Black-box Collision Attacks on the NeuralHash Perceptual Hash Function

Diane Leblanc-Albarel, Bart Preneel
KU Leuven, Belgium

***Abstract*—Perceptual hash functions map multimedia content that is perceptually close to outputs strings that are identical or similar. They are widely used for the identification of protected copyright and illegal content in information sharing services: a list of undesirable files is hashed with a perceptual hash function and compared, server side, to the hash of the content that is uploaded. Unlike cryptographic hash functions, the design details of perceptual hash functions are typically kept secret. Several governments envisage to extend this detection to end-to-end encrypted services by using Client Side Scanning and local matching against a hashed database. In August 2021, Apple hash published a concrete design for Client Side Scanning based on the NeuralHash perceptual hash function that uses deep learning. There has been a wide criticism of Client Side Scanning based on its disproportionate impact on human rights and risks for function creep and abuse. In addition, several authors have demonstrated that perceptual hash functions are vulnerable to cryptanalysis: it is easy to create false positives and false negatives once the design is known. This paper demonstrates that these designs are vulnerable in a weaker black-box attack model. It is demonstrated that the effective security level of NeuralHash for a realistic set of images is 32 bits rather than 96 bits, implying that finding a collision requires 2^{16} steps rather than 2^{48}. As a consequence, the large scale deployment of NeuralHash would lead to a huge number of false positives, making the system unworkable. It is likely that most current perceptual hash function designs exhibit similar vulnerabilities.**

***Index Terms*—perceptual hashing, collisions, Client Side Scanning, NeuralHash, CSAM detection**

I. Introduction

The goal of perceptual hash functions is to identify perceptually similar multimedia content such as images, videos, or sounds. Unlike cryptographic hash functions, that produce completely different outputs for even slight changes in input, perceptual hash functions should ensure that *visually* or *audibly* similar inputs produce close or identical hash values.

The primary applications of perceptual hash functions include the identification of copyright violations [27], [48], the detection of problematic content [31], such as Child Sexual Abuse Material (CSAM) [2] or terrorist content [14], and biometric authentication [30]. Comparing hash values is more efficient and allows for comparison without access to the illegal content itself.

The US National Center for Missing and Exploited Children (NCMEC) has reported a significant increase of CSAM detections in recent years [41]. Therefore NCMEC, along with researchers such as Bursztein et al. [10] and Farid [23] have urged the development of automated detection techniques, such as perceptual hash functions, to combat the proliferation of CSAM. In addition, centralized detection methods that were used so far have been hampered by the increased deployment of end-to-end encryption.

In response, the UK [53] and the European Union [21] have launched regulatory proposals to mandate Client-Side Scanning (CSS): CSS performs local detection of illegal content. By scanning content directly on user devices before it is encrypted, users can be reported and authorities can, in principle, take actions before the content is shared [2], [35], [36].

However, CSS has faced heavy criticism from a broad range of groups including academics, industry and NGOs; see Abelson et al. [1] for an overview. Experts have pointed out that the mechanism is a highly intrusive tool for mass surveillance that undermines the fundamental right to privacy, while creating a chilling effect. Moreover, it opens the door to abuse as it may be impossible to verify which content is being detected: once such a system is deployed for CSAM detection, it could be easily extended to other criminal content, to identify whistleblowers or sources of journalists and to detect critics of oppressive regimes. Moreover, it is unclear whether it is effective because of the risk of false positives and the ease by which the mechanism can be circumvented. As governments kept pushing for legislation, members of the EU Parliament [20] and experts have issued open letters and public statements that have warned the public for these ethical and privacy concerns [43], [45].

This paper addresses the need for an evaluation of perceptual hash functions for the detection of illegal content. One of the key criticisms is that perceptual hash functions are ineffective as it is easy to create false positives and negatives; while designing a secure cryptographic hash function is known to be difficult, designing a secure perceptual hash function is much harder. The literature contains many designs but most of them have serious weaknesses. Most deployed designs are not published and rely instead on security by obscurity: by violating Kerckhoffs' principle, this approach impedes public evaluation and makes it impossible to create trust.

This work focuses on a public construction for image hashing developed by Apple to detect CSAM, called NeuralHash [2]. More in particular, it focuses on false positives, that is two images that are perceptually different but with identical hash values. While earlier work [49] has pointed out that it is easy to deliberately create false positives by manipulating images, this work considers a setting in which NeuralHash

would be deployed on a large scale and users who share legitimate images would be falsely accused of sharing CSAM. In its Q&A report [3] on NeuralHash, Apple addresses the concern related to false positives, stating:

> *"Will CSAM detection in iCloud Photos falsely report innocent people to law enforcement? No. The system is designed to be very accurate, and the likelihood that the system would incorrectly identify any given account is less than one in one trillion per year."*

Yet, to our knowledge, no rigorous evaluation of NeuralHash has been conducted to confirm this statement.

In this paper, we analyze the efficiency and accuracy of NeuralHash when used on a large scale. We aim to provide insights into its effectiveness in real-world applications and to highlight potential risks associated with its mass deployment. Our main conclusion is that NeuralHash would lead to a huge number of false positives even if users would not use manipulated images. It is likely that the same conclusions apply to most other current perceptual hash function designs.

Section II provides background information about perceptual hashing and NeuralHash and defines perceptually identical content. Section III analyzes the properties of NeuralHash properties and in particular the collision properties for different types of images. Quantitative results and examples collisions for each type of image are shown in Section IV. Section V discusses the impact on the false positive rate if this function is used at a large scale. Section VI presents the conclusions for large-scale CSAM detection with the current databases.

Disclaimer.: Child sexual abuse and exploitation are serious crimes that need to be addressed in the digital society. We firmly condemn the creation and distribution of CSAM. This paper intends to inform the discussion on the efficacy and implications of using perceptual hashing and Client Side Scanning on a large scale. Our analysis of NeuralHash is not intended as a critique of Apple Inc. or any initiatives aimed at mitigating the spread of CSAM, but it identifies the risk associated with its large-scale deployment. NeuralHash serves as a case study to highlight the critical need for accuracy and robustness in these systems to prevent false positives. We hope this work will encourage further investigation and dialogue aimed at combating CSAM while ensuring a fair balance between efficiency and user privacy.

II. Background

This section defines the notion of *perceptually identical* content and discusses the properties of perceptual hash functions, and their typical building blocks. Next it reviews the most important perceptual hash function designs and presents NeuralHash.

