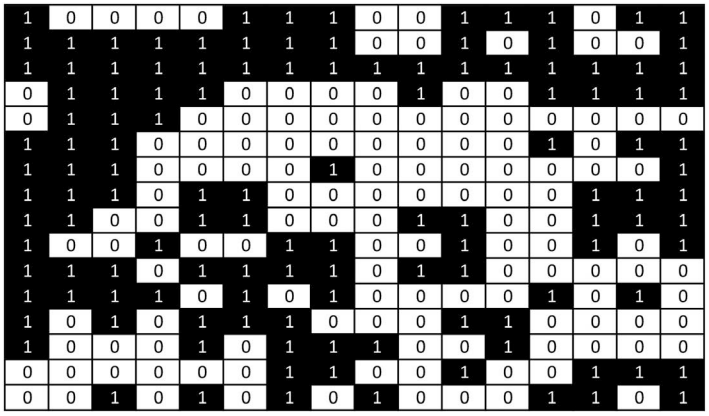

Figure 9 Hash of original image.

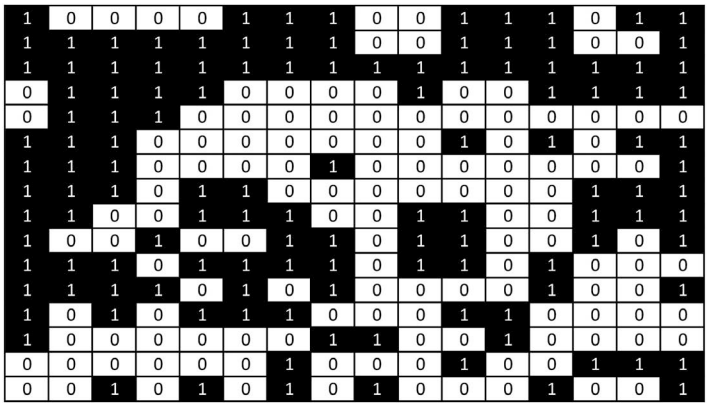

Figure 10 Hash of image compressed by JPEG quality factor 50.

In Figure 13 we subtract the median from the pixel values and see values between -118 and 85. Now we can observe two things: The positions where a hash change has occurred all have low values: $\mathrm{L2} = -4$, $\mathrm{K6} = -4$, $\mathrm{G9} = -1$, $\mathrm{J10} = -1$ and so on. But not all positions with small values show hash changes. As examples, positions $H1 = 1$ and $N1 = -2$ but the hash values remain the same.

Therefore one can estimate where hash changes occur; small distances from the median obviously lead to weaker hash values with a higher likelihood to skip. But not all small distances will flip. Without additional hints where changes will occur due to image structure and impact of

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 11 Difference of both hashes shown above. Row and column numbers have been added for easy reference.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	213	111	65	71	112	186	203	136	126	95	148	166	153	133	187	172
2	215	209	198	171	144	172	196	157	129	104	154	131	185	70	131	141
3	175	189	184	203	200	197	181	207	201	197	204	192	205	206	199	204
4	102	190	181	211	181	112	22	92	87	198	91	46	178	183	213	185
5	69	202	189	174	49	50	65	87	76	64	106	52	123	100	117	112
6	178	172	182	56	65	62	64	83	73	37	131	50	141	49	161	192
7	200	216	183	88	17	23	79	149	34	47	97	25	97	73	123	173
8	166	220	173	86	157	158	28	40	31	24	37	31	119	188	178	207
9	177	155	81	91	158	172	134	34	27	162	179	26	97	172	200	182
10	157	131	121	155	110	123	145	148	19	134	145	131	37	149	128	145
11	165	140	143	114	150	163	204	141	50	142	151	125	134	63	128	127
12	153	144	164	143	110	168	135	157	26	112	84	35	174	113	139	133
13	182	128	191	115	164	162	166	135	109	110	151	141	131	49	110	111
14	160	128	92	102	142	135	139	154	148	51	124	172	120	119	93	111
15	92	55	94	39	40	31	217	160	122	63	186	133	99	149	169	148
16	93	132	167	123	164	93	158	102	149	133	126	69	143	147	77	144

Figure 12 Pixel values of image used to calculate the hash from Figure 9.

compression algorithms, our strategy can only assume that all small values up to a given threshold will skip.

