IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 July 2024, accepted 22 July 2024, date of publication 29 July 2024, date of current version 7 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3435027

RESEARCH ARTICLE

ChaSAM: An Architecture Based on Perceptual Hashing for Image Detection in Computer Forensics

HERICSON DOS SANTOS[1], TIAGO DOS SANTOS MARTINS[1], JORGE ANDRE DOMINGUES BARRETO[2], LUIS HIDEO VASCONCELOS NAKAMURA[3,4], CAETANO MAZZONI RANIERI[5], ROBSON E. DE GRANDE[6], (Member, IEEE), GERALDO P. ROCHA FILHO[7], AND RODOLFO I. MENEGUETTE[3], (Senior Member, IEEE)

[1]São Paulo Scientific Police, Araçatuba 16018-120, Brazil
[2]São Paulo State Police, São Paulo 15015-400, Brazil
[3]Institute of Mathematical and Computer Sciences, University of São Paulo (USP), São Carlos 13566-590, Brazil
[4]Federal Institute of Science, Education and Technology of São Paulo (IFSP), Catanduva 15808-305, Brazil
[5]Institute of Geosciences and Exact Sciences, São Paulo State University (UNESP), Rio Claro 13506-692, Brazil
[6]Department of Computer Science, Brock University, St. Catharines, ON L2S 3A1, Canada
[7]Department of Exact and Technological Sciences, State University of Southwest Bahia (UESB), Vitória da Conquista 45083-900, Brazil

Corresponding author: Rodolfo I. Meneguette (meneguette@icmc.usp.br)

This work was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior–Brasil (CAPES).

ABSTRACT The growing prevalence of digital crimes, especially those involving Child Sexual Abuse Material (CSAM) and revenge pornography, highlights the need for advanced forensic techniques to identify and analyze illicit content. While cryptographic hashing is commonly used in computer forensics, its effectiveness is often challenged because criminals can modify original information to create a new cryptographic hash. Perceptual hashes address this problem by focusing on the visual identity of the file rather than its bit-by-bit representation. This study introduces ChaSAM Forensics, a methodology that efficiently identifies illicit material using perceptual hashing techniques to track and identify illicit content, with a focus on child abuse material. Two new perceptual hashing algorithms, chHash and domiHash, were designed for integration into ChaSAM. The results showed that, under the tested conditions, the proposed chHash algorithm was more accurate than the established pHash algorithm when applied in a single iteration. Combinations of algorithms in two iterations were also assessed.

INDEX TERMS Forensic computing, perceptual hashing, image detection, similarity.

I. INTRODUCTION

The phenomenon of digital inclusion in Brazil began in the early 2000s with the introduction of broadband connectivity, which allowed access to the internet for a wider population, rather than being limited to only affluent individuals, businesses, and universities [1], [2], [3]. At the same time, technological advancements have led to significant increases in the processing power and storage capacity of computing equipment, allowing for a plethora of new potential applications of digital systems.

The associate editor coordinating the review of this manuscript and approving it for publication was Jeon Gwanggil.

However, access to technology has brought with it various forms of cybercrimes [4], [5], [6], including copyright infringement of music, movies and software, crimes against honor perpetrated through social networks, crimes involving the acquisition, storage and distribution of material containing scenes of child sexual abuse [7], [8], and other crimes recently incorporated into Brazilian law, such as revenge pornography and sextortion [9], [10].

Currently, one of the most prevalent forms of cybercrime in the digital realm is the sharing of CSAM (Child Sexual Abuse Material). Between January and April of 2021, Safernet Brazil reported approximately 15,800 pages related to CSAM. Out of these, 7,248 pages were removed due to

suspected criminal activity. This reflects a 33.45% increase in reports compared to the same period in 2020 when more than 11,800 pages were reported [11].

Computer Forensics or Digital Forensic Analysis has become an important tool in assisting the justice system in fighting digital crimes by using analysis and methods to identify and collect efficient evidence [12], [13], [14]. When examining digital files such as videos and photos, for instance in the case of CSAM files, most methods extract the cryptographic hash code of the file and compare it to a database of pre-existing hashes [15], [16], [17]. If the file matches a hash in the database, it is considered a positive match; otherwise, it is discarded.

This verification becomes highly cumbersome to the forensic analyst when analyzing large volumes of data from different types of seized computer equipment in operations is necessary. Furthermore, criminals have been applying what the doctrine calls anti-forensic techniques to digital media files (videos and photos) [18], [19], [20], [21]. These are intentional alterations to part of the digital file content to produce a result contrary to the truth to deceive detection and hash-to-hash comparison software. The most common digital alterations or modifications include inserting watermarks or texts, resizing the image, vertical or horizontal mirroring, color alteration, and application of tilts [22].

Modified or adulterated images have a different cryptographic hash from the original, which hinders the matching procedure. In other words, forensic systems based on cryptographic hashes identify these adulterated images as new and without matches in the database [15], [23], [24].

To mitigate this identification issue and reduce the occurrence of false negatives in forensic examinations, one possible solution is the application of perceptual hashing concepts that consider the visual identity [25] of the file rather than its bit-by-bit computational representation. Thus, in a hypothetical scenario, supposed modified or altered images would not be discarded but instead separated and presented for the forensic analyst to decide visually whether they are relevant for examination. Several studies in the literature have addressed perceptual hashing and the challenge of comparing large volumes of images [15], [22], [25], [26].

This paper presents a forensic methodology named ChaSAM Forensics and the chHash algorithm, which consists of a computational architecture capable of identifying images based on their similarities, particularly in the search for CSAM files and revenge pornography on computer devices seized in police operations. To achieve this, ChaSAM searches for files within the *target* folder only if they have similar counterparts in the candidate files folder within the *source* folder. Additionally, we propose two perceptual hashes to minimize false negative results.

For evaluation purposes, we compared the proposed hashing algorithms with the main perceptual hashes established in the literature using images produced and cataloged by the authors. The proposed image set consisted of 12,920 diverse files housed in a folder named *target*, supported by a second folder named *source* for the comparison of candidate images.

The main contribution of this work lies in the development and integration of two novel perceptual hashing algorithms, chHash and domiHash, within the ChaSAM Forensics tool. Unlike existing perceptual hash algorithms, chHash is particularly effective in identifying similar images while minimizing false positives, especially when used in combination with dHash or dHash-v algorithms. This dual-iteration approach significantly enhances accuracy and efficiency in detecting manipulated images, such as those found in CSAM. Additionally, the use of parallel processing routines within ChaSAM further improves performance, making it a robust tool for forensic investigations involving large volumes of digital images.

More specifically, the contributions of this work include:

- Designing a forensic analysis tool for processing and identifying modified or adulterated images;
- Designing and evaluating two perceptual hashing algorithms;
- Building a test image database and using it in the simulations for evaluating the proposed methodology.

The remainder of this paper is organized as follows. Section II presents the background and fundamentals on hashing and hamming distance. Section III describes recent works related to the scope of image detection in computer forensics. Section IV introduces ChaSAM Forensics and its architecture. Section V describes the study case utilizing ChaSAM Forensics and assesses its efficiency. Section VI presents an effectiveness analysis of ChaSAM Forensics and discusses it. Finally, Section VII concludes the paper and presents future work directions.

II. BACKGROUND

This section briefly reviews the technical definitions of conventional hash functions, perceptual hashes, and supporting variables.

A. HASH FUNCTION

A hash function is an alphanumeric representation of fixed length resulting from a mathematical formula uniquely identifying digital data. Once the hash function is calculated and its value h obtained, it is not possible to identify the original message m [27], [28], [29].

One of the basic requirements of any cryptographic hash function is that it is computationally unfeasible to find two distinct values from the same input content. The opposite premise is also valid: the hash function must always yield different outputs for different contents [29].

Conventional hash functions are very efficient for comparing two files with identical content. Thus, if A and B are two samples being compared and h is a hash function, $A = B$ if and only if $h(A) = h(B)$. In other words, two images will be identical whenever their hash values are equal. However, even the slightest alteration, imperceptible to the naked eye in one

of the images, will result in $h(A)$ being different from $h(B)$. Perceptual hashes are used to compare different images with visually similar content.

B. PERCEPTUAL HASH

According to [30], a perceptual hash is a fingerprint of a multimedia file derived from various features of its content. Unlike conventional cryptographic hash functions that rely on the avalanche effect of small input changes leading to drastic output changes, perceptual hashes are close to each other if the features are similar.

In the same sense [31] asserts that the perceptual hash is a fingerprint of the content of a digital media. He adds that if the perceptual hash is compared with the cryptographic hash, the former shows many advantages in defending against image-based fake news attacks, because it is possible to detect deliberate manipulation of images, while at the same time tolerating normal changes in format or resolution.

In practice, the result of applying a perceptual hash to two different but visually similar images is presented as possible matches as long as their differences fall within a minimum value range. Unlike conventional cryptographic hash functions, which are highly sensitive to bit changes, perceptual image hash is scalable and tolerates the imprecision of how computers represent image content.

The perceptual image hash, which maps the content of an image to a short binary string, can be considered a digital summary of the image content. It is essentially a mapping that satisfies some constraints [32]. There are several types of perceptual hash algorithms. Among them, the most well-known ones are: Average hashing (aHash), Perceptual hashing (pHash), Difference hashing (dHash), and Wavelet hashing (wHash) [25].

C. HAMMING DISTANCE AND THRESHOLD

The Hamming Distance (HD) calculates the number of positions in which two strings differ. The objective is to identify, within a set of characters, their counterparts in another set, pointing out divergent ones [25]. One point is added to this distance for each of the divergent characters. The greater the value of the Hamming Distance, the less similar the compared binary strings are.

In the context of similarity matching, it is necessary to assign an acceptable limit to this distance so that visually, the strings retain some similarity [33], [34]. This acceptable limit is given by the Threshold variable *Th*. Thus, two binary strings or two images are similar if, and only if, the Hamming Distance is less than or equal to the previously configured threshold [25].

We need to compute the respective binary strings to apply this concept in image comparison. Consider two different images computationally but with similar perceptual elements, as shown in Figure 1.

When we apply the conventional binarization method, we found the binary strings of the image on the left and right horses as shown in Figure 2. The images have seven different bits between them, it is, their Hamming Distance equals 7. The algorithm considers an image a possible similarity match if the Hamming distance is less than a threshold.

Suppose the variable has been configured with the threshold limit set to five; in this case, the algorithm would not indicate the two images as similar because the Hamming Distance is higher than the threshold. However, if the threshold were configured to ten, then the two images would be considered similar.

It is possible to calculate another variable called Percentage of Equality (PE), which is the percentage of similarity between the images. For the previous example, seven different bits out of a total of 64 equal 57 corresponding bits; that is, the images are 89% similar or 11% different from each other.