A. Perceptually Identical Content

While content can consist of images, video, audio, 3D objects or 3D-immersive environments, this paper limits itself to the first category which is the target of NeuralHash. Perceptually identical images are those that appear identical or nearly identical for a *human observer*. From this human perception, similarity between images can be characterized by the following factors:

- **Color Consistency:** Images with minor differences in brightness, contrast, or color balance but identical in overall color composition and layout are considered perceptually the same. For example, an image with slightly adjusted brightness levels remains perceptually identical to its original.
- **Structural Similarity:** Images that share the same shapes, edges, and textures, even if subjected to minor geometric transformations such as rotations, translations, or small distortions, are considered perceptually similar. For instance, an image and its slightly rotated version would be perceptually identical.
- **Content Preservation:** Images that maintain the same core content but differ in resolution or compression are also perceptually similar. For example, a high-resolution image and a compressed version with some loss of detail are perceived as the same image.
- **Noise Robustness:** Images with minor noise additions, such as random pixel variations or blurring, which do not significantly alter the perceived content, are considered perceptually the same. For instance, an original image and one with a limited amount of Gaussian noise are perceptually identical.

Mathematically, some of the perceptual similarity elements mentioned above are typically formalized using metrics that attempt to quantify human visual perception. Two widely used metrics are the *Structural Similarity Index Measure*, that measure the similarity between two images based on luminance, contrast, and structure [9], [57] and the *Peak Signal-to-Noise Ratio*, that measures the ratio between the maximum possible pixel value of the image and the number of corrupting pixel [12].

Even if widely used, these metrics cannot detect every case of perceptually similar images. Hence in this paper the perceptually similarity between images is evaluated taking into account the four properties identified above. Section III provides the exact definition used to classify images as similar.

B. Properties

While cryptographic hash functions and perceptual hash functions share some similarities, they have different purposes and thus different properties. Both hash functions efficiently process large inputs and reduce these to short outputs in a deterministic way.

The properties [19], [23] of perceptual hash functions in the context of image hashing are:

- **Pre-image resistance:** Similar to cryptographic hash functions, perceptual hash functions should be one-way, meaning it should be computationally infeasible to reconstruct an original input x from its hash value $H(x)$.
- **Second pre-image resistance:** Given x_1 and its hash value $H(x_1)$, it should be computationally infeasible to

find x_2 an input, perceptually different from x_1 with $H(x_1) = H(x_2)$.

- **Illegitimate-Collision Resistance:** Perceptual hash functions should ensure that perceptually different inputs produce different or sufficiently different hash values. More formally, it should be computationally infeasible to find two perceptually different inputs x_1 and x_2 for which $H(x_1) = H(x_2)$.
- **Accuracy:** Perceptual hash functions should produce the same (or similar) hash values for perceptually identical or very similar inputs. This property ensures that minor variations in the input (such as slight changes in brightness or minor cropping in an image) do not result in significantly different hash values.

C. Perceptual Hashing Process

A first generation perceptual hash functions does not involve deep learning techniques. Examples include pHash [61] and Microsoft's PhotoDNA [48]. These designs typically rely on hand-crafted features and transformations to generate hash values that are robust to minor changes in input.

Deep perceptual hash functions [37], [38], on the other hand, are based on a Machine Learning (ML) model. They involve training deep neural networks to learn feature representations that capture the perceptual similarity of inputs. Deep perceptual hash functions are less used than non-deep ones as they are more recent. Research on deep perceptual hashing includes hashing for image retrieval [62], for label prediction [58], and for CSAM detection [2].

Techniques such as Locality-Sensitive Hashing (LSH) and Binary Reconstructive Embeddings (BRE) are also used in various perceptual hashing functions. LSH maps similar input items to the same hash bucket with high probability [25], [29], while BRE creates compact binary values for similar inputs [34].

Perceptual hash functions typically follow similar sequences of steps to generate hash values [19].

The first step is preprocessing, which prepares the input image by normalizing it into a standardized format. This often involves resizing the image to fixed dimensions, such as (360×360) pixels, and normalizing pixel values to a specific range, for example, $[-1, 1]$. Additional preprocessing techniques may include computing image gradients or converting the image to grayscale.

Deep perceptual hash functions extract features from the preprocessed image using an ML model, This model is typically trained using techniques such as contrastive learning [37], that helps the model differentiating between perceptually similar and dissimilar images.

The next step is hash value extraction, where specific features of the image are extracted to form the hash result. The features selected depend on the hash function application. Common methods for hash extraction include computing the average color of the image, its gradient, or using BRE or LSH.

Finally, the generated hash values are converted to binary strings and possibly compared to another hash value using a hash comparison technique. The Euclidean distance [19], [23], [48] is the most common distance metric for comparisons.

D. Perceptual Hash Functions Designs

Perceptual hash functions have been developed and deployed in various contexts since 2000 [19]. We mention below some of the most notable designs and their applications.

One of the earliest implementations of perceptual hashing for content identification was YouTube's Content ID [27]. Content ID identifies copyrighted material to assist copyright holders in managing their rights.

The 2010 thesis of Zauner [47], [61] provides a detailed introduction to perceptual hash function and introduces the open-source construction pHash, that inspired several other designs.

PhotoDNA [13] was developed by Microsoft and Farid in 2009. It is extensively used for content moderation [48] and on platforms such as Gmail, Twitter, Facebook, Reddit, and Discord for detecting illegal content, particularly CSAM. Despite its widespread use, some successful attacks have been reported (see Section II-F).

eGlyph [14], based on PhotoDNA, was implemented by the Counter Extremism Project (CEP), a nonprofit international policy organization combating extremist ideologies, with the help of Farid. In particular in 2018, eGlyph was used to detect extremist videos on YouTube.

In 2019, Facebook released PDQ and TMK+PDQF [22] as open-source perceptual hash functions based on pHash [16]. These functions are designed to enhance the identification and moderation of prohibited content, in particular CSAM content, across Facebook's platform and beyond. PDQ function is designed for images while TMK+PDQF targets videos.

Apple introduced in 2021 NeuralHash [2], a perceptual hash function specifically designed for CSAM detection. NeuralHash uses a convolutional neural network (CNN) to generate hash values from images, aiming to detect illegal content.

E. NeuralHash

This section presents NeuralHash: it explains why it was created, in which context it is supposed to work and describes briefly the algorithm.

1) Context and Deployment: In August 2021, Apple announced NeuralHash as a key component of its new CSAM detection system [2], [8]. This system was designed to identify known CSAM images stored in iCloud Photos by comparing on-device image hashes to a database of known CSAM hashes provided by child safety organizations.