A quick look at the image used as an example shows that the task to guess the correct hash positions where changes will occur. In Figure 14 we show the original image and the 16×16 grey scale image derived from it which is the base of the block hash calculation. It is a visualization of the matrix given in Figure 12. Figure 15 shows the first four positions where hash flipping occurred by a modifying the color channel values for the respective hash

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	78	-24	-70	-64	-23	51	68	1	-9	-40	13	31	18	-2	52	37
2	80	74	63	36	9	37	61	22	-6	-31	19	-4	50	-65	-4	6
3	40	54	49	68	65	62	46	72	66	62	69	57	70	71	64	69
4	-33	55	46	76	46	-23	-113	-43	-48	63	-44	-89	43	48	78	50
5	-66	67	54	39	-86	-85	-70	-48	-59	-71	-29	-83	-12	-35	-18	-23
6	43	37	47	-79	-70	-73	-71	-52	-62	-98	-4	-85	6	-86	26	57
7	65	81	48	-47	-118	-112	-56	14	-101	-88	-38	-110	-38	-62	-12	38
8	31	85	38	-49	22	23	-107	-95	-104	-111	-98	-104	-16	53	43	72
9	42	20	-54	-44	23	37	-1	-101	-108	27	44	-109	-38	37	65	47
10	22	-4	-14	20	-25	-12	10	13	-116	-1	10	-4	-98	14	-7	10
11	30	5	8	-21	15	28	69	6	-85	7	16	-10	-1	-72	-7	-8
12	18	9	29	8	-25	33	0	22	-109	-23	-51	-100	39	-22	4	-2
13	47	-7	56	-20	29	27	31	0	-26	-25	16	6	-4	-86	-25	-24
14	25	-7	-43	-33	7	0	4	19	13	-84	-11	37	-15	-16	-42	-24
15	-43	-80	-41	-96	-95	-104	82	25	-13	-72	51	-2	-36	14	34	13
16	-42	-3	32	-12	29	-42	23	-33	14	-2	-9	-66	8	12	-58	9

Figure 13 Distance of pixel values from median 135.

Figure 14 The image and its grey scale 16×16 hash calculation base.

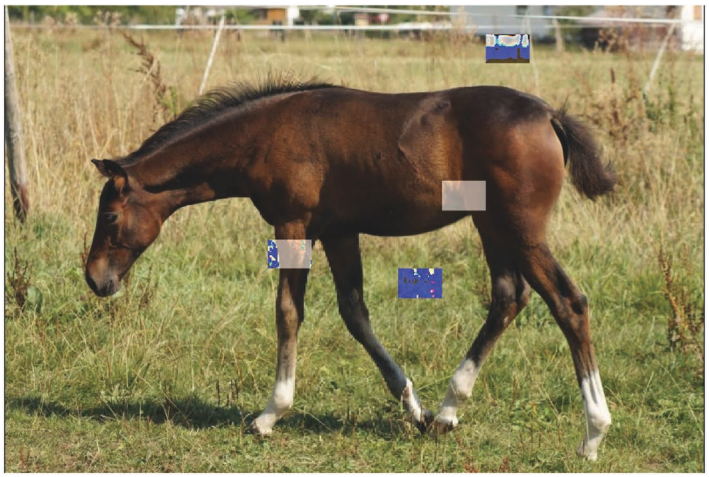

Figure 15 Four positions where hash bit flipping occurred.

block areas. Among the flipping positions are grass areas, one patch of fur and one leg position with grass in the background. These are plain areas as well as textured ones, and areas with and without strong changes in color and lightness.