III. RELATED WORK

The cryptographic hash comparison process takes into account the computational identity of the images, while the perceptual hash considers their similar characteristics. The latter is capable of presenting as search results images that are not computationally the same, but are visually similar. The big problem with this method is that it can present a large number of false positive results, reducing qualitative effectiveness. Furthermore, by presenting a larger set of images as a result, the processing power of the comparisons is greater, requiring the algorithm responsible for this work to be efficient, that is, faster than currently existing algorithms.

An overview of perceptual hashing was presented by Farid [35], which addressed the persistence of harmful online content such as terror-related materials and child sexual abuse imagery. The paper reviews the mechanisms, advantages, disadvantages, and real-world applications of perceptual hashing.

Samanta and Jain [22] analyzed the detection of image manipulations using perceptual hash algorithms and discussed the efficiency of the algorithms in detecting image similarity. They mentioned that simple modifications such as changes in color, brightness, and texture are perfectly detected by the algorithms. However, the same does not occur when drastic changes are present, such as cropping, rotations, and mirroring.

McKeown and Buchanan [36] presented a large-scale evaluation of several perceptual hashing algorithms against various image modifications to understand the effectiveness of these algorithms in detecting similar images despite these modifications. The study provided insights into the strengths and weaknesses of popular perceptual hashing algorithms against content-preserving modifications.

Zheng et al [37] proposed MR-DCAE, a deep convolutional autoencoder based on manifold regularization, which aims to improve the detection of unauthorized transmissions. This method uses manifold regularization to preserve the intrinsic structure of the data, providing more accurate and robust detection. The approach has demonstrated superior

FIGURE 1. Hamming Distance and image binarization.

1 ➔ 101111111001111100000001000000000000000000000000000000000000000
2 ➔ 111111101001111000000000100000001000000100000000000000000000000

FIGURE 2. Bit-by-bit comparison of images.

effectiveness compared to traditional techniques, especially in scenarios with noisy and complex data.

Zheng et al. [38] propuseram um método de classificação de imagens de constelação em tempo real utilizando a rede leve MobileViT. Esta abordagem combina a eficiência computacional do MobileNet com a capacidade de modelagem de visões globais do ViT (Vision Transformer), resultando em um modelo que oferece alta precisão de classificação enquanto mantém baixos requisitos de processamento.

Torres [25] addresses the television and film industry's difficulty in protecting their media content against digital pirates. His work involved detecting alterations in video files transmitted by cable TVs. According to the author, this analysis resulted in a significant computational processing problem, as it required the computational analysis of each frame of a video that has 30 frames per second, resulting in thousands of static images to be processed and compared in search of possible channels on the Internet that are broadcasting the content without legal authorization.

Faria [15] argues that the analysis of files performed by computer experts is limited to the hash-to-hash comparison of cryptographic algorithms, mainly in examinations where files of child pornography are sought, and there is a lack of methods that allow the computational expert to compare similar files. He also mentions free software tools and general-purpose similarity hashes; however, they are not very performative.

The work by Menezes and Silva [26] addressed what the authors called the problem of the modern world, which is the accumulation of various identical or similar files at the expense of disk storage space. The author argues that programs can compare files for similarity; some are available for free download on the Internet. However, he points out that these programs are slow.

In light of this problem, the work of Santos [8] discusses the exponential increase in crimes related to the production, storage, and sharing of child sexual abuse material on the internet, which, after police operations, will be subject to forensic examination to seek the criminal materiality, thus requiring computational methods capable of supporting the work of the criminal expert.

Hao et al. [39] examined the robustness of perceptual hashing algorithms used in digital forensics and cybercrime studies. The study highlighted the vulnerabilities of perceptual hashing algorithms and demonstrated the need for more robust designs to protect against such attacks

It is observed, therefore, that the studies converge to the increasing existence of devices and files to be analyzed by criminal experts, with the aggravating factor that many of these files undergo some anti-forensic technique for their malicious manipulation, aiming to cover up or produce adverse results from the truth.

Much of these manipulations are related to copyright infringement [25], revenge pornography [10], and/or production and storage of child sexual abuse material [8]. In the first type of crime, Torres [25] encountered the difficulty of processing thousands of frames (video frames) of interest to detect whether there was an alteration of the original media. In the last type of crime mentioned, the authors [8], [15] discuss the lack of methods to examine files containing child abuse content, mainly in the search for similarities, whose cryptographic hash databases are deficient in this regard.

We sought to test the most common perceptual hashes in the literature and, from them, develop two variants: chHash and domiHash.

Unlike the other described works, our work comprehensively evaluates the main perceptual hashes, comparing them with two new strategies proposed in this paper. For

deal with the difficulties of comparing images that have undergone major modifications, such as: drastic cropping and inclinations with angulations above 5º. The domiHash algorithm proved to be very effective when detecting files with inclinations above 5º, however its efficiency drops drastically in relation to other modifications such as: vertical and horizontal mirroring. Furthermore, in our evaluation, we employ parallelism in the execution of the image comparison process to enhance the efficiency of the results.

IV. CHASAM FORENSICS

The ChaSAM Forensics acronym combines the meanings of two keywords: Child Sexual Abuse Material (CSAM) and the word ''hash''. This section outlines this methodology in terms of the computational architecture, the construction of the original image dataset, and the two proposed perceptual hashes. The architecture source code is available in the GitHub repository.[1]

A. CHASAM FORENSICS ARCHITECTURE

The ChaSAM Forensics architecture is depicted in Figure 3. The basic module is named ''Media Search Stream'' (1), and its purpose is to load candidate files from the ''Target'' image folder into memory, extract their respective perceptual hashes using the chosen algorithm during execution, and compare them with the files from the ''Source'' folder.

In the ''Media Search Stream'' module, the parallel routines of ChaSAM are triggered to increase the speed of obtaining results. Each routine is responsible for extracting the perceptual hash (2) from an image in the ''Target'' folder (3) and comparing it with the input images from the ''Source'' folder (4).

In our prototype of ChaSAM, it is possible to specify the number of parallel routines to be used. In a hypothetical case, by specifying that ChaSAM should use 32 routines, each routine will execute the loading, extraction, and comparison actions on an image from the ''Target'' folder, thus increasing the system's execution performance. When a routine finishes its work, and there are still images to be processed, it will move to the next image until all images in the folder are compared and the final result is synchronized and presented (5).

B. DATASET IMAGES DEFINITON

The dataset images' definition is shown in Figure 4.

We produce a dataset of 380 static images from a Xiaomi Redmi Note 10 mobile device. The images have dimensions of 4000 × 2250 pixels in landscape orientation and 2250 × 4000 pixels in portrait orientation, with a resolution of 72 DPI. The set of images used in these tests was produced by these authors focusing on urban and rural landscapes, animals and vehicles.

Subsequently, the dataset was balanced and the images resized keeping all other attributes 640 × 480 pixels in landscape orientation and 480 × 640 pixels in portrait orientation. All metadata, including EXIF and geolocation, was preserved. This process was necessary to reduce the average size, saving approximately 90% of disk storage space (1).

After creating the set of original images, we generated the set of altered images by applying the most common transformations found in alterations or tampering of images in forensic examinations [19], [22] to each image (2).

A total of nine different transformations were applied: blur, crop, flip-h, flip-v, grayscale, text insertion, and threshold. These resulted in 3,420 distinct files.

However, in the rotation transformations, 17 different inclinations were applied. Only the folder containing rotated images contained 6,460 files with the following rotations: 1°, 2°, 3°, 4°, 5°, 6°, 7°, 8°, 9°, 10°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°.

The ''Watermark'' folder holds the images with watermarks. For each image in this folder, nine possibilities of tampering were assigned by dividing the image into 3×3 quadrants, foreseeing the insertion of an element in each of these nine possible positions, resulting in a total of 3,420 files.

The ''original'' image folder was added without tampering or transformation to complete the dataset's set of images, serving as the control file folder. Therefore, the final set comprised 12,920 files distributed across 10 different folders. This collection of folders and files was named ''target''.

Finally, for the simulations in this article, a folder of candidate images (3) was created, whose files were extracted from the set of images in the ''target'' folder (2). The ChaSAM algorithm should search the ''target'' image folder for all images similar to those in the ''source'' image folder and present the results in a third folder labeled ''extract'' (5) using parallel routines (4).

C. PROPOSALS FOR PERCEPTUAL HASHES

For this work, we propose two perceptual hashes. The first one is a derivation of the dHash algorithm called Domi Hash (domiHash). The second one is the ChaSAM Hash (chHash), which, unlike perceptual algorithms found in the literature, does not utilize conventional grayscale transformation. In the following subsections, we describe the two proposed hashes in detail.

1) DOMI HASH (DOMIHASH)

The domiHash is a perceptual hash algorithm derived from dHash. It was developed to enhance the detection of rotated images, which is a limitation of existing perceptual hash algorithms. While the former performs a linear comparison, either by row or column, domiHash compares the difference along the diagonal of the image matrix, making it possible to identify rotated images. To achieve this, the algorithm performs the following tasks for perceptual hash extraction:

[1] Available at: https://github.com/tsmweb/chasam-cli.git

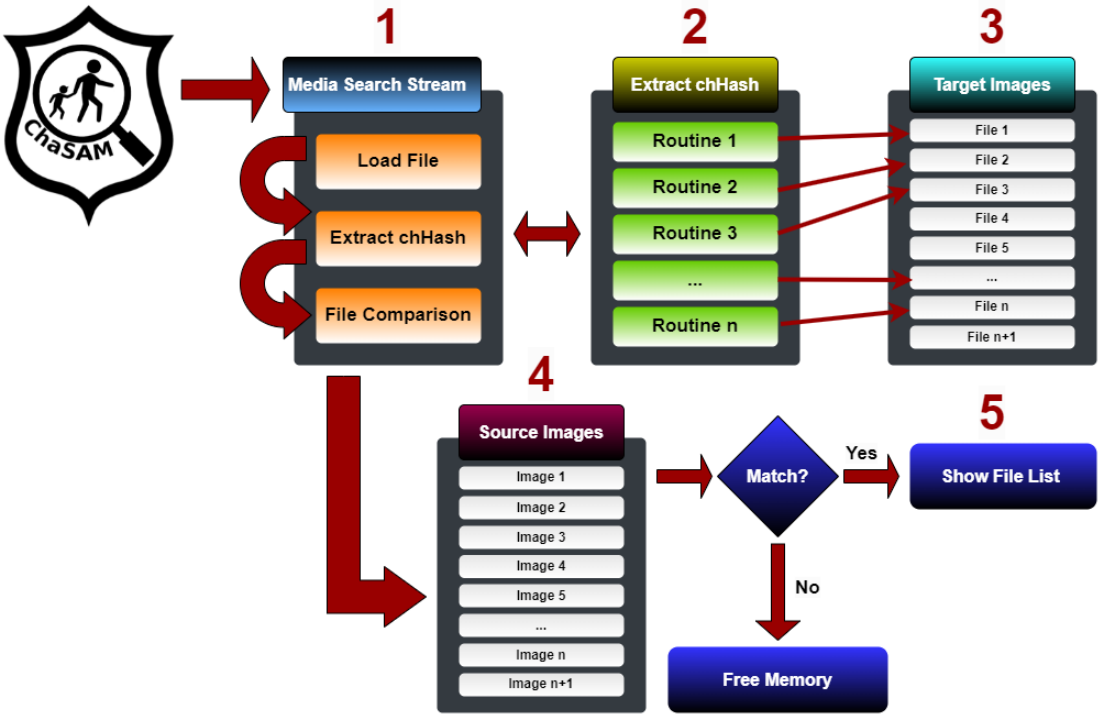


FIGURE 3. ChaSAM Forensics architecture.