The deployment of NeuralHash involves integrating the hashing algorithm directly into the iOS operating system. When an image is uploaded to iCloud Photos, NeuralHash generates a hash value representing the image. This hash value is then compared to a database of known CSAM hash values. If a match is found, the image is flagged and, if confirmed, the user is reported to the authorities.

Apple's approach to implementing NeuralHash raised significant public debate and controversy [44]. As a consequence, Apple has officially withdrawn the proposal and postponed the large-scale deployment of NeuralHash; in spite of this, NeuralHash has been added to all Apple devices.

2) Client-Side Scanning: NeuralHash was intended to work conjointly with CSS. We briefly describe the process presented in Figure 1.

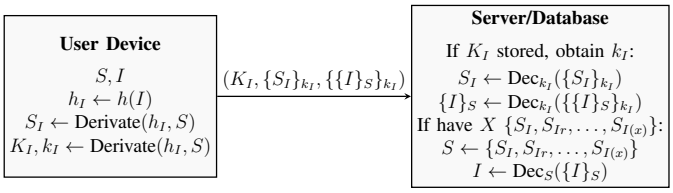


Fig. 1: Process designed by Apple to decrypt CSAM images with CSS

First, the user device generates a long term secret key S, that will be used for threshold secret sharing as explained below. For an image I, the device computes the NeuralHash value h_I and derives K_I and k_I from h_I. Using the key S and the hash result h_I, the device computes a secret sharing key S_I.

The client device then sends to the server the value K_I, the encrypted secret sharing key $\{S_I\}_{k_I}$, and the doubly encrypted image $\{\{I\}_S\}_{k_I}$.

The server stores a database of pairs $(K_I,\ k_I)$ for every problematic image. When receiving the values $(K_I, \{S_I\}_{k_I}, \{\{I\}_S\}_{k_I})$, the server checks whether K_I is in its database; if so, it gets the corresponding k_I. With k_I, the server decrypts $\{S_I\}_{k_I}$ to obtain S_I; it also decrypts the first layer of $\{\{I\}_S\}_{k_I}$ and thus retrieves $\{I\}_S$.

If the server collects at least X shares S_I, it can reconstruct the key S. Once S is known, the server can decrypt any image having a hash value present in its database by using S to decrypt $\{I\}_S$.

The threshold X is defined by Apple, but its numerical value is not disclosed. Apple claims [2] that X is chosen to ensure *"an extremely low probability (1 in 1 trillion) of incorrectly flagging a given account."*

In its technical report, Apple refers to the value we have called K_I as a *"cryptographic header"* derived from h_I. Apple does not provide any information on the size of K_I nor how it is obtained. It also does not specify in its report that the property $K_I \neq K_J,\ \forall\ I \neq J$ is guaranteed. If this property is not guaranteed, then a legitimate image that does not have a NeuralHash value flagged as problematic content can still generate a K_I known by the server and thus be flagged, even if neither the image nor its hash value is flagged. Similarly, a legitimate image with a hash value h_I equal or close to one in the database will lead to the use of a flagged pair $(K_I,\ S_I)$ and thus result in the flagging of a legitimate image.

3) NeuralHash Computation: The NeuralHash algorithm, as deployed on user devices, consists of two primary components: a CNN and a LSH step. The main steps of the hash computation are described bellow.

The algorithm begins by the preprocessing stage that resizes the image to dimensions $(360 \times 360 \times 3)$ within a normalized pixel range of $[-1, 1]$. The resized image is then sent into the embedding CNN.

The CNN produces a vector z of 128 bits. The goal is that for perceptually similar images the vectors z are close and that for perceptually different images they lie far apart.

The LSH step multiplies the vector z of length 128 by a (128×96) matrix B to obtain a real vector y of length 96: $y = B \cdot z$. This step checks the position of the vector z relative to each hyperplane of the matrix B. Each vector is mapped to a specific bucket, with similar vectors (and thus similar images) placed in the same or adjacent buckets.

The final output is a bitstring of length 96. If $y_i > 0$, the corresponding bit is set to 1, otherwise, it is set to 0.

For our experiments, we extracted the neural network as well as the hashing matrix used by NeuralHash from an iPhone with firmware version 16.2. We then used [60] to convert the NeuralHash model into ONNX format, allowing us to run NeuralHash in any script or device.

F. Related Work

Perceptual hashing have been studied in academic papers for over two decades. A comprehensive introduction to perceptual hashing was provided by Farid in 2021 [23], detailing the fundamental techniques and challenges. Additionally, a survey of perceptual hashing techniques and their applications can be found in [19].

Recent research in perceptual hashing has primarily focused on the creation of collisions and information leakage. The creation of collisions involves modifying one of two perceptually different image to produce the same hash value, effectively creating false positives. Information leakage, on the other hand, refers to the potential of inferring information about the input from its hash value.

One prevalent method of attacking perceptual hash functions is through gradient-based hash attacks. These attacks involve imperceptibly altering pixels to achieve specific outputs [11], [15], [17], [50], [51]. Various optimization techniques have been proposed to enhance the effectiveness of these attacks [26], [40], [46]. These attacks generally follow the same principles as classic image processing attacks, manipulating inputs to achieve specific outputs [7], [54], [55], [59], [63].

Some papers have analyzed the resistance of perceptual hash functions to image modifications and the potential to recover original images from their hash values. This includes research on the robustness of perceptual hash functions derived from pHash [18], [31] and attempts to reconstruct original images using Binary Reconstructive Embeddings (BRE) instead of Locality-Sensitive Hashing (LSH) [56].

Recent work has revealed a series of attacks that exploit vulnerabilities in the internal structure of NeuralHash and other perceptual hash functions. Notably, Struppek et al. [49] showed how to create second pre-images for NeuralHash, where input images were imperceptibly modified and published a proof-of-concept implementation (other tools for second pre-images can be found in [6], [33]). Struppek et al. also described classification attacks, in which hash values are used to categorize inputs, achieving a maximum success rate of 52% in some categories. Similarly, attacks on pHash have

shown how images can be manipulated to produce specific hash values [28]. Additionally, partial inversion of PhotoDNA has been achieved using neural networks [5].

Despite this research on attacking perceptual hash functions and NeuralHash in particular, there remains a significant gap in the evaluation of these functions concerning real false positive rates and their performance when approached as black boxes. This gap underscores the necessity of our analysis, which aims to evaluate the performance, accuracy, and false positive rate of NeuralHash in the context of large-scale CSAM detection.

III. Analysis of NeuralHash

This section presents the dataset of images used in the paper and analyses the NeuralHash output for several types of images.