4 Implementation

The implementation of the concept we describe above is straightforward. We briefly describe the individual steps in the following section.

4.1 Robust Hash

For the robust hash function, we use our own implementation of the block hash as described in 2.5. The only important difference to standard implementations like phash [11] is that not only the binary hash vector is exported by the hash function, but also the distance D_i of each block value from the median.

4.2 Hash Processing

The processing of weak bits is a central point in the concept of this work. In order to generate suitable cryptographic hashes from the robust hash values, it is not sufficient to calculate the cryptographic hash value for a given robust hash of an image. To locate weak bits, the first step is to use a parameter to distinguish the bits into strong and weak bits via their distance form the median D_i. The method checks which bits have a distance smaller a given threshold and stores the positions of the respective bits.

Then depending on the three strategies, the identified weak bits are either replace by their neutral values '0' or 'X' or the set of combinations of individual robust hash variants is calculated.

4.3 Cryptographic Hash

We need a cryptographic hash function that hashes the preceding robust hashes. Thereby we generate a unique value that can be directly compared in the database without further calculations. The cryptographic hash algorithm should generate hash values that correspond approximately to the length of the robust hash, namely 256 bits. No hash function using a secret key is required. The so-called avalanche effect is supposed to occur. This means that if only one bit is changed, a completely different hash value is output.

This means that an attack cannot estimate the original robust hash from the cryptographic hash with reasonable effort. In addition, the hash function should be easily calculable, a one-way function and collision-resistant. Since the SHA-2 meets all requirements, SHA-256 is used as the cryptographic hash algorithm in this paper.

4.4 Hash Comparison

Re-identifying an image is done by searching a hash database for the hash of the image under evaluation. As we deal with cryptographic hashes here, we can use a simple SQL implementation for a lookup of the hash. The hamming distance based search strategies of robust hashes are not required in this case.

5 Evaluation

The evaluation uses the pictures of a cheerleader team (Galaxy test set). The image set was already used in the optimization of [21] for evaluation. This set is particularly suitable, since it is a set of persons in similar poses with only small differences. This is challenging for a hash strategy sensible to similarity of image content and addresses the fact that we see as the core challenge of our work: robust re-identification of images without telling about the content of the references.

5.1 Test Set

The test set contains 3,804 images. These are first divided into two equally sized, randomly selected image sets. Since the creation of a database through combinations can require a large number of entries and would therefore take a long time (at 12 critical bits, 2^{12} hashes are required per image), we will use only 10% of a quantity to create the database. So we get two sets of 190 and 1,902 images. Of these two sets, the smaller is defined as known and the larger as unknown. Most pictures have a resolution of 1000×667 pixels, in landscape and portrait format.

5.2 Attacks

In order to test the new method for its robustness and reliability, images must be used for evaluation, which are manipulated in common ways. In order to send or upload images, they are usually subjected to JPEG compression and resized to minimize their storage requirements. Horizontal mirroring does not

affect the quality of the content, therefore the basic robust hash was deigned to be robust against it. It is therefore included in the attacks. The following 15 attacks are carried out on the images:

- JPEG compression (quality factors 90 to 10 in steps of 10)
- Scaling (sizes: 150%, 110%, 90%, 75%, 50%)
- Horizontal mirroring

GIMP version 2.10.6 was used to execute the attacks. Since saving images with JPEG compression is itself an attack, images in lossless PNG format were used for scaling and mirroring.

5.3 Design

Our aim is to evaluate the potential to apply our strategies in real-world scenarios. Therefore we have a close look at the error rates of the approaches. To compare them, we calculate the recall and precision based on our measurements. When assume that in most scenarios, the challenge will be to find the best recall with a precision close to 100%. In other words: we want to find the threshold where false positives are close to zero with the best possible number of true positives. From our experience with automation tools for investigations, a minimal false positive rate is preferred to a minimal false negative rate, as the distribution of samples to analyze often contains many more negative than positive examples.With a non-minimal false positive rate, therefore many false alarms would be raised leading to a significant overhead in the investigation. Still, to prevent bias, we use the same amount of positive and negative examples for evaluation.