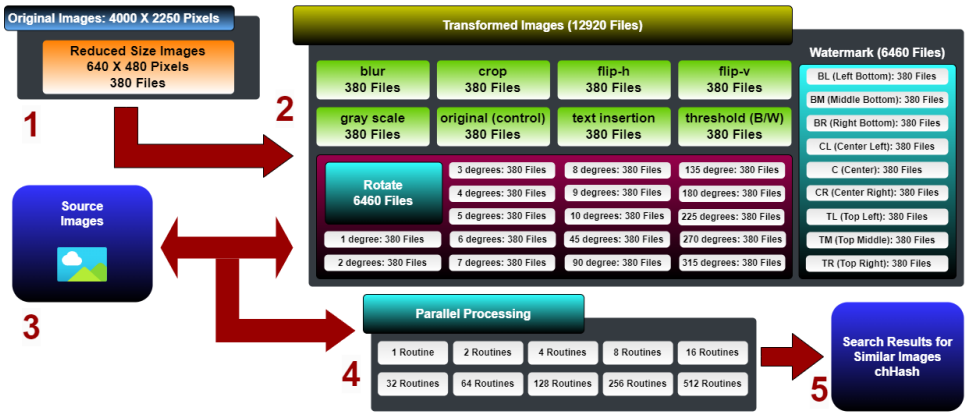


FIGURE 4. Dataset images' definition.

- **Resizing:** The image is resized to a fixed dimension, typically 9×9 pixels. Note that in this method, the matrix must be square, unlike dHash;
- **Conversion:** The image is converted to grayscale. This reduces the complexity of the image by retaining only pixel intensities;
- **Calculation:** The difference between the values of each pixel along the diagonal of the image is calculated, starting from the upper-left corner of the matrix and extending radially to the other corners;
- **Binarization:** Each resulting difference is compared with the corresponding pixel. If the pixel value exceeds the difference, it is considered 1; otherwise, it is 0. This step transforms the image into its binary representation;
- **Hash Generation:** The resulting 64-bit binary string is transformed into its hexadecimal representation. This representation is the perceptual hash of the image.

Algorithm 1 describe the domihash, first of all, the image is resized to a width and height of 9 X 9 pixels using the bilinear resizing method (lines 5 and 6). Then, we use the ConvertToGrayArray method to convert the pixels to grayscale (line 7):

In the sequence, the domiHash algorithm utilizes the first "for" loop (lines 10 to 18) to iterate over the pixel matrix

Algorithm 1 *domiHash* Function

```
1:  function domiHash(image: Image)
2:      if image = null then
3:          return 0, error.New(''The Image cannot null'')
4:      end if
5:      width, height ← 9, 9
6:      resized ← Resize(image, width, height, Bilinear)
7:      pixels ← transform.ConvertToGrayArray(resized)
8:      index ← 0
9:      hash_value (64 bits integer)
10:     for x ← width − 1; x ≥ 0; x−− do
11:         for y ← 0; y < (width − x −1); y++ do
12:             _x ← x + y
13:             if pixels[y][_x] > pixels[y+1][_x+1] then
14:                 hash_value |= 1 ≪ integer(64−index−1)
15:             end if
16:             index ← index+1
17:         end for
18:     end for
19:     for y ← height − 1; y > 0; y−− do
20:         for x ← 0; x < (w − y − 1); x++ do
21:             _y ← y + x
22:             if pixels[_y][x] > pixels[_y+1][x+1] then
23:                 hash_value |= 1 ≪ integer(64-index-1)
24:             end if
25:         end for
26:     end for
27:     return hash_value, null
28: end function
```

and compare their adjacent values. If the current pixel is greater than the pixel in the bottom-right diagonal, the bit 1 is appended to the binary string; otherwise, the string receives the bit 0.

After completing the iterations of the first ''for'' loop, the algorithm traverses the second half of the matrix (lines 19 to 26). This time, the algorithm iterates the pixel matrix from bottom to top. If the current pixel is greater than the pixel in the bottom-right diagonal, the bit 1 (one) is appended to the binary string; otherwise, the string receives the bit 0 (zero).

2) CHASAM HASH (CHHASH)

The chHash is a perceptual hashing algorithm different from others in several aspects: first, the dimensions of the reduced image form a matrix of 32 × 32 pixels, whereas in most conventional algorithms, the pixel matrix is reduced to 8 × 8 pixels. We do not use the traditional luminance grayscale transformation but rather a process called image thresholding, which aims to convert grayscale images into two distinct threshold categories: black and white [40]. Therefore, chHash combines image thresholding methods with perceptual hashing concepts.

The chHash algorithm starts to differentiate itself from the standard when the chosen threshold for its programming is not 127 [41] but 114.[2] For the binarization, we use the Discrete Cosine Transform (DCT) concept, similar to the pHash algorithm.

In summary, the algorithm performs the following tasks for extracting the perceptual hash of an image:

- **Resizing:** Initially, the image is resized to a fixed dimension of 32 × 32 pixels, unlike other methods that resize images to dimensions close to 8 × 8;
- **Conversion:** The image is converted to black and white (thresholding) with a threshold of 114, unlike the original method that works with a luminance of 128;
- **Calculation:** The Discrete Cosine Transform (DCT) is applied to the image to convert space information to frequency information. The low-frequency part of the spectrum is retained, while higher frequencies are cut off, reducing the amount of information to an 8 × 8-bit matrix, similar to the pHash algorithm;
- **Binarization:** Each image pixel is compared with the calculated mean. If the pixel value exceeds the mean, it is considered 1; otherwise, it is considered 0. This process transforms the image into its binary representation;
- **Hash Generation:** The resulting 64-bit binary string is transformed into its hexadecimal representation. This representation is the perceptual hash of the image;

The chHash was developed as an alternative to conventional perceptual hashes to improve the image comparison process.

Algorithm 2 demonstrates how chHash works in image processing: initially, it is resized to a width and height of 32 X 32 pixels using the bilinear resizing method (lines 5 and 6). Then, we use the ConvertToBlackWhiteArray method to convert the pixels to black and white (line 7) using the ''Thresholding'' method. chHash uses a luminance threshold of 114, unlike the standard algorithm that uses 128 for pixel comparison. Next, the Discrete Cosine Transform is applied to the resulting pixel matrix (line 8). Algorithm.

Algorithm 2 demonstrates how chHash works in image processing: initially, it is resized to a width and height of 32 X 32 pixels using the bilinear resizing method (lines 5 and 6). Then, we use the ConvertToBlackWhiteArray method to convert the pixels to black and white (line 7) using the ''Thresholding'' method. chHash uses a luminance threshold of 114, unlike the standard algorithm that uses 128 for pixel comparison. Next, the Discrete Cosine Transform is applied to the resulting pixel matrix (line 8).

We redefine the width and height variables to be used in the second part of the code. This step reduces the pixel matrix from 32 × 32 to 8 × 8 positions (line 9). Using a concatenated ''for'' loop, we calculate the sum of the resulting values of this matrix and extract its average (lines 12 to 21).

Finally, we use another ''for'' loop (lines 23 to 27) to iterate through the pixel matrix row by row. If the current pixel is greater than the average of the matrix, bit 1 is appended to

[2]Value obtained empirically, by conducting tests and observing the behaviour of the results.

Algorithm 2 *chHash* Function

```
1:  function chasamHash(image: Image)
2:      if image = null then
3:          return 0, error.New(''The Image cannot null'')
4:      end if
5:      width, height ← 32, 32
6:      resized ← Resize(image, width, height, Bilinear)
7:      pixels ← transform.ConvertBWarray(resized, 114)
8:      dct ← transform.cosine_transform(pix, wd, hg)
9:      width, height ← 8, 8
10:     flatDct [64]float64
11:     sum ← 0.0
12:     for y ← 0; y < height; y++ do
13:         for y ← 0; y < (width − x -1); y++ do
14:             for x ← 0; x < width; x++ do
15:                 sum ← sum + dct[y][x]
16:                 flatDct[height*width+x] ← dct[y][x]
17:             end for
18:         end for
19:     end for
20:     sum ← sum − dct[0], [0]
21:     average ← sum / float64(63)
22:     var hash_value (64 bits integer)
23:     for index, pixel ← range flatDct do
24:         if pixel > average then
25:             hash_value | = 1 ≪ integer(64-index-1)
26:         end if
27:     end for
28:     return hash_value, null
29: end function
```

the binary string; otherwise, the string receives the bit 0. This part of the code (lines 9 to 28) forms the aHash (average hash algorithm) basis.

V. CHASAM EFFICIENCY CASE STUDY

This section presents the case study conducted with ChaSAM Forensics to obtain efficiency metrics regarding the number of parallel routines to be triggered while comparing similarities between large volumes of images.

The programming of ChaSAM Forensics was conducted using the Go programming language, developed by Google. Its main characteristic is high performance achieved through better utilization of processing hardware, using parallelism and concurrency with the goroutines mechanism, as these parallel routines are called in this language.

A. CHASAM FORENSICS EXECUTION SYNTAX

A basic version of ChaSAM Forensics was built for the simulations in this work, usable on both Windows and Linux operating system architectures. It compares all images through perceptual hash algorithms, with parameters being the source and target folders. As a result, ChaSAM creates a third folder named ''extracted'', which stores the result of files that the algorithm deemed similar.

The system call consists of providing the following parameters: (i) the path of the source image folder, which contains the candidate files; (ii) the path of the target image folder, which contains the volume of files to be examined; (iii) the type of perceptual hash used in the comparison; (iv) the number of tolerable different bits considering the Hamming distance; and (v) the number of parallel routines for task execution.

At this initial stage, we only analyzed ChaSAM's execution efficiency. This characteristic is linked to the execution time of the comparison task and is directly proportional to the number of parallel routines used by the program. In the first simulation, with 128 parallel routines, ChaSAM completed the task in just 23.97 seconds. In contrast, when configured to run with two parallel routines, the execution time jumped to 52.18 seconds.

The results of these initial tests aimed to demonstrate that parallelism is necessary for comparing large volumes of images. We also sought to ascertain whether this assertion holds true for all computing platforms. In other words, whether different types of hardware (computers) and software (operating systems) behave similarly when configuring more or fewer parallel routines for task execution.

B. HARDWARE AND SOFTWARE DEVICES USED IN THE SIMULATION

For the simulations, 6 (six) different configurations of hardware and software were used, including desktops, notebooks, and workstations running Windows (in various versions) and Linux (Ubuntu 22).