A. Image Sets

We use two different image datasets. The first dataset is made from non-human images extracted from the PASS dataset [4], while the second dataset consists of 202 599 real celebrity face images from the CelebA dataset [39].

Our primary goal is to evaluate whether hashing face images with NeuralHash yields different results compared to non-human images. Specifically, we aim to analyze the number of collisions and the statistical properties of the bits of the hash values.

We selected face images for two main reasons: first, to approximate a use case related to CSAM detection, as such images often contain faces; second, because face images are common on mobile devices, making this a realistic use case scenario.

1) Image Types: The following six image types are used: non-human images, human face images, and human face images with varying levels of blurring (see Figure 7 provided in appendix for examples); each set contains 202 599 images. Throughout the paper, the abbreviation *BF* stands for *blurred faces*. The six types of images are referred to as follows:

- *Non-human*: randomly selected images from the PASS [4] dataset.
- *Non BF*: images from the CelebA dataset [39] without any transformations.
- *Light BF*: CelebA images with light Gaussian blur applied to the entire image.
- *Medium BF*: CelebA images with medium Gaussian blur applied to the entire image.
- *High BF*: CelebA images with high Gaussian blur applied to the entire image.
- *BF only*: CelebA images with Gaussian blur applied only to the face region.

2) Perceptually Identical Images: Among the 202 599 face images in the CelebA dataset, we identified at least 1404 images that appear at least twice or have perceptually similar duplicates, resulting in a total of at least 2823 images among the 202 599 images that should, in theory, share their hash with at least one other image. Examples of identical images and different images that are perceptually similar are presented in Figure 2.

(a) Perceptually identical images (different filter) (b) Perceptually identical images (same person, very close pose)

Fig. 2: Perceptually identical images

B. Statistical Properties of the Hash Values

This section presents experiments performed on the datasets to assess the distribution of the NeuralHash values and to test the independence of the hash value bits.

1) Distribution: We first analyse the distribution and in particular the uniformity and variance of NeuralHash bits values for different types of images.

a) Experiments.: As stated in Section II, the bit distribution of the output of a hash function must be uniform with a small variance to prevent information leakage about the input and to resist (2nd) pre-image and collision attacks. A perceptual hash function is also expected to have a uniform distribution to avoid leaking information about its input.

To compute the average distribution of Neuralhash bits for each image type, we hashed 30 000 randomly selected images from each set. For each image type, we count for each of the 96 bits the number of times the bit is equal to 1 and 0 respectively. The results for the Non-human type and the Non-blur face type are presented in Figure **??**, where the percentage of times each bit is equal to 1 is plotted. The expected result is that each bit is equal to 1 approximately 50% of the time, with a small variance.

For non-human images, the results meet expectations, with the percentage of bits equal to 1 uniformly distributed around 50%. In contrast, for non-blurred human face images, a significant number of bits deviate from being equal to 1, 50% of the time. For non-human images, none of the bits of the hash is less than 40% or more than 60% of the time equal to 1. In contrast, for human face images, 44 bits fall outside this range, with even 2 bits being less than 20% or more than 80% of the time equal to 1.

The results for the four remaining image types are presented in Figure **??**. The resulting hash values are even less uniformly distributed compared to the non-blurred face image type. For lightly blurred images (orange dots), 56 bits fall outside the $40\% - 60\%$ range. For images with only the face blurred (green dots), 59 bits fall outside the range. For medium blurred images (black dots), 67 bits fall outside the range, and for highly blurred images, 70 bits out of the 96 final bits fall outside the $40\% - 60\%$ range.

Additionally, Figure 8 provided in appendix, shows a boxplot of the distribution of bit values for each image type: except the non-human images, the distribution deviates strongly from the expected value.

b) Collision Probability with Independence Hypothesis.: An illegitimate collision corresponds to the event where

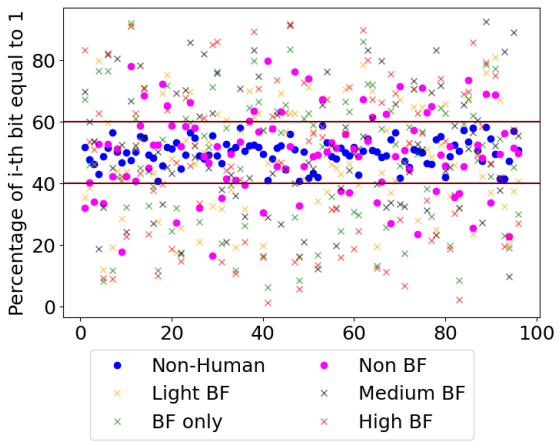


Fig. 3: Bit distribution for each image type

the hash value of two perceptually different images are the same. For regular hash functions, this event should be exceedingly rare. For an uniformly distributed 96-bit hash value, where each bit is independent of the others, the birthday paradox states that the average number of hash values to compute before encountering an illegitimate collision is equal to 2^{48}.

Assuming that all bits of the hash value are mutually independent, and considering the observed distribution for each dataset, Proposition 1 provides the probability that two hash values are identical. Proposition 2 gives the expected number of hash values required before encountering the first illegitimate collision. To make this result more practical, Proposition 3 provides a tight bound on the expected number of hash values necessary before observing the first illegitimate collision.

Proposition 1. *Consider a hash value of n bits and a vector p for which the i-th element p_i denotes the probability that the i-th bit of the hash value is equal to 1 ($0 \leqslant p_i \leqslant 1$). If E_p^n denotes the event that two hash values are equal, then*

$$\mathbb{P}(E_p^n) = \prod_{i=1}^{n} \left(p_i^2 + (1-p_i)^2\right).$$

Proof. Let $H_{i,j}^n$ denote the first j bits of the i-th hash value of n bits, and let a_i^n, b_i^n be the first i bits of two given hash values of size n. E_p^n is thus the event that the two strings a_n^n and b_n^n are equal.