We also need to calculate how many bits are likely to be below a given threshold and therefore seen as weak or critical. This allows us to estimate the size of the hash sets for the combination strategy depending on the threshold. Figure 16 shows the number of bits seen as critical (or weak) depending on the threshold for 38 test images. As for each image a 256 bit hash was calculated, the total number of hash bits was 9728. The threshold is a normalized value between 0 and 1 with 1 being the maximum distance from the median.

5.4 Results

First we discuss the results for the neutral element approaches 0-neutral and X-neutral. From our tests we learned that the 0-neutral approach leads to a high number of false-positives and a low precision as Table 2 shows. The

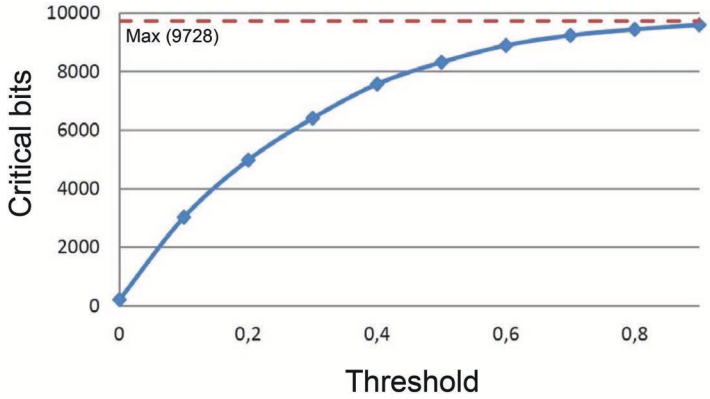

Figure 16 Number of critical bits for 38 images based on the given threshold.

Table 2 Recall and Precision for the 0-neutral approach depending on various thresholds

		Threshold					
		0.1	0.2	0.3	0.4	0.5	0.6
Recall	[%]	45.3	50.9	51.9	60.7	72.3	78.2
Precision	[%]	100	100	100	100	92.4	52
ACT	[ms]	38.4	38.4	38.8	38.9	39.2	39.3
AVT	[ms]	17.5	17.8	17.6	17.7	17.6	17.7

Table 3 Recall and Precision for the X-neutral approach depending on various thresholds

		Threshold								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Recall	[%]	28.4	30.4	36.1	47.2	57.5	66.3	80	84	90
Precision	[%]	100	100	100	100	100	100	100	100	99.6
ACT	[ms]	38.9	39.2	38.2	38.9	39.2	38.3	40.1	38.3	40
AVT	[ms]	17.7	18	17.9	18.1	17.8	18.1	18	17.9	18.2

trade-off controlled by the threshold between recall and precision was not satisfying. The recall with a precision of 100% at a threshold of 0.4 was 60.7%. Increasing the threshold and therefore the recall leads to a precision of 52% at a threshold of 0.6 and a recall of 78.2%. Computation time is only slightly affected by the strategy. ACT (average creation time) shows the average time for creating the hash, AVT (average verification time) the average time for the database lookup.

The X-neutral approach performs much better. As one can see in Table 3, a precision of 100% at the same time as a recall of 84% can be achieved with

threshold 0.8. Only beyond this threshold, the precision starts to drop. Timing differences are minimal again.

Therefore we see the threshold 0.4 for 0-neutral and the threshold 0.8 for X-neutral as the optimal thresholds for the respective strategies.

For the combination approach, in addition to threshold another parameter 'limit' is necessary. This parameter limits the maximum number of bits to be flipped by checking for an image how many weak bits have already been detected. If this number exceeds the specified upper limit, the remaining bits determined by the threshold parameter are ignored. Depending on the image, more or less critical bits can be identified. For example, in the creation under the same threshold value, an image can have only 9 critical bits while another image has 14 critical bits. This makes a huge difference in the number of resulting combinations and size of the hash database.