The tests were conducted on computers accessible to most people in Brazil, except for the workstation named ''Gray'' which has more advanced hardware and software configurations. All other machines have configurations commonly used by home users, which are also found in most locations involved in cybercrime, especially in operations combating child sexual abuse.

The computers used in the tests were given color names to facilitate the presentation of results in tables and graphs that we will present. During the tests, all six machines performed the same type of processing: comparing the similarity between candidate files in the source folder with the volume of files transformed from the target folder. In each of the simulations, we varied only the number of parallel routines to observe the algorithm's behavior during the processing and comparison of large volumes of images to measure its efficiency in execution time.

C. SIMULATION COMPILATION AND RESULTS

Each of the computers described in Figure 5 performed 400 iterations (comparison task), expecting as a return the time consumed in seconds to process and compare the similarity between the images in the source and target folders. After each cycle, the simple arithmetic mean of the execution times was extracted to compare the tested equipment. The results were compiled into their respective tables and graphs.

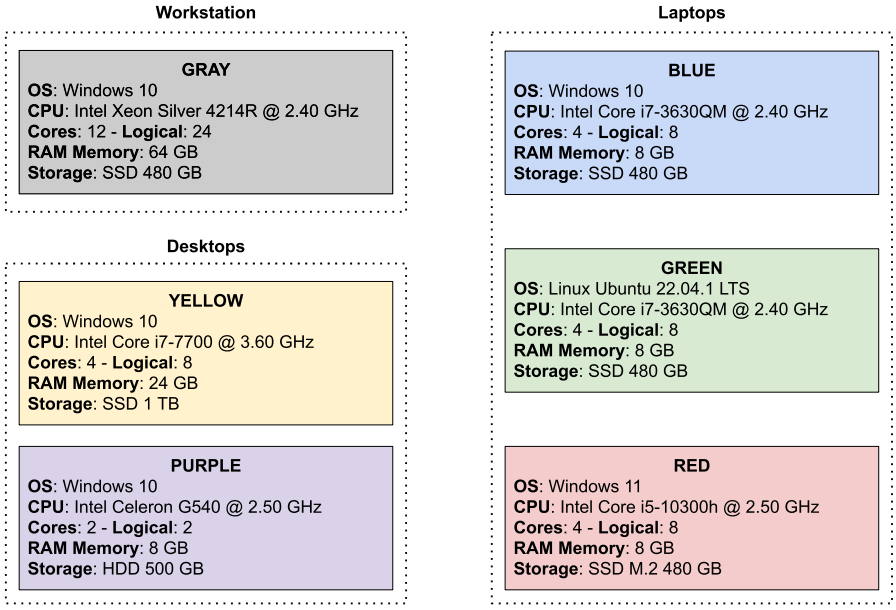


FIGURE 5. Configuration of tested devices.

1) SIMULATION RESULTS

Table 1 presents the consolidated data from the iterations of each tested computer according to the number of parallel routines. These numbers express the arithmetic mean of the 40 iterations, each performed using a specific number of parallel routines. The perceptual hashing algorithm used was aHash.

The analyses of each of the results from the simulations conducted on the respective tested computers are as follows:

- **Gray workstation**: the temporal average of this computer executing with only one routine was 253.06 seconds. When the same task was executed with two parallel routines, the execution time dropped to 127.81 seconds, an efficiency of 50%. Increasing the execution to four routines, the efficiency reached 84% (40.25 seconds). Finally, with 128 routines, ChaSAM completed the task in just 10.35 seconds, representing a 95% increase in speed. An important observation is that, after configuring the task execution for 32 parallel routines, the subsequent temporal averages remained practically stable. Between 32 and 512 routines, the observed gain was less than 7%, at the cost of high processor overload.
- **Yellow desktop**: The temporal average of this computer executing with only one parallel routine was 81.97 seconds; with two routines, the execution time decreased by 40% to 49.59 seconds, and with four routines, the efficiency reached 63%, with 29.99 seconds. When configured to be executed with 32 routines, ChaSAM performed the comparison in just 23.20 seconds, or 71% faster. After configuring the execution for 64 parallel routines, the temporal average remained stable. With 256 routines, the temporal average execution time of the task was 22.43 seconds, representing a gain of only 3% compared to execution with 32 routines.
- **Red laptop**: The temporal average of this computer executing with only one routine was 77.73 seconds, the best average among the tested equipment; with two parallel routines, the execution time dropped by 43% to 44.20 seconds; with four routines, the efficiency reached 60% (31.11 seconds); with 32 routines, ChaSAM performed the task in just 25.27 seconds, or 67% more efficient. After configuring the execution for 64 parallel routines, the temporal average remained stable. With 128 routines, the temporal average execution time of the task was 24.80 seconds, representing a gain of less than 2% compared to execution with 32 parallel routines.
- **Green laptop**: The temporal average of this computer executing with only one routine was 259.60 seconds, representing the worst average observed among the tested machines. When configured to execute with two parallel routines, the execution time dropped by 71% to 73.76 seconds; with four routines, the efficiency reached 81% (48.04 seconds); with 32 routines, ChaSAM performed the task in just 39.93 seconds, or 84% more efficiently. After configuring the execution from 32 routines, the temporal average remained relatively stable. With 512 routines, the average execution time was 36.83 seconds, a decrease of only 7% compared to execution with 32 parallel routines;
- **Blue laptop**: The temporal average of this equipment executing with only one routine was 116.56 seconds. When configured to execute with two parallel routines, the comparison time dropped by

TABLE 1. Compilation of simulation data from the computers tested in relation to the number of parallel routines, resulting in the average time of forty simulations per computer.

	Number of Routines x Average (seconds)									
Computer	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**
Gray	253,06	127,81	40,25	16,35	12,01	10,78	10,54	10,35	10,19	10,05
Yellow	81,97	49,59	29,99	24,29	23,56	23,20	22,81	22,60	22,43	22,49
Red	77,73	44,20	31,11	26,18	25,56	25,27	24,89	24,80	24,61	24,41
Green	259,60	73,76	48,04	40,77	39,79	39,93	38,68	38,45	37,72	36,83
Blue	116,56	66,63	50,76	42,30	40,85	40,65	41,16	40,95	40,75	40,52
Purple	178,12	110,26	108,30	107,19	104,69	102,68	101,72	101,60	98,95	98,79

42% to 66.63 seconds; with four routines, the efficiency reached 56% (50.76 seconds). With 32 routines, ChaSAM performed the task in 40.65 seconds, or 65% faster. The temporal average remained stable after configuring 16 parallel routines. With 512 routines, the temporal average execution time of the task was 40.52 seconds, a gain of only 0.31% compared to execution with 32 routines;

- **Purple desktop**: The temporal average of this computer executing with only one routine is 178.12 seconds; with two parallel routines, the execution time decreased by 38% (110.26 seconds); with four routines, the efficiency reached 39% (108.30 seconds). With 32 routines, ChaSAM performed the task in 102.68 seconds, or 42% more efficiently. The temporal average remained stable after configuring ChaSAM to execute with 32 parallel routines. The greatest gain was observed when configuring the execution for 256 parallel routines, where the temporal average execution time of the task was 98.79 seconds, representing a gain of almost 4% compared to execution with 32 routines. Another observation is that this computer showed a gain of only 10% when configured to trigger above two up to 512 parallel routines, while the other tested computers showed gains above 80% when configured with 64 or more routines;

The graph illustrated in Figure 6 depicts, in the form of colored bars, each of the average execution times achieved by the tested computers. Concisely, it is possible to observe that the Gray computer performed the best, while the Purple computer performed the worst. The explanation for this difference is that the former is a robust forensic workstation, whereas the latter is a relatively modest desktop, as shown in Figure 5.

Another observation is that the computers labeled Green and Blue have exactly the same hardware configuration. The difference lies in the operating system; while the former used a version of Linux Ubuntu, the latter used Windows 10. In practice, the Blue computer is 55% more efficient than the Green one. However, Green becomes faster in processing when we configure ChaSAM to use four or more parallel routines. This is because Linux inherently supports multitasking. Ultimately, the Green computer with Linux is about 9% better than the Blue one with Windows operating system.

D. CONCLUSIONS ABOUT PERFORMANCE

The performance simulations conducted in this case study demonstrated that utilizing parallel routines in applications requiring high computational processing power can reduce task execution time by an average of 70%, especially when processing and comparing large volumes of images.

We infer that the performance, i.e., processing speed, increases up to the configuration of 16 parallel routines, as illustrated by the graph in Figure 7. Afterward, the performance stabilizes with minimal gains when further routines are added.

Regarding this, it was observed that the excessive increase in parallel routines does not reduce the execution time of the ChaSAM comparison task, as the algorithm needs to synchronize the result at the end of each comparison, consuming the time gained in processing. Therefore, a high number of parallel routines nullifies the performance gain.

Hence, the simulations indicated that the ideal number for executing the similarity comparison used by ChaSAM ranges between 32 and 64 parallel routines, providing performance for the ''speed'' processing variable.

VI. CHASAM EFFECTIVENESS CASE STUDY

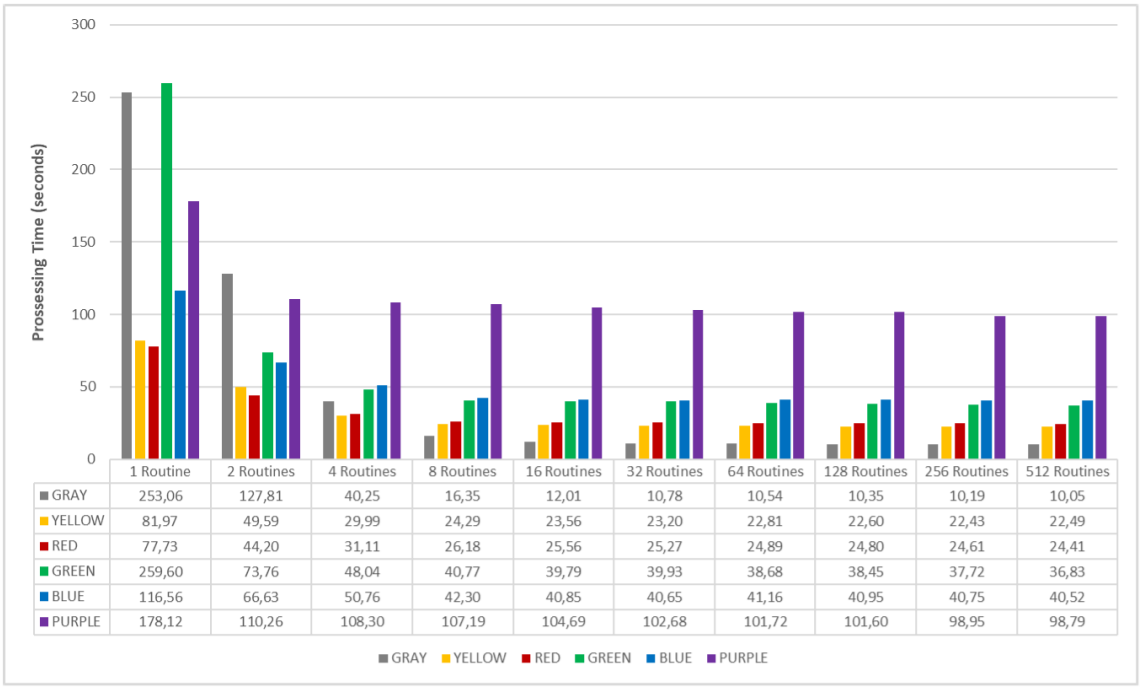
This section demonstrates the case study conducted with ChaSAM Forensics to obtain effectiveness metrics regarding the various established perceptual hash algorithms and the variants presented in Section IV-C: the domiHash and the chHash.