We have:

$$\mathbb{P}(E_p^n) = \sum_{h=H_{i,n}^n} \mathbb{P}(a_n^n = h)\mathbb{P}(b_n^n = h) = \sum_{h=H_{i,n}^n} \mathbb{P}(a_n^n = h)^2.$$

Considering the sum in detail:

$$\begin{aligned}\mathbb{P}(E_p^n) &= \sum_{h=H_{i,n}^n} \mathbb{P}(a_n^n = h \mid h(n) = 1)^2 \\ &+ \sum_{h=H_{i,n}^n} \mathbb{P}(a_n^n = h \mid h(n) = 0)^2 \\ &= (p_n^2 + (1-p_n)^2) \sum_{h=H_{i,n-1}^n} \mathbb{P}(a_{n-1}^n = h)^2.\end{aligned}$$

The result follows by recurrence over the remaining $n-1$ bits. □

Proposition 2. *Denote with $\mathbb{P}(E_p^n)$ the probability that two n-bit hash values with distribution p are equal and define $M = 2^n$. The expected number of hash values to compute before the first collision $\mathbb{E}(D_p^n)$ is equal to:*

$$\mathbb{E}(D_p^n) = 2 + (M+1) \times y^{\frac{M(M+1)}{2}} + \sum_{x=1}^{M-1} y^{\frac{x(x+1)}{2}}$$

with

$$y = 1 - \mathbb{P}(E_p^n).$$

Proof. $\mathbb{P}(E_p^n)$ denotes the probability of a collision between two hash values, and D_p^n denotes the random variable representing the number of hash evaluations required before encountering the first collision. The probability that the first collision occurs after x hash function evaluations is denoted by $\mathbb{P}[D_p^n = x]$.

It is worth noting that $\mathbb{P}[D_p^n < 2] = 0$ and $\mathbb{P}[D_p^n > M+1] = 0$, as there are a maximum of $M = 2^n$ different hash values.

By definition:

$$\sum_{i=1}^{M+1} \mathbb{P}[D_p^n = i] = 1 \quad \texttt{and:} \quad \mathbb{E}(D_p^n) = \sum_{x=2}^{M+1} x \times \mathbb{P}[D_p^n = x].$$

We first define $\mathbb{P}[D_p^n = n]$. The probability $\mathbb{P}[D_p^n = 2]$ is the probability that a collision occurs after 2 hash evaluations, which is simply: $\mathbb{P}[D_p^n = 2] = \mathbb{P}(E_p^n)$. The probability $\mathbb{P}[D_p^n = 3]$ is the probability that a collision occurs after exactly 3 hash evaluations. This is the probability that the first two hash values are different, multiplied by the probability that the third hash value matches one of the first two. Therefore:

$$\mathbb{P}[D_p^n = 3] = (1 - \mathbb{P}(E_p^n)) \times (1 - (1 - \mathbb{P}(E_p^n))^2).$$

Similarly, the probability $\mathbb{P}[D_p^n = 4]$ equals:

$$\mathbb{P}[D_p^n = 4] = (1 - \mathbb{P}(E_p^n))^3 \times (1 - (1 - \mathbb{P}(E_p^n))^3).$$

This approach generalizes to obtain $\mathbb{P}[D_p^n = n]$, the probability that a collision occurs after exactly n hashes with $(3 \leq n \leq M+1)$:

$$\mathbb{P}[D_p^n = n] = \left(\prod_{i=1}^{n-2} (1 - \mathbb{P}(E_p^n))^i\right) \times (1 - (1 - \mathbb{P}(E_p^n))^{n-1}),$$

which simplifies to:

$$\mathbb{P}[D_p^n = n] = (1 - \mathbb{P}(E_p^n))^{\sum_{i=1}^{n-2} i} \times (1 - (1 - \mathbb{P}(E_p^n))^{n-1}).$$

Expanding the sums, we obtain:

$$\mathbb{P}[D_p^n = n] = (1 - \mathbb{P}(E_p^n))^{\frac{(n-2)(n-1)}{2}} - (1 - \mathbb{P}(E_p^n))^{\frac{(n-1)n}{2}}.$$

The expected number of hash values to compute before the first collision is given by $\mathbb{E}(D_p^n)$, which is expressed as:

$$\mathbb{E}(D_p^n) = \sum_{x=2}^{N+1} x \times \mathbb{P}[D_p^n = x],$$

which is equivalent to:

$$\mathbb{E}(D_p^n) = 2 \times \mathbb{P}(E_p^n) + \sum_{x=3}^{N+1} x \times \mathbb{P}[D_p^n = x]\,.$$

Substituting the expression for $\mathbb{P}[D_p^n = x]$ and expanding the sum, we obtain:

$$\begin{aligned}\mathbb{E}(D_p^n) = 2\times\mathbb{P}(E_p^n)+\big(3 \times (1-\mathbb{P}(E_p^n)) - 3 \times (1-\mathbb{P}(E_p^n))^3\big)\\ +4\left((1-\mathbb{P}(E_p^n))^3 - (1-\mathbb{P}(E_p^n))^6\right)\\ +5\left((1-\mathbb{P}(E_p^n))^6 - (1-\mathbb{P}(E_p^n))^{10}\right)+\dots\\ +(M+1)\left((1-\mathbb{P}(E_p^n))^{\frac{(M-1)M}{2}} - (1-\mathbb{P}(E_p^n))^{\frac{M(M+1)}{2}}\right)\,.\end{aligned}$$

Reordering the terms, we then obtain:

$$\begin{aligned}\mathbb{E}(D_p^n) = 2 \times \mathbb{P}(E_p^n) + 2 \times (1-\mathbb{P}(E_p^n)) + (1-\mathbb{P}(E_p^n))\\ +(M+1) \times (1-\mathbb{P}(E_p^n))^{\frac{M(M+1)}{2}}\\ + \sum_{x=4}^{M+1} \big(1-\mathbb{P}(E_p^n)\big)^{\frac{(x-2)(x-1)}{2}}\,.\end{aligned}$$

Simplifying the terms of the sum and shifting the bounds by changing the variable x to $x-2$, we include the term $(1-\mathbb{P}(E_p^n))$ as the first term of the sum:

$$\mathbb{E}(D_p^n) = 2+(M+1)(1-\mathbb{P}(E_p^n))^{\frac{M(M+1)}{2}}+\sum_{x=1}^{M-1}(1-\mathbb{P}(E_p^n))^{\frac{x(x+1)}{2}}.$$

With $y = 1-\mathbb{P}(E_p^n)$, we conclude that

$$\mathbb{E}(D_p^n) = 2 + (M+1) \times y^{\frac{M(M+1)}{2}} + \sum_{x=1}^{M-1} y^{\frac{x(x+1)}{2}}\,.$$

□

The computation of the exact value of $\mathbb{E}(D_p^n)$ is infeasible given the sum over M elements (in the NeuralHash case $M = 2^{96}$). Proposition 3 provides a bound for $\mathbb{E}(D_p^n)$ that is easy to compute.