Images that have a large number of critical bits due to the set threshold could cause expansive calculations and database entries. Countering this by a low threshold could cause other images to have only few combinations and therefore have a low robustness. To compensate these differences, a limit of critical bits is used. This prevents too many combinations without the need of a small threshold.

We start with a limit of 14. For each limit, the threshold is selected that provides the best possible results. The limit is increased to 17. It would be possible to choose a higher limit. However, a time limit of 17 proved to be just within the practicable range, without using too many resources. A total of 20 runs are performed with regard to recall, precision and efficiency. The number of passes results from the two parameters threshold and limit. For each upper limit between [14,17], the values of threshold between [0.005; 0.025] are executed in steps of size 0.005. Larger thresholds are not used because the maximum number of critical bits has been used for almost all upper limits. Thresholds smaller than 0.005 do not further reduce the number of critical bits and thus always led to the same result. Table 4 shows the recall for these combinations. The optimal parameter set is a limit of 17 and a threshold of 0.01. By this, we achieve a recall of 85.8% and a precision of 100%. A single hash set at limit 17 has a maximum size of $2^{17} \times 32$ bytes or 4 MB.

5.4.1 JPEG

JPEG compression is the most common change an image will run through. Due to the lossy nature of JPEG, even with identical quality factors two generations of one image will differ with respect to its binary representation.

Table 4 Recall of the combination strategy for 14 to 17 bits to be flipped

	Limit			
Threshold	14	15	16	17
0.005	60	60	60	60
0.01	80	82.6	83.3	85.8
0.015	75.8	72.2	77.4	79.3
0.02	74.2	76.8	78.2	79.5
0.025	69.5	70.3	72.8	74.6

Therefore robustness to JPEG compression is on of the basic requirements for any robust hash strategy. In Figure 17 we show the recall of our strategies to JPEG quality factors between 10 and 90. As one can see, the block hash is perfectly robust towards JPEG. At a quality factor of 50, the combination strategy achieves a recall of over 90%. At the common quality factor 70, the recall of 0-neutral and X-neutral strategies is close to 80% with X-neutral performing better than 0-neutral in all cases.

5.4.2 Scaling

Another common processing step is the scaling of images. At moderate rates, image information is unchanged while the image file is modified significantly due to interpolation. At Figure 18 we see the recall of our strategies with respect to different scale factors. The basic block hash is again perfectly robust. The other strategies perform better with upscaling than with downscaling. Again, the combination strategy performs best with 70% at a downscaling of 50% and almost 90% with scaling of 90% size and above. X-neutral again performs better than 0-neutral.

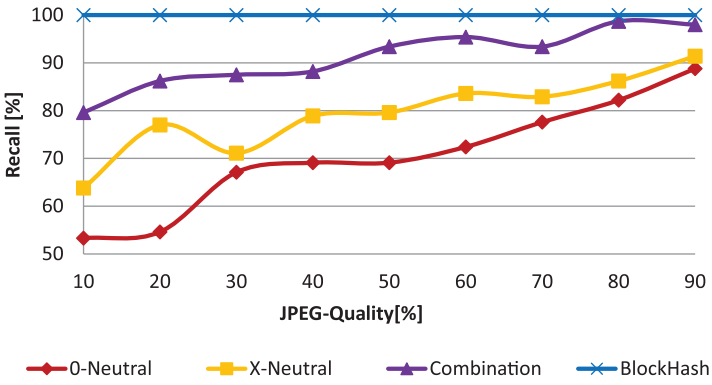

Figure 17 Recall of strategies for different JPEG quality factors.