In addition to the domiHash and chHash algorithms, the literature presents other perceptual hash algorithms we employ in this case study to provide reference results. Specifically, we consider the Average hashing (aHash), the Perceptual hashing (pHash), the Difference hashing (dHash), and the Wavelet hashing (wHash) [25].

The answer to the second question of this research is related to the simulation and analysis of the comparison results of each of these algorithms, thus defining which is the most effective in comparing similar images.

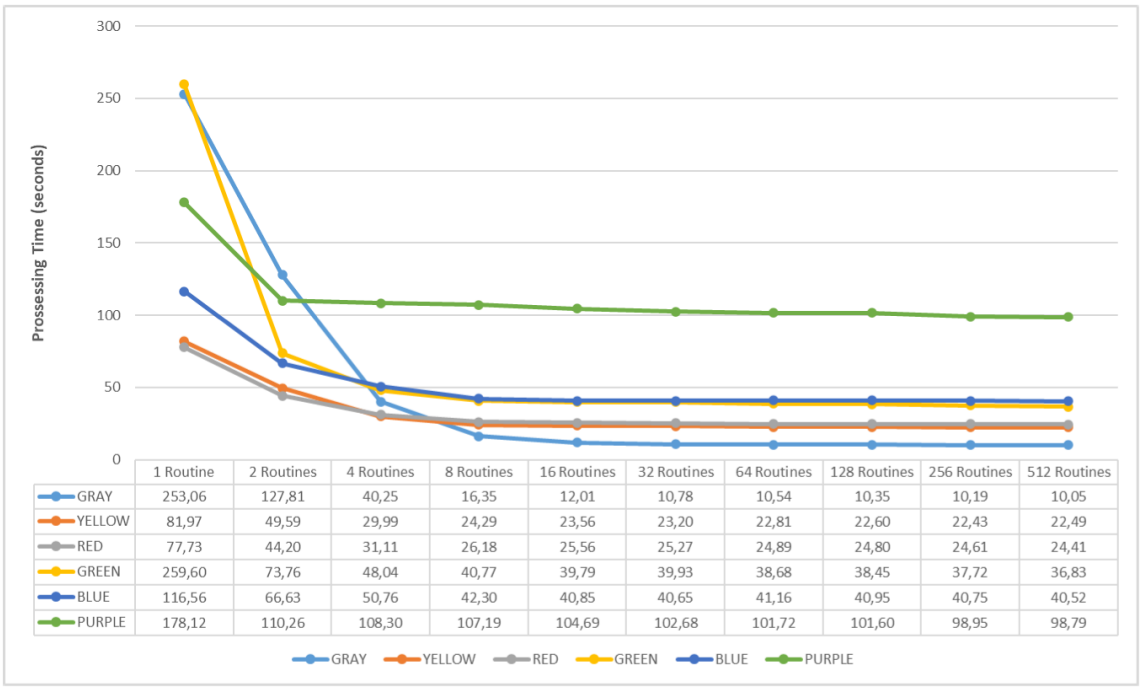
A. ESTABLISHED PERCEPTUAL HASHING ALGORITHMS

According to Samanta and Jain [22], several steps are involved in extracting an image's perceptual hash, including preprocessing, feature extraction, bit quantization, and hash generation. As in Section IV-C, we have organize these steps



	1 Routine	2 Routines	4 Routines	8 Routines	16 Routines	32 Routines	64 Routines	128 Routines	256 Routines	512 Routines
GRAY	253,06	127,81	40,25	16,35	12,01	10,78	10,54	10,35	10,19	10,05
YELLOW	81,97	49,59	29,99	24,29	23,56	23,20	22,81	22,60	22,43	22,49
RED	77,73	44,20	31,11	26,18	25,56	25,27	24,89	24,80	24,61	24,41
GREEN	259,60	73,76	48,04	40,77	39,79	39,93	38,68	38,45	37,72	36,83
BLUE	116,56	66,63	50,76	42,30	40,85	40,65	41,16	40,95	40,75	40,52
PURPLE	178,12	110,26	108,30	107,19	104,69	102,68	101,72	101,60	98,95	98,79

FIGURE 6. Comparison of the simulation data extracted from the tested machines.



	1 Routine	2 Routines	4 Routines	8 Routines	16 Routines	32 Routines	64 Routines	128 Routines	256 Routines	512 Routines
GRAY	253,06	127,81	40,25	16,35	12,01	10,78	10,54	10,35	10,19	10,05
YELLOW	81,97	49,59	29,99	24,29	23,56	23,20	22,81	22,60	22,43	22,49
RED	77,73	44,20	31,11	26,18	25,56	25,27	24,89	24,80	24,61	24,41
GREEN	259,60	73,76	48,04	40,77	39,79	39,93	38,68	38,45	37,72	36,83
BLUE	116,56	66,63	50,76	42,30	40,85	40,65	41,16	40,95	40,75	40,52
PURPLE	178,12	110,26	108,30	107,19	104,69	102,68	101,72	101,60	98,95	98,79

FIGURE 7. Stabilization of the number of parallel routines.

as follows: resizing, conversion, calculation, and perceptual hash generation:

- **Average Hashing (aHash)**: The average hashing algorithm is a perceptual hash algorithm used to represent the average structure of an image by quickly identifying visual similarities between images. To achieve this, the algorithm performs the following tasks for perceptual hash extraction:
  - Resizing: The image is resized to a fixed dimension of 8×8 pixels. This is done to ensure that the image has consistent resolution regardless of its original size;

  - Conversion: The image is converted to grayscale. This reduces the image's complexity, retaining only luminance information;
  - Calculation: The average of the grayscale values of each pixel in the image is calculated. This results in an average value representing the image's average luminosity. Subsequently, each pixel in the image is compared to the calculated average. If the pixel value is greater than the average, it is considered 1; otherwise, it is considered 0. This transforms the image into its binary representation;
  - Hash Generation: The resulting 64-bit binary string is transformed into its hexadecimal representation. This representation is the perceptual hash of the image.
- **Difference Hashing (dHash)**: The difference hashing algorithm is a perceptual hash algorithm that focuses on capturing the differences between neighboring pixels in an image column by column. To achieve this, the algorithm performs the following tasks for perceptual hash extraction:
  - Resizing: Similar to aHash, the image is resized to a fixed dimension, typically 8×9 pixels, one column more than aHash;
  - Conversion: The image is converted to grayscale, reducing its complexity and retaining only luminance information;
  - Calculation: The difference between pixels in each image column is calculated. Each resulting difference is compared to the corresponding pixel. If the pixel value is greater than the difference, it is considered 1; otherwise, it is 0. This transforms the image into its binary representation;
  - Hash Generation: The resulting 64-bit binary string is transformed into its hexadecimal representation. This representation is the perceptual hash of the image.
- **Vertical Difference Hashing (dHashv)**: Vertical difference hashing is a variant of the conventional dHash algorithm. Similar to the original algorithm, it focuses on capturing the differences between neighboring pixels in an image, but the comparison is made row by row. To achieve this, the algorithm performs the following tasks for perceptual hash extraction:
  - Resizing: Similar to aHash, the image is resized to a fixed dimension, typically 9×8 pixels, one row more than aHash;
  - Conversion: The image is converted to grayscale, reducing its complexity and retaining only luminance information;
  - Calculation: The difference between pixels in each image row is calculated. Each resulting difference is compared to the corresponding pixel. If the pixel value is greater than the difference, it is considered 1; otherwise, it is 0. This transforms the image into its binary representation;
  - Hash Generation: The resulting 64-bit binary string is transformed into its hexadecimal representation. This representation is the perceptual hash of the image.
- **Perceptual Hashing (pHash)**: Perceptual hashing is a more complex algorithm than others, as it incorporates concepts of Fourier transform to extract perceptual features from the image. To achieve this, the algorithm performs the following tasks for perceptual hash extraction:
  - Resizing: Initially, the image is resized to a fixed dimension of 32×32 pixels;
  - Conversion: The image is converted to grayscale;
  - Calculation: The Discrete Cosine Transform (DCT) is applied to the image to convert space information into frequency information. The low-frequency part of the spectrum is retained while higher frequencies are cut off, reducing the amount of information to an 8×8-bit matrix. Then, the same concept as the calculation of aHash is applied: calculating the average of the pixels. Subsequently, each pixel in the image is compared to the calculated average. If the pixel value is greater than the average, it is considered 1; otherwise, it is considered 0. This transforms the image into its binary representation;
  - Hash Generation: The resulting 64-bit binary string is transformed into its hexadecimal representation. This representation is the perceptual hash of the image.

Indeed, we observe that the perceptual hash extraction phases follow a similar logic. The exception lies in the treatment of the resized image and the calculation method that each algorithm applies to the corresponding pixels. Table 2 presents the parameters of literature algorithm and the proposed solutions: domiHash and chHash.

B. IMAGES USED IN CHASAM EFFECTIVENESS SIMULATIONS

Nine images were selected from the dataset described in Section IV-B belonging to the ''original'' subfolder which has no modification or tampering.

We selected five more images from the same folder of original files to which we applied modifications such as cropping, watermarks, and text. These transformations simulated the most common alterations found in images during a digital forensic examination, especially when forensic examiners analyze large volumes of files containing scenes of child sexual abuse.

Furthermore, we selected an external file obtained from outside the dataset, though with perceptual characteristics common to some files in the image collection. The purpose of inserting this image was to test the behavior of the different algorithms, knowing that it does not have any corresponding image in the dataset housed in the target folder, as seen in Figure 8.

TABLE 2. Parameter of the algorithms.