Proposition 3. *Denote with $\mathbb{P}(E_p^n)$ the probability that two n-bit hash values with distribution p are equal and define $M = 2^n$. The expected number of hash values to compute before the first collision $\mathbb{E}(D_p^n)$ is upper bounded by:*

$$\mathbb{E}(D_p^n) \leq 1 + (M+1) \times y^{\frac{M(M+1)}{2}} + \frac{\theta_2(0; y^{\frac{1}{2}})}{2 \times y^{\frac{1}{8}}}\,,$$

with $\theta_2(0; y^{\frac{1}{2}})$ the Jacobi theta function $\theta_2(z;q)$ with $z=0$ and $q = y^{\frac{1}{2}}$ and with

$$y = 1 - \mathbb{P}(E_p^n)\,.$$

Proof. From Proposition 2 we have:

$$\mathbb{E}(D_p^n) = 2 + (M+1) \times y^{\frac{M(M+1)}{2}} + \sum_{x=1}^{M-1} y^{\frac{x(x+1)}{2}}\,.$$

Note that $\sum_{n=1}^{M-1} y^{\frac{n(n+1)}{2}} = \sum_{n=0}^{M-1} y^{\frac{n(n+1)}{2}} - 1$. As $\sum_{n=0}^{M-1} y^{\frac{n(n+1)}{2}} \leq \sum_{n=0}^{\infty} y^{\frac{n(n+1)}{2}}$ we thus have:

$$\sum_{n=1}^{M-1} y^{\frac{n(n+1)}{2}} \leq -1 + \sum_{n=0}^{\infty} y^{\frac{n(n+1)}{2}}\,. \tag{1}$$

The Jacobi theta function $\theta_2(z;q)$ is defined as follows:

$$\theta_2(z;q) = 2 \times q^{\frac{1}{4}} \times \left(\sum_{n=0}^{\infty} q^{n(n+1)} \cos((2n+1)z)\right)\,.$$

Using $z=0$ and $q = y^{\frac{1}{2}}$ we obtain:

$$\theta_2(0; y^{\frac{1}{2}}) = 2 \times y^{\frac{1}{8}} \times \left(\sum_{n=0}^{\infty} y^{\frac{n(n+1)}{2}}\right)\,.$$

By introducing the Jacobi function in the right hand side of Equation (1), we conclude that $\mathbb{E}(D_p^n)$ satisfies

$$\mathbb{E}(D_p^n) \leq 1 + (M+1) \times y^{\frac{M(M+1)}{2}} + \frac{\theta_2(0; y^{\frac{1}{2}})}{2 \times y^{\frac{1}{8}}}\,.$$

□

Table I is derived from Proposition 3 and the distributions presented in Figure 3. It provides the expected number of hash values to compute before reaching the first illegitimate collision for each image type.

TABLE I: Expected number of hash operations before first illegitimate collision

Type	Non-human	Non BF	Light BF	BF only	Medium BF	High BF
$\mathbb{E}(D_p^n)$	$2^{47.8}$	$2^{43.5}$	$2^{41.1}$	$2^{38.9}$	$2^{37.5}$	$2^{34.1}$

c) Conclusion on Distribution.: From Table I, we conclude that, except for the non-human images, the expected number of hash values to compute before the first illegitimate collision is significantly lower than the theoretical value 2^{48}. For other image types, the expected number of hash values to compute before an illegitimate collision varies between $2^{43.5}$ (non-blurred face) and $2^{34.1}$ (highly blurred face).

It is worth noting that these values assume each bit of the hash is independent of the others. If this independence does not hold, the expected number of hash values before reaching an illegitimate collision decreases substantially.

2) Independence of Bits: In the previous section, we assumed that the bits of the hash values are independent of each other. In this section, we test this independence hypothesis. If the bits are independent, the value of any bit j of a hash value should not provide any information about the value of bit i with $j \neq i$.

a) Simple Matching Coefficient.: The simple matching coefficient (*SMC*) is a statistic ranging from 0 to 1 used to compare the similarity of symmetric binary sample sets. When two samples are identical, the *SMC* is equal to 1. When two samples have no common values (i.e., they are completely

different), the *SMC* is equal to 0. The expected *SMC* of two independent samples is 0.5. It is worth noting that while the *SMC* can be used to verify non-independence, it cannot be used to confirm independence.

For two bits from the same hash value h, Definition 1 provides the value $m_{i,j}(h)$, corresponding to the simple matching value of bits i and j denoted as $h(i)$ and $h(j)$ respectively.

Definition 1. *For a hash value h, $m_{i,j}(h)$ is defined as:*

$$m_{i,j}(h) = \begin{cases} 1 \text{ if } h(i) = h(j) \\ 0 \text{ if } h(i) \neq h(j) \,. \end{cases}$$

Using Definition 1, Definition 2 provides the *SMC* for two bits i and j of a sample of N hash values of n bits.

Definition 2. *Consider a set D_p^n of ℓ ordered hash values of length n. Given h_k, the k-th element of D_p^n, the SMC of bits i and j is defined as:*

$$SMC_{i,j}(D_p^n) = \frac{1}{N} \cdot \sum_{k=1}^{\ell} m_{i,j}(h_k) \,.$$

By definition, $\forall i, j$, $SMC_{i,j}(D_p^n) = SMC_{j,i}(D_p^n)$ and $SMC_{i,j}(D_p^n) = 1$ when $i = j$.

Using Definition 2, the simple matching matrix (*SMM*) of a set D_p^n is built by computing the *SMC* for every pair of bits. We thus use the *SMM* to compute the *similarity* between every pair of bits of hash values from each of the six image types.

b) Bits Sample Similarity.: For each image type, the *SMM* of the set is computed using Definition 2 and the hash values of 30 000 random images per set. A color visualization of the *SMM* for the Non-human and Non-blurred face sets is presented in Figure 9 provided in appendix.

Table II summarizes, for each set, the values of the *SMM* that fall within different ranges, from $0.0 - 0.1$ (indicating that the pair of bits almost always have opposite values) to $0.9 - 1.0$ (indicating that the pair of bits almost always have equal values). For hash values with independent bits, 100% of these values should fall within the $0.4 - 0.6$ ranges.