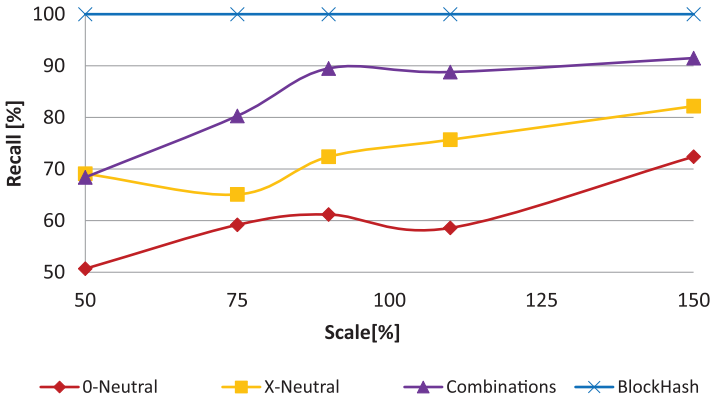

Figure 18 Recall of strategies for different scale factors.

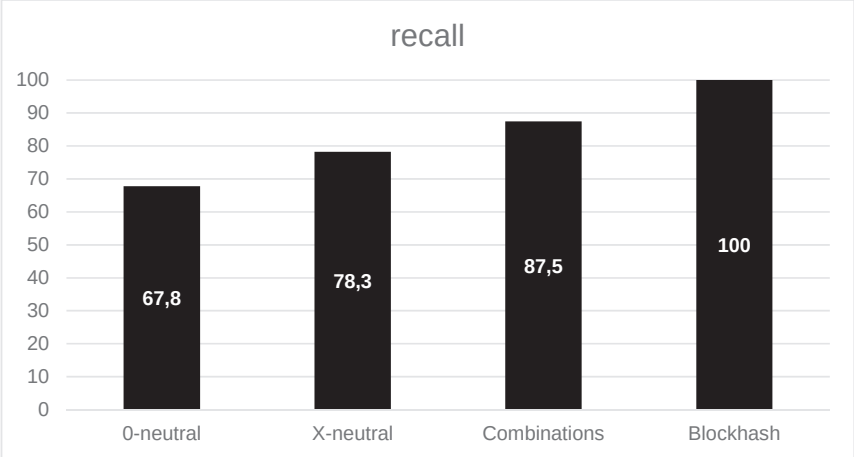

Figure 19 Recall comparison of the hashing strategies after horizontal mirroring.

5.4.3 Horizontal mirroring

A quasi-lossless attack on images known to fool cryptographic hashes is horizontal mirroring. The therefore have a closer look at the recall of our approach with respect to his attack in Figure 19. We see that the basic hash method by Steinebach et al. has recall of 100%, while there is a loss of recall when applying our strategies. Here the optimal thresholds were applied.

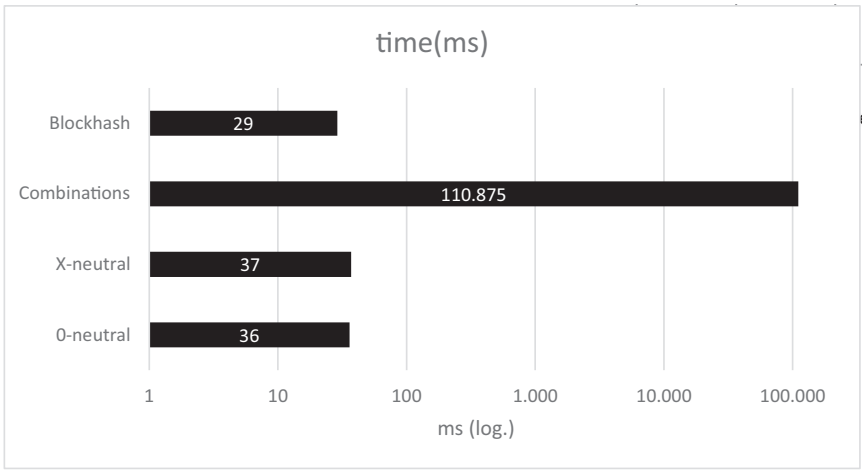

Figure 20 Processing time comparison of the hashing strategies.