Perceptual Hash	Initial Resize	Transformation	Calculation	Final Resize	Binarization
aHash	8×8	Gray scale	Average	8×8	64 bits
dHash	9×8	Gray scale	Horizontal difference	8×8	64 bits
dHashV	8×9	Gray scale	Vertical difference	8×8	64 bits
pHash	32×32	Gray scale	DCT + Average	8×8	64 bits
domiHash	9×9	Gray scale	Diagonal difference	8×8	64 bits
chHash	32×32	Thresholding B&W (114)	DCT + Average	8×8	64 bits

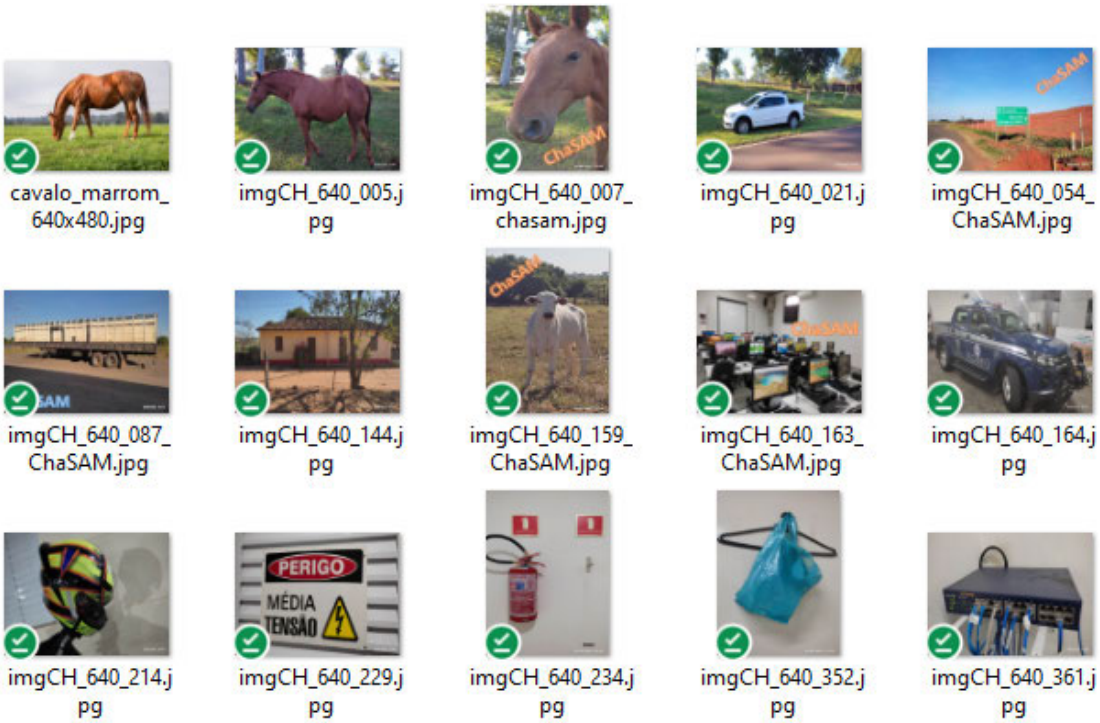


FIGURE 8. Images used in the simulations.

Therefore, 15 images were selected for the simulations, on which we performed the similarity comparison process using each of the algorithms described in Table 2 to assess the efficacy results.

C. SIMULATION ENVIRONMENT AND PARAMETERS

In extracting the perceptual hash from the image, it is resized (final resize) to an 8×8 pixel matrix, generating a 64-bit binary string. Therefore, the Hamming distance discussed in Section II-C plays a fundamental role in our simulations.

The simulations conducted in this study demonstrated that the comparison process within 1 (one) to 20 (twenty) bits of difference between the compared images yielded interesting results.

By using the variation of bits or Hamming distance, we are instructing the algorithm to include in its results not only the identical images (100%) but also those that are similar, varying up to 20 bits of difference, that is, those that are 100% to 68.75% similar. The tests showed that increasing the Hamming distance above 20 bits resulted in many false positive files, rendering its use unfeasible.

A table is built for each of our simulations with the Hamming distance ranging from 1 to 20. The rows of the table represent the number of different bits that limit the comparison of image similarity, and the columns of the table received the transformations applied to the images that formed our dataset of transformed images, as described in Section IV-B.

For each transformation, three sub-columns are introduced to receive the quantification of visually similar (positive) and visually different (negative) images, and a third column contains the total number of files compared by the algorithm.

As a result, we have a Positive column with the total number of files visually similar to the human eye and a False Positive column, which received all images considered similar by the algorithm but visually showed no similarity.

The quantity of False Positives and their percentage representation is a limiting factor for the algorithm's effectiveness, since the higher this number, the less effective the results presented by the algorithm are.

We conducted two types of simulations: the first, which had as a parameter the singular execution of each of the algorithms, and the second, which had the execution in pairs,

to identify which of them or their combination presented the best results.

D. RESULTS OBTAINED IN THE FIRST ITERATION OF THE ALGORITHMS

For each of the 15 images, 20 iterations were performed per algorithm, totaling 120 iterations per image. The 20 iterations per algorithm refer to the Hamming distance that ranged from one to 20 different bits tolerated in the similarity comparison.

Based on the simulations from the first iteration, it was possible to identify that the chHash algorithm exhibited the lowest false positive rate, making it ideal for cleaning the results of files that are visually dissimilar, with an efficiency of 43.75%, while pHash achieved 37.50%.

However, due to their stricter nature, the positive results tend to be more conservative than those of other algorithms. This characteristic led us to select it for application in a second iteration, aiming to provide the analyst with a set of genuinely similar files with a low false positive rate.

E. RESULTS OBTAINED IN THE SECOND ITERATION OF THE ALGORITHMS

Just like in the first iteration, in the second one, we used the same 15 images. In this stage, we aimed to identify the ideal pair of algorithms for the ChSAM Forensics tool.

Hypothetically, an algorithm with many positive files would be a good choice for the first iteration. We observed that this algorithm could be aHash. However, the number of false positives it generates is very high, and not even chHash, considered a rigid algorithm, could "clean up" these results.

Therefore, we decided to create a table with various combinations of pairs of algorithms used in this paper. We performed several dual iterations for each of the six algorithms to determine which one would be the ideal pair. Table 3 is an example of these simulations.

TABLE 3. Example of algorithm pair simulation.

	Image file: imgCH_640_005.jpg - Brown horse			
Hamming = 20	**Positive**	**Negative**	**Total**	**%**
dHash + chHash	27	1	28	96,43
dHash + domiHash	19	2	21	90,48
dHash + pHash	17	8	25	68,00
dHash + dHashV	19	2	21	90,48
dHash + aHash	31	30	61	50,82

In Table 3, we present an example where the dHash algorithm forms pairs with the other five algorithms, including chHash. We then performed the first iteration using dHash and subsequently applied the chHash, domiHash, pHash, dHashV, and aHash algorithms, with both iterations using a Hamming distance limit of 20.

In the example, the first iteration of the dHash algorithm for the image generated 31 positive files, i.e., 31 visually similar images, and 74 files that do not have any similarity (false positives), achieving a positivity rate of 29.52% and efficacy.

When we performed the second iteration by applying the other algorithms in pairs, we found that the efficacy rates increased significantly, as shown in the example table. It can be observed that the dHash + chHash pair showed a positivity rate of 96.43% compared to the initial 29.52% with just one iteration.

For this pair, the second iteration eliminated nothing less than 73 false positive files, and only four positive files were removed from the results, representing less than 5.5%.

The other algorithms, i.e., the pairs dHash + domiHash, dHash + dHash-v, and dHash + pHash, also achieved high true positive rates. Even the aHash algorithm (with poor performance, as demonstrated), when used as a pair in this simulation, increased the initial accuracy percentage from 29.52% to 50.82%.

1) BEST PAIR OF ALGORITHMS

We conducted double iterations with all six tested algorithms to determine the best pair for ChaSAM Forensics. To perform these tests, we used the same set of images described in Section VI-B. We fixed the first algorithm for each image and varied the second one. The graph in Figure 9 displays the efficacy of the dHash + chHash pair compared to the other pairs of tested algorithms, with dHash fixed as the first algorithm in the first iteration.

Analyzing the results of the simulations, we noticed that the chHash demonstrates significant efficiency as a second iteration algorithm. Out of the five possible combinations of pairs of algorithms, it ranked first in 80% of the cases, as listed below:

- The aHash as the first iteration algorithm, had the most efficient pair with the chHash algorithm, where 50% of the simulations using the 15 images achieved the best performance percentage compared to other pairs;
- The dHash as the first iteration algorithm had the most efficient pair with the chHash algorithm, where 81.25% of the simulations using the 15 images achieved the best performance percentage compared to other pairs;
- The dHashv as the first iteration algorithm had the most efficient pair with the chHash algorithm, where 50% of the simulations using the 15 images achieved the best performance percentage compared to other pairs;
- The pHash, as the first iteration algorithm, had the most efficient pair with the dHashv algorithm, where 43.75% of the simulations using the 15 images achieved the best performance percentage compared to other pairs;
- The domiHash as the first iteration algorithm had the most efficient pair with the chHash algorithm, where 75% of the simulations using the 15 images achieved the best performance percentage compared to other pairs;

We also conducted double iterations with chHash as the first algorithm in partnership with the other algorithms. In this case, chHash as the first iteration algorithm, had the most efficient pair with the dHash algorithm, where 62.50% of the simulations using the 15 images achieved the best performance percentage compared to other pairs, as shown in Figure 10:

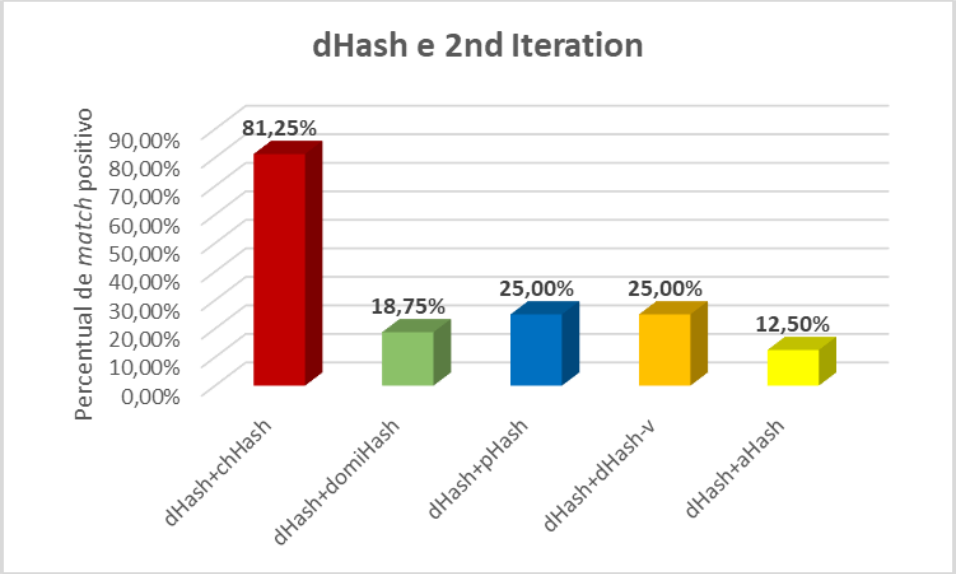


FIGURE 9. The dHash and the other algorithms used in the second iteration.