TABLE II: Similarity score, in percentage, for each image type

SMC \ Type	Non-Human	Non BF	Light BF	BF Only	Medium BF	High BF
$0 - 0.1$	0.0	0.0	0.0	0.0	0.0	0.2
$0.1 - 0.2$	0.0	0.0	0.02	0.2	0.33	1.69
$0.2 - 0.3$	0.0	0.29	1.1	2.7	3.53	6.67
$0.3 - 0.4$	0.18	6.97	10.72	11.64	15.7	15.75
$0.4 - 0.5$	48.6	43.05	38.62	33.93	31.47	26.45
$0.5 - 0.6$	51.18	42.98	38.07	35.68	29.69	29.99
$0.6 - 0.7$	0.04	6.56	10.48	12.7	14.76	13.64
$0.7 - 0.8$	0.0	0.13	0.96	2.89	3.88	7.61
$0.8 - 0.9$	0.0	0.02	0.02	0.26	0.64	2.08
$0.9 - 1$	0.0	0.0	0.0	0.0	0.0	0.11

c) Conclusion on Independence.: Table II shows that for the Non-human type image, only 0.22% of the values fall outside the $0.4 - 0.6$ range. This corresponds to 20 pairs of bits over the 9120 possible pairs. These 20 values are all very close to either 0.4 or 0.6. These values are not sufficient to conclude that bits of non-human images are dependent.

However, for all other types of images, a significant number of values fall outside the $0.4 - 0.6$ range. For the non-blurred face type, 13.97% of the pairs are out of range. For the blurred face types, the percentages of values outside the range goes from 23.3% for the lightly blurred face type to 47.75% for the highly blurred face type. For the blurred face only type, 30.39% of the values are outside the range.

These results indicate that, except for the non-human images, the bits of the hash value are not independent. Consequently, Table I substantially overestimates the expected number of hash values required to obtain an illegitimate collision.

IV. Collision Search

The results from Section III indicate that, for all types of images except the non-human ones, the expected number of hash evaluations before reaching a collision is significantly lower than the theoretical value of 2^{48}.

Therefore, we hashed all 202 599 images of each set, searching for illegitimate collisions between hash values. For each identified collision, we classified it as *legitimate* when the colliding hash values correspond to images identical or perceptually similar, and *illegitimate* when the colliding hash values correspond to images that are perceptually different. In the few cases of colliding hash between images that cannot clearly be identified as perceptually similar or not, we classified the collision as legitimate. Next we determined the number of illegitimate collisions (false positives) and illegitimate non-collisions (false negatives).

A. False Positives

The number of illegitimate collisions for each image type and the number of hash evaluations before the first collision are presented in Table III. Since some collisions involve more than two images (multiple images sharing the same hash), we count the number of illegitimate collisions as the number of images sharing their hash values with at least one perceptually different image.

Table III shows that out of the 202 599 hash values computed for each image type, the non-human type is the only one without any illegitimate collisions. All other image types present several illegitimate collisions. The number of these collisions, given the number of image hashed, is much higher than expected. This number of false positive for sets of only 202 599 images implies that the illegitimate collision rate will be much higher when millions or billions of images will be hashed. Section V presents estimates for the number of illegitimate collisions for large-scale use.

Figure 4 presents an example of a collision obtained for each image type. It is worth noting that for all blurred face images (light, medium, and high), there are hash values shared by three or more perceptually different images. An example of three images sharing a hash value is given for the light blurred type. For the light blurred face set, three images share

the same hash. For the medium blurred face set, eight different hash values are shared by three different images each, and four hash values are shared by four different images each. For the highly blurred face set, instances where three images share the same hash value are very frequent, and two hash values are shared by five different images each.

TABLE III: False positives/negatives for each image type

Type \ Collisions	# illegitimate collisions	# hash bef. collission	# Legitimate collisions	False negative rate
Non-Human	0	–	–	–
Non BF	24	$2^{16.1}$	1559	44.8%
Light BF	25	$2^{15.2}$	1513	46.4%
BF only	12	$2^{16.1}$	1005	64.4%
Medium BF	260	$2^{12.6}$	1864	33.9%
High BF	588	$2^{11.6}$	2333	17.3%

B. False Negatives

As mentioned in Section III-A, at least 1404 distinct images in the CelebA dataset are present at least twice, or have perceptually identical duplicates. In total, at least 2823 images out of the 202 599 should result in legitimate collisions. The minimum false negative rate can thus be computed as follows:

Given m legitimate collisions, the false negative rate in % is at least $100 \cdot (2823 - m)/2823$.

The total number of legitimate collisions and the corresponding false negative rate is provided in Table III.

It is worth noting that the 2823 images expected to legitimately collide is a lower bound. For the medium and highly blurred face sets, as the blur degrades the images, some images that are not perceptually the same before blurring become perceptually the same after blurring, thus increasing the theoretical number of legitimate collisions. Despite this higher number of theoretical legitimate collisions, a significant number of images are still not detected as colliding images, even though they should be.

For the non-blurred face, lightly blurred face, and blurred face only image types, a significant number of perceptually identical images do not have the same hash value. For the blurred face only, this rate reaches 64.4%. These results indicate that the function does not properly detect perceptually identical images. This ratio should be considered alongside the false positive rate. For the blurred face only set, for instance, the number of illegitimate collisions is lower than in other sets, but the number of legitimate collisions is also lower. This indicates that the function is particularly inaccurate when only faces are blurred.

V. Estimation for Large-Scale Applications

We use results presented in Sections III and IV to estimate the implications of the use of functions such as NeuralHash in the context of large scale CSAM detection. We also discuss the impact of a 2023 NeuralHash design update.

A. Model

This section explains how we estimate the number of illegitimate collisions when scaling up the number of images hashed.

1) Approximation: To estimate the number of illegitimate collisions when hashing a larger number of images than in our current sets, we reduce the problem to a uniformly distributed hash values with independent bits problem. This approach provides a worst-case scenario, ensuring that our estimates are conservative.

Given the birthday paradox for a uniform distribution with independent bits, we can estimate the number of images that share the same hash values. Let q denote the number of images that illegitimately share their hash values with at least one other image. For a set of uniformly distributed hash values, where each bit is independent of the others, Equation (2), derived from the birthday paradox, estimates q for a set of N images:

$$q = N \left(1 - \left(\frac{2^n - 1}{2^n}\right)^{N-1}\right). \tag{2}$$

To address the non-uniform distribution and interdependence of bits in NeuralHash case, we reduce the case of NeuralHash to the case of an 'ideal' hash value of $n' < n = 96$ bits, that are uniformly distributed and independent.

To obtain n' from our experiments, we use Equation (2) with q taken from Table III and $N = 201\,195$ unique images[1] in the set. By transforming Equation (2) into Equation (3), we obtain an approximation for the value n':

$$n' \approx -\log_2\left(1 - \left(1 - \frac{q}{N}\right)^{\frac{1}{N-1}}\right). \tag{3}$$

Table IV provides the corresponding n' for each set.