5.4.4 Processing time

One aspect of interest besides the required memory size is the processing time for the creation of the hashes with by different strategies. Figure 20 shows the average processing time. As expected, the combination strategy requires by far the most processing time with almost 111.000 milliseconds. The other strategies are similar in their behavior. Block hash is the fastest, which is easily explainable as it is the base for X-neutral and 0-neutral, which cause a small overhead due to processing of the hash. The test system was a 3.2 GHz Intel Core i5-4570 with 8GB RAM.

6 Conclusion and Further Work

In this work we show that a hybrid approach combining characteristics of both robust and cryptographic hashes is possible and has its advantages. It is significantly more robust than a cryptographic hash which would not tolerate any of our evaluated changes and would end up with a recall of 0%. On the other hand it is less robust than a standard robust hash. In our evaluation, our strategies were up to 50% less robust than the basic block hash. The average recall achieved ranged between 67.8% for the 0-neutral strategy and 87.5% for the combination strategy. The latter was the most robust, but also most expensive strategy due to time and memory usage. The X-neutral strategy provides the best trade-off; it is almost as fast as the standard block hash and

requires no additional memory as only one cryptographic hash is calculated from the modified robust hash. It provides an average recall of 78.3% with a precision of 100%.

In other words: When accepting a loss of 20% of recall compared to a standard block hash, we can use a hybrid strategy combining the robustness of a block hash with the one-way nature of SHA-2 to re-identify images without any information leakage. This increases the overall privacy during forensic investigations as no information on the subject of the images can be derived from the hashes.

6.1 Further Work

This work can be seen as a first step into evaluating the behavior of robust hashes when used for cryptographic hashing. A deeper analysis of which bits are most likely to flip beyond a simple calculation of their distance from the median could be promising. For example, the neighborhood of the blocks the bits are calculated from could be of interest. A bit within a flat area may be more stable than one in a highly textured environment.

The block hash is not the only known robust hash function. It is one where previous research regarding the varying stability of its bits has been executed. Research with respect to other robust hash functions could help to find robust hash function with fewer bits likely to flip. Also other approaches for robust hashing not based on a binary representation could show characteristics suitable for a hybrid strategy.

Acknowledgment

This research work has been funded by the German Federal Ministery of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity.

References

[1] C. W. Adams. Legal issues pertaining to the development of digital forensic tools. In *2008 Third International Workshop on Systematic Approaches to Digital Forensic Engineering*, pages 123–132, May 2008.

[2] F. Armknecht and A. Dewald. Privacy-preserving email forensics. Technical Report CS-2015-03, Department Informatik, 2015.

[3] C. De Roover, C. De Vleeschouwer, F. Lefebvre, and B. Macq. Robust video hashing based on radial projections of key frames. *IEEE Transactions on Signal processing*, 53(10):4020–4037, 2005.

[4] P. N. Druzhkov and V. D. Kustikova. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1):9–15, 2016.

[5] J. Fridrich and M. Goljan. Robust hash functions for digital watermarking. In *Proceedings International Conference on Information Technology: Coding and Computing (Cat. No. PR00540)*, pages 178–183. IEEE, 2000.

[6] S. Gong, M. Cristani, C. C. Loy, and T. M. Hospedales. The re-identification challenge. In *Person re-identification*, pages 1–20. Springer, 2014.

[7] A. Haouzia and R. Noumeir. Methods for image authentication: a survey. *Multimedia tools and applications*, 39(1):1–46, 2008.

[8] S. Hou, T. Uehara, S. M. Yiu, L. C. K. Hui, and K. P. Chow. Privacy preserving multiple keyword search for confidential investigation of remote forensics. In *2011 Third International Conference on Multimedia Information Networking and Security*, pages 595–599, Nov. 2011.

[9] P. Kamavisdar, S. Saluja, and S. Agrawal. A survey on image classification approaches and techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(1):1005–1009, 2013.

[10] J. Katz, A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[11] E. V. A. N. Klinger and D. A. V. I. D. Starkweather. phash-the open source perceptual hash library. Technical report, accessed 2016-05-19.[Online]. Available: http://www.phash.org/apps, 2010.