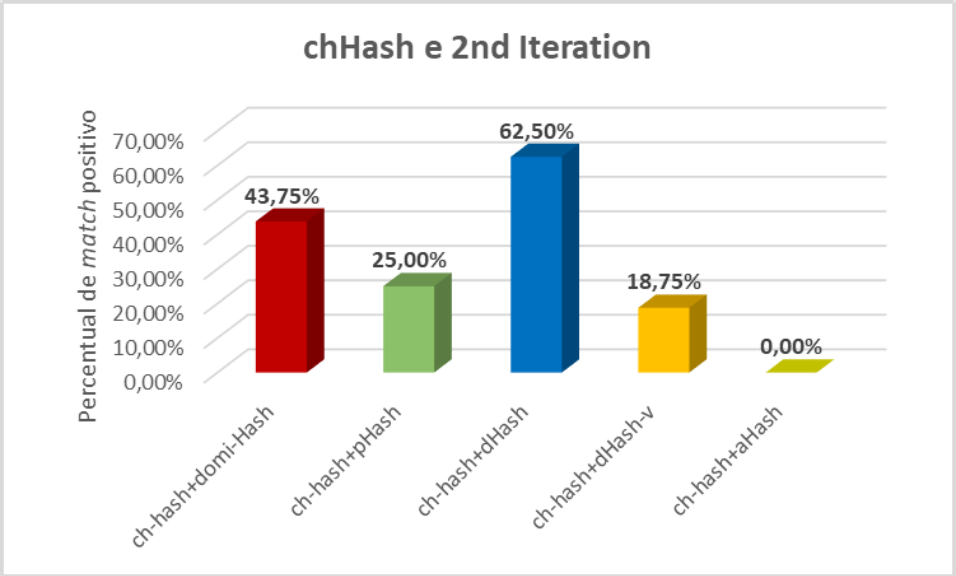


FIGURE 10. chHash and other algorithms used in the 2nd iteration.

Therefore, after analyzing the data and graphs presented throughout this section, we observed that the pair of algorithms dHash and chHash exhibited the best overall performance, achieving over 80% of the top-performing results tested.

F. AI-SYNTHESIZED VERSUS REAL-WORLD IMAGE

To bring veracity to our dataset, an artificial intelligence tool was used to generate images of children to validate ChaSAM. Subsequently, the real police database was used for final implementation.

The artificial image generation tool used was Bing, by Microsoft. The prompt for generating the content of one of the images was: ''Generation of an image of a small child sitting on the floor in a photography studio''. The result was the generation of images of young children, as shown in Figure 11.

Subsequently, we applied to each of the images generated by artificial intelligence the same transformations carried out to the set of dataset files used for the general simulations of this work: blurring, cropping, tilting, inserting a watermark, and other modifications commonly found in tampered images.

The simulations demonstrated that our chHash algorithm was again very effective for detecting similarity between files generated from artificial intelligence, whose content impersonated the image of a child. Like chHash, the well-known pHash algorithm also provided good results. The result of the first iteration using the six algorithms is described in Table 4:

FIGURE 11. Child generated by artificial intelligence.

TABLE 4. Result of the 1st iteration using AI-generated image.

Hamming 20	Positive	Negative	Total of Files	Percentage (%)
aHash	27	194	221	12,22
dHash	25	464	489	5,11
dHash-v	25	50	75	33,33
pHash	25	81	106	23,58
domiHash	29	183	212	13,68
chHash	26	39	65	40,00

In the first iteration, it is possible to observe that chHash identified 65 images as being similar to the little girl generated by artificial intelligence. However, when we visually checked the files, we found that only 26 were similar. This represented a 40% success rate. The most assertive algorithm is triggered.

When we performed the second iteration fixing the chHash algorithm, we obtained the data tabulated in Table 5. Note that the pair of dHash-v + chHash algorithms achieved 100% accuracy, and the other pairs with chHash also presented extremely interesting results.

TABLE 5. Result of the 2nd iteration using AI-generated image.

Hamming 20	Positive	Negative	Total of Files	Percentage (%)
aHash+chHash	25	3	28	89,29
dHash+chHash	24	14	38	63,16
dHash-v+chHash	25	0	25	100,00
pHash+chHash	25	1	26	96,15
domihash+chHash	25	1	26	96,15

If we compare the data from the first iteration with the second, focusing only on the files visually separated as ''positive'', that is, the second columns of both tables, it is possible to observe that the number of files remains relatively stable.

However, one thing that makes the percentage of assertiveness increase a lot is the cleaning of the ''negative'' files with the application of the second iteration. An example of this is the simulation using the dHash-v algorithm (third line of Table 4). This algorithm finds 25 (twenty-five) ''positive'' files in the first iteration. However, it also found 50 files that we checked and concluded that they are not similar. Thus, the accuracy of this algorithm for the tested image was 33.33%.

When we apply the second iteration using chHash, the same 25 ''positive'' files remain (third row of the Table 5). However, in this second iteration, chHash eliminated all ''negative'' files, and thus, this pair of algorithms achieved 100% accuracy, thus representing excellent performance compared to the other pairs of simulated algorithms.

Finally, to test the robustness of the chHash algorithm, we submitted a real CSAM image, collected in one of the police operations, but for legal reasons we cannot replicate it in this paper, and we applied several modifications, creating a set of 40 images. The result of the simulation of the first iteration with a real image is distributed in Table 6:

TABLE 6. Real CSAM file: 1st iteraction.

Hamming 20	Positive	Negative	Total of Files	Percentage (%)
aHash	19	221	240	7,92
dHash	16	10	26	61,54
dHash-V	17	168	185	9,19
pHash	18	41	59	30,51
domiHash	16	111	127	12,60
chHash	13	14	27	48,15

It is possible to observe that the dHash algorithm performed well (61.54%), as did chHash itself with 48.15% of positive results and finally the pHash algorithm with 30.51%. These three algorithms are, in fact, the most effective. Even so, the number of false-positive files was quite significant. Therefore, after the first iteration, we applied the second iteration, whose data is below:

TABLE 7. Real CSAM file: best pair of algorithms.

Hamming 20	Positive	Negative	Total of Files	Percentage (%)
aHash+chHash	18	12	30	60,00
dHash+chHash	16	0	16	100,00
dHash-v+chHash	16	0	16	100,00
pHash+chHash	18	1	19	94,74
domihash+chHash	16	1	17	94,12

After the second iteration, it was possible to observe that chHash together with the dHash and dHash-v algorithms could clean all ''negative'' files from the first iteration.

VII. CONCLUSION

In the context of increasing digital content and the necessity for accurate and efficient forensic tools, this work explores the application of perceptual hashing in computer forensics. The primary motivation was to develop and evaluate methods for comparing manipulated images to identify similarities, particularly in cases involving child sexual abuse material (CSAM). Traditional cryptographic hashes fall short in this context, necessitating the use of perceptual hashes which can detect visual similarities despite manipulations.

Perceptual hashing algorithms, such as average hash (aHash), difference hash (dHash), vertical difference hash (dHash-v), and perceptual hash (pHash), were implemented and evaluated. Additionally, two novel variations, diagonal

difference hash (domiHash) and chHash (based on image thresholding), were developed. These algorithms were integrated into ChaSAM Forensics, a tool designed to efficiently process and compare large volumes of images.

The evaluation involved extensive simulations comparing images with various manipulations using the different hashing algorithms. The results indicated that chHash, particularly when used in conjunction with dHash or dHash-v, was highly effective in identifying similar images while minimizing false positives. The use of parallel processing significantly improved the efficiency, with optimal performance observed using 32 to 64 parallel routines.

Future work will focus on extending ChaSAM Forensics to handle video content, which presents a unique challenge due to the large number of frames per second that need to be processed and compared. Addressing this challenge will require further advancements in both algorithmic efficiency and computational power to maintain the accuracy and speed necessary for forensic investigations.

REFERENCES

[1] R. Martini, "Inclusão digital & inclusão social," *Inclusão social*, vol. 1, no. 1, pp. 21–23, 2005.

[2] M. C. Neri, "Mapa da inclusão digital," CPS/FGV, Rio de Janeiro, Brazil, Tech. Rep. 1, 2012.

[3] *Lei N° 9.998, De 17 De Agosto De 2000*, Diário Oficial da União, Brazil, 2000.

[4] W. Halboob and J. Almuhtadi, "Computer forensics framework for efficient and lawful privacy-preserved investigation," *Comput. Syst. Sci. Eng.*, vol. 45, no. 2, pp. 2071–2092, 2023.

[5] A. G. Barreto and H. Santos, "Deep web investigação no submundo da internet," Brasport, Washington, DC, USA, Tech. Rep. 1, 2019, p. 144.

[6] I. S. D. Almeida, J. C. Santana, and J. Araújo, "Cybercrimes no Brasil: Uma abordagem sobre a tipificação dos crimes virtuais," in *Proc. Anais do XVII Workshop de Informática na Escola (WIE)*, Nov. 2011, pp. 1592–1595.

[7] A. Al-Dhaqm, W. M. Yafooz, S. H. Othman, and A. Ali, "Database forensics field and children crimes," in *Kids Cybersecurity Using Computational Intelligence Techniques*. Springer, 2023, pp. 81–92.

[8] H. D. Santos, J. A. D. Barreto, N. V. Dalarmelina, M. A. Teixeira, and R. I. Meneguette, "Similitude de ocorrências de CSAM na internet e o registro perante às autoridades no estado de São Paulo," in *Proc. Anais do V Workshop de Computação Urbana (CoUrb)*, Aug. 2021, pp. 209–222.

[9] D. Aja-Eke, R. Gillanders, I. Ouedraogo, and W. H. E. Maiga, "Sextortion and corruption," *Appl. Econ. Lett.*, vol. 30, pp. 1–5, Dec. 2023.

[10] S. T. Sydow, "Cybercrimes: A sextorsão chega ao Brasil," Revista dos Tribunais, São Paulo, Brazil, Tech. Rep., 2018.

[11] S. Brasil, "Denúncias de pornografia infantil cresceram 33,45% em 2021, aponta a safernet Brasil," Safernet, São Paulo, Brazil, Tech. Rep. 1, 2021.

[12] K.-S. Choi and H. Lee, "The trend of online child sexual abuse and exploitations: A profile of online sexual offenders and criminal justice response," *J. Child Sexual Abuse*, vol. 32, pp. 1–20, May 2023.

[13] N. Lorenzo-Dus, C. Evans, and R. Mullineux-Morgan, *Online Child Sexual Grooming Discourse* (Elements Forensic Linguistics). Cambridge, U.K.: Cambridge Univ. Press, 2023.

[14] P. M. da Silva Eleutério and M. P. Machado, *Desvendando a Computação Forense*. São Paulo, Brazil: Novatec Editora, 2019.