[12] O. Koval, S. Voloshynovskiy, F. Beekhof, and T. Pun. Security analysis of robust perceptual hashing. In *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, page 681–906. International Society for Optics and Photonics, 2008.

[13] H. Liu, M. Steinebach, R. Stein, and F. Mayer. Privacy preserving forensics for jpeg images. *Electronic Imaging*, 2018(7):1–6, 2018.

[14] F. Mayer and M. Steinebach. Forensic image inspection assisted by deep learning. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ARES '17, pages 53:1–53:9, New York, NY, USA, 2017. ACM.

[15] A. Neelima and K. M. Singh. A short survey on perceptual hash function. *ADBU Journal of Engineering Technology*, 1, 2014.

[16] A. Peter, T. Hartmann, S. Muller, and S. Katzenbeisser. Privacypreserving architecture for forensic image recognition. pages 79–84, 12 2012.

[17] M. Schneider and S.-F. Chang. A robust content based digital signature for image authentication. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 3, pages 227–230. IEEE, 1996.

[18] S. Srinivasan. Security and privacy in the computer forensics context. In *2006 International Conference on Communication Technology*, pages 1–3, Nov 2006.

[19] P. Stahlberg, G. Miklau, and B. Neil Levine. Threats to privacy in the forensic analysis of database systems. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 91–102, New York, NY, USA, 2007. ACM.

[20] M. Steinebach. Robust hashing for efficient forensic analysis of image sets. In *International Conference on Digital Forensics and Cyber Crime*, pages 180–187. Springer, 2011.

[21] M. Steinebach, H. Liu, and Y. Yannikos. Forbild: Efficient robust image hashing. In *Media Watermarking, Security, and Forensics 2012*, volume 8303, page 83030O. International Society for Optics and Photonics, 2012.

[22] L. Weng and B. Preneel. Attacking some perceptual image hash algorithms. In *2007 IEEE International Conference on Multimedia and Expo*, pages 879–882. IEEE, 2007.

[23] B. Yang, F. Gu, and X. Niu. Block mean value based image perceptual hashing. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 167–172. IEEE, 2006.

Biographies

Martin Steinebach is the manager of the Media Security and IT Forensics division at Fraunhofer SIT. From 2003 to 2007 he was the manager of the Media Security in IT division at Fraunhofer IPSI. He studied computer science at the Technical University of Darmstadt and finished his diploma thesis on copyright protection for digital audio in 1999. In 2003 he received his PhD at the Technical University of Darmstadt for this work on digital audio watermarking. In 2016 he became honorary professor at the TU Darmstadt. He gives lectures on Multimedia Security as well as Civil Security. He is Principle Investigator at CRISP and represents IT Forensics and Big Data Security. Before he was Principle Investigator at CASED with the topics Multimedia Security and IT Forensics. In 2012 his work on robust image hashing for detection of child pornography reached the second rank "Deutscher IT Sicherheitspreis", an award funded by Host Görtz.

Sebastian Lutz received his B.Sc. in business informatics at the University of Mannheim, Germany, in 2015. He specialized his knowledge in security related topics as part of a M.Sc. IT-Security study in 2016 and successfully completed his degree in 2019. A major research interest was on information security, robust hashing and digital forensics. He wrote his master's thesis

"Privacy and Robust Hashing" at Fraunhofer SIT (Darmstadt, Germany). Currently he is an employee at ITK-Engineering (Rülzheim, Germany) and works as a Cyber Security Engineer. His duties include working as a developer and creating security concepts in the automotive and medical sectors.

Huajian Liu received his B.S. and M.S. degrees in electronic engineering from Dalian University of Technology, China, in 1999 and 2002, respectively, and his Ph.D. degree in computer science from Technical University Darmstadt, Germany, in 2008. He is currently a senior research scientist at Fraunhofer Institute for Secure Information Technology (SIT). His major research interests include information security, digital watermarking, robust hashing and digital forensics.