[15] J. F. P. Faria-Joao. (2018). *Detecção De Imagens Similares: Aplicabilidade De Ferramentas Software Livre De Hash De Similaridade De Uso Geral*. [Online]. Available: https://www.ipog.edu.br/revista-especialize-online/edicao-16-2018-dez/deteccao-de-imagens-similares-aplicabilidade-deferramentas-software-livre-de-hash-de-similaridade-de-usogeral

[16] Q. Zheng, X. Tian, Z. Yu, Y. Ding, A. Elhanashi, S. Saponara, and K. Kpalma, "MobileRaT: A lightweight radio transformer method for automatic modulation classification in drone communication systems," *Drones*, vol. 7, no. 10, p. 596, Sep. 2023.

[17] J. McGarvie, "From hashtag to hash value: Using the hash value model to report child sex abuse material," *Seattle J. Technol., Environ. Innov. Law*, vol. 13, no. 2, p. 4, 2023.

[18] I. Kara, C. Korkmaz, A. Karatatar, and M. Aydos, "A forensic method for investigating manipulated video recordings," *Comput. Fraud Secur.*, vol. 2023, no. 1, pp. 1–12, Jan. 2023.

[19] E. D. Vecchia, D. Weber, and A. Zorzo, "Antiforense digital: Conceitos, técnicas, ferramentas e estudos de caso," in *Minicursos Do XIII Simpósio Brasileiro De Segurança Da Informação E De Sistemas Computacionais*. Ghaziabad: SBC, 2013.

[20] Q. Zheng, P. Zhao, H. Wang, A. Elhanashi, and S. Saponara, "Fine-grained modulation classification using multi-scale radio transformer with dual-channel representation," *IEEE Commun. Lett.*, vol. 26, no. 6, pp. 1298–1302, Jun. 2022.

[21] T. S. Rodrigues, F. Junior, and D. César, "Análise de ferramentas forenses na investigação digital," in *Revista De Engenharia E Tecnologia*. Ponta Grossa, Brazil: UEPG, 2010.

[22] P. Samanta and S. Jain, "Analysis of perceptual hashing algorithms in image manipulation detection," *Proc. Comput. Sci.*, vol. 185, pp. 203–212, 2021.

[23] L. Twenning, H. Baier, and T. Göbel, "Using perceptual hashing for targeted content scanning," in *Proc. IFIP Int. Conf. Digit. Forensics*. New York, NY, USA: Springer, 2023, pp. 125–142.

[24] B. Westlake and E. Guerra, "Using file and folder naming and structuring to improve automated detection of child sexual abuse images on the dark web," *Forensic Sci. Int., Digit. Invest.*, vol. 47, Dec. 2023, Art. no. 301620.

[25] A. dos Santos Silva Torres, "Hash perceptivo de imagens e sua aplicação na identificação de cópia de vídeo," Master's thesis, Pontifícia Universidade Católica de Campinas, Campinas, 2019.

[26] R. Menezes and V. Silva, "Uso da abordagem hash como ferramenta de análise de similaridade entre imagens," in *Proc. Anais da XIX Escola Regional de Computação Bahia, Alagoas e Sergipe*, 2019, pp. 202–207.

[27] S. Vale, "Confira a definição, funcionamento e aplicações do hash, função popular da criptografia," Voitto, Juiz de Fora, Brazil, Tech. Rep. 1, 2020.

[28] *Assegurando a Integridade Dos Dados Com Códigos Hash*, Microsoft, Redmond, WA, USA, 2022.

[29] F. Souza, "Funções hash ou hashing," Medium, San Francisco, CA, USA, Tech. Rep. 1, 2020.

[30] E. Klinger and D. Starkweather, "Phash: The open source perceptual hash library," Upper Austria Univ. Appl., Hagenberg, Austria, Tech. Rep. 1, 2010.

[31] M. Alkhowaiter, K. Almubarak, and C. Zou, "Evaluating perceptual hashing algorithms in detecting image manipulation over social media platforms," in *Proc. IEEE Int. Conf. Cyber Secur. Resilience (CSR)*, Jul. 2022, pp. 149–156.

[32] X. Wang, K. Pang, X. Zhou, Y. Zhou, L. Li, and J. Xue, "A visual model-based perceptual image hash for content authentication," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1336–1349, Jul. 2015.

[33] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.

[34] V. Pless, *Introduction to Theory Error-Correcting Codes*, vol. 48. Hoboken, NJ, USA: Wiley, 1998.

[35] H. Farid, "An overview of perceptual hashing," *J. Online Trust Saf.*, vol. 1, no. 1, pp. 1–22, Oct. 2021.

[36] S. McKeown and W. J. Buchanan, "Hamming distributions of popular perceptual hashing techniques," *Forensic Sci. International: Digit. Invest.*, vol. 44, Mar. 2023, Art. no. 301509.

[37] Q. Zheng, P. Zhao, D. Zhang, and H. Wang, "MR-DCAE: Manifold regularization-based deep convolutional autoencoder for unauthorized broadcasting identification," *Int. J. Intell. Syst.*, vol. 36, no. 12, pp. 7204–7238, Dec. 2021.

[38] Q. Zheng, S. Saponara, X. Tian, Z. Yu, A. Elhanashi, and R. Yu, "A real-time constellation image classification method of wireless communication signals based on the lightweight network MobileViT," *Cognit. Neurodynamics*, vol. 18, no. 2, pp. 659–671, Apr. 2024.

[39] Q. Hao, L. Luo, S. T. K. Jan, and G. Wang, "It's not what it looks like: Manipulating perceptual hashing based applications," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 69–85.
[40] N. Krawetz, "Looks like it," Hacker Factor, Fort Collins, CO, USA, Tech. Rep. 2, 2011.
[41] S. Patra, R. Gautam, and A. Singla, "A novel context sensitive multilevel thresholding for image segmentation," *Appl. Soft Comput.*, vol. 23, pp. 122–127, Oct. 2014.

HERICSON DOS SANTOS is currently pursuing the bachelor's and master's degrees in computer science with the Institute of Mathematical and Computer Sciences, University of São Paulo (ICMC/USP). He is also a Forensic Expert of São Paulo Scientific Police, a Professor of Police Investigation with the Police Academy of São Paulo, a Professor of cyber intelligence with the Ministry of Justice and Public Security, and an Instructor of Child Sexual Abuse Investigation, Child Rescue Coalition (CRC), USA, accredited by Homeland Security Investigation (HSI), United States Embassy. He specialist in networks and telecommunications and forensic computing. He is the author of the book *Deep Web: Investigation in the Underworld of the Internet* and the co-author of the books *Treatise on Technological Investigation and Combating Corruption* and *Technological Criminal Investigation*. He is an active member of the Internet Governance and Crimes Against Children working group, University of São Paulo.

TIAGO DOS SANTOS MARTINS received the degree in internet systems technology. He is currently pursuing the bachelor's degree in computer engineering with the Federal Institute of São Paulo. He is also a Forensic Technical Photographer of São Paulo Scientific Police, an Enthusiast of Open-Source Software and Embedded Systems, and a Systems Developer.

JORGE ANDRE DOMINGUES BARRETO received the bachelor's degree in electrical engineering and the master's degree in computer science from the Institute of Mathematical and Computer Sciences, University of São Paulo (ICMC/USP), Brazil. He is currently an Investigator of the Civil Police of the State of São Paulo, a Professor of cyber intelligence with the Ministry of Justice and Public Security, and an Instructor of Child Sexual Abuse Investigation, Child Rescue Coalition (CRC), USA, accredited by Homeland Security Investigation (HSI), United States Embassy. He is also a Specialist in Police Intelligence. He is the co-author of the books *Treatise on Technological Investigation and Combating Corruption* and *Technological Criminal Investigation*. He is an active member of the Internet Governance and Crimes Against Children working group, University of São Paulo.

LUIS HIDEO VASCONCELOS NAKAMURA received the degree in data processing technology from the Faculty of Technology of Taquaritinga (FATEC-TQ), in 2006, and the master's and Ph.D. degrees in mathematical and computational sciences from the Institute of Mathematical and Computer Sciences (ICMC), University of São Paulo (USP), in 2012 and 2017, respectively. He is currently a Professor and a Researcher with the Federal Institute of Education, Science and Technology, Catanduva Campus. His research interests include distributed systems, the Internet of Things, intelligent transportation systems, the semantic web, ontologies, and cloud computing.

CAETANO MAZZONI RANIERI graduated in computer science from Sao Paulo State University (UNESP) in 2013, and the master's and Ph.D. degrees from ICMC-USP, in 2016 and 2021, respectively. During his Ph.D., he worked as a Visiting Scholar at Heriot-Watt University, Scotland in 2020. He has experience in Activity Recognition, Deep Learning, the Internet of Things, Machine Learning, and Robotics. He is an Assistant Professor at the Institute of Geosciences and Exact Scientes, Sao Paulo State University (IGCE-UNESP). He was a Postdoctoral Research Fellow at the Institute of Mathematical and Computer Sciences, University of Sao Paulo (ICMC-USP), with research focused on artificial intelligence in the context of the Internet of Things.

ROBSON E. DE GRANDE (Member, IEEE) received the Ph.D. degree in computer science from the University of Ottawa, Canada, in 2012. He is currently an Associate Professor with the Department of Computer Science, Brock University, Canada. His research interests include large-scale distributed and mobile systems, cloud computing, performance modeling and simulation, computer networks, vehicular networks, intelligent transportation systems, and distributed simulation systems, actively contributing to these areas. He has served as a Technical Program and the Special Session Co-Chair of several IEEE and ACM-sponsored conferences, including IEEE/ACM DS-RT, ACM MobiWac, ACM DIVANet, and IEEE DCOSS International Workshop on Urban Computing.

GERALDO P. ROCHA FILHO received the master's and Ph.D. degrees in computer science and computational mathematics from the Institute of Mathematical and Computer Sciences, University of São Paulo (ICMC-USP). From 2019 to 2022, he was an effective Professor with the Department of Computer Science, University of Brasília (UnB). In 2022, he requested a vacancy from UnB to UESB. He was a Researcher with the Institute of Computing, UNICAMP, through the Postdoctorate funded by FAPESP. He is currently a Professor with the Department of Exact and Technological Sciences, State University of Southwest Bahia (UESB). In the last five years, he has obtained more than 24 publications in international journals and more than 32 publications in conferences. His research interests include wireless sensor networks, vehicular networks, smart grids, smart homes, and machine learning. He received the FAPESP Scholarship for the master's and Ph.D. studies.

RODOLFO I. MENEGUETTE (Senior Member, IEEE) received the bachelor's degree in computer science from Paulista University (UNIP), Brazil, in 2006, the master's degree from the Federal University of São Carlos (UFSCar), in 2009, and the Ph.D. degree from the University of Campinas (Unicamp), Brazil, in 2013. In 2017, he was a Postdoctoral Researcher with the Paradise Research Laboratory, University of Ottawa, Ottawa, ON, Canada. He is currently a Professor with the University of São Paulo (USP). His research interests include vehicular networks, resource management, flow of mobility, and vehicular clouds.