TABLE IV: Size of equivalent uniformly distributed and independent hashes

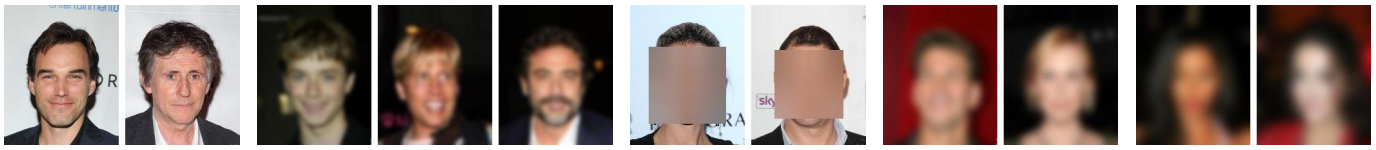
Type	Non BF	Light BF	BF only	Medium BF	High BF
Size of equivalent hash	30.8	30.6	31.9	27.2	26.04

Given a hash size n', we apply the usual birthday paradox (Equation (2)), for any number N of hash values. This provides an approximation of the number of illegitimate collisions for any number of hashes for each image type.

In the following sections, we test our approximation on our set of images to verify that the approximation matches the reality.

2) Verification on 56 Bits: For the medium blurred and highly blurred types, the number of illegitimate collisions for the entire 96-bit hash value is sufficiently high to verify that the approximation given by Equation (3) fits our experiments. For the other three sets, we randomly selected 56 bits of the NeuralHash hash values and then computed the approximate

[1]To ensure a conservative estimate, since at least 1 404 images appear more than once, we use $202\,599 - 1\,404 = 201\,195$ unique images instead of the full set of 202 599.

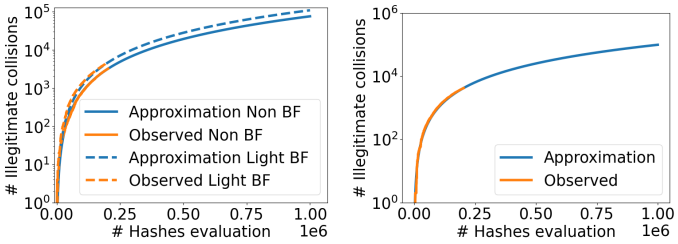


(a) Non BF (b) Light BF (c) BF Only (d) Medium BF (e) High BF

Fig. 4: Collision examples for each image type

number of illegitimate collisions for these 56 bits using Equation (3), with $N = 201\,195$.

By varying N for n' obtained from Equation (3) we compute the number of observed illegitimate collisions (orange curve) alongside the approximate number of illegitimate collisions predicted by Equation (2) (blue curve). The number of observed illegitimate collisions represent the number of images in our non-blurred face, light blurred face, and blurred face only sets that have a hash value with at least one other image sharing the selected 56 bits. The results are presented in Figure 5. The observed number of illegitimate collisions closely matches the theoretical values when using the approximation of a well-distributed and independent hash value. Therefore, we apply this approximation to the entire 96-bit hash result and for every image type in the following section.



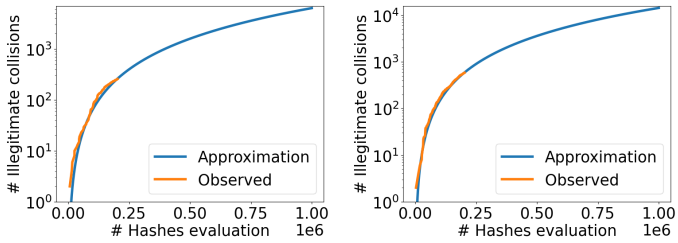
(a) Non BF and Light BF (b) BF only

Fig. 5: Approximated and observed number of illegitimate collisions on 56 bits

B. Estimation of False Positives

To estimate the potential for false positives in a large-scale context, we extrapolated the number of illegitimate collisions for each image type. Figure 6 shows the approximated (blue curve) and observed (orange curve) number of collisions for medium and highly blurred images. For both types, the actual number of false positives observed in our set closely matches the approximation. For other image types, the number of false positives is lower, making the comparison on 96 bits between the approximation and observed collisions less relevant. However, the approximation has been validated in the previous section on 56 bits. We provide figure 10 in the appendix comparing on 96 bits the approximation and the observed collisions for the remain three image types.

The close alignment between the approximated and observed false positives across image types supports the reliability of our method for estimating false positive rates, even at larger scales. The approximation results, derived from Equation (2) and the values of n for each image type (see Table IV), are summarized in Table V, which shows the theoretical percentage of collisions for 1 million and 10 million images.



(a) Medium BF (b) High BF

Fig. 6: Approximated and observed number of illegitimate collisions on 96 bits

TABLE V: Estimated number of false positive rate (in percentage)

Type	Non BF	light BF	BF only	medium BF	high BF
Real 202 599 hashes	0.01	0.01	0.006	0.11	0.25
Extrapol. 1M hashes	0.05	0.06	0.03	0.54	1.2
Extrapol. 10M hashes	0.53	0.6	0.29	5.31	11.44

From Table V, we observe that the percentage of collisions ranges from 0.03% to 1.2% for 1 million hash values. For 10 million hash values, the percentage of illegitimate collisions increases significantly, ranging from 0.29% to 11.44% according to the image type.

These findings suggest a concerning trend: as the number of hash values increases, the likelihood of false positives also rises substantially. This has significant implications for the deployment of perceptual hash functions such as NeuralHash in large-scale CSAM detection systems. The increasing rate of false positives could lead to numerous legitimate users being incorrectly flagged, thereby undermining the effectiveness and reliability of the detection system. The impact of our results on the realistic CSAM detection is provided in the next section.

C. Updated Model

The NeuralHash files deployed on macOS devices have been updated around the end of 2023. The new model uses seeds in fp16 and is trained with fp16 precision (instead of fp32 for the previous model). Apple has not provided any

official communication regarding this change. This update is surprising as Apple announced in December 2022 that it canceled its plan to scan photos on Apple devices for CSAM.

Previous attacks on NeuralHash were conducted in a white-box setting, meaning the attackers had full knowledge of the model. With this new model, which has not yet been reverse-engineered, it is currently impossible to fully understand or manipulate the inner workings of the algorithm, especially the CNN involved.

However, it is still possible to run NeuralHash using the files present on a device through code available on the GitHub project [32], without detailed knowledge of the internal processes. We applied our collision attacks using this method, and the results in Table VI are very similar to those obtained in the previous section.

TABLE VI: False positives for each image type for the fp16 model of NeuralHash

Type	Non BF	Light BF	BF only	Medium BF	High BF
# illegitimate collisions	12	50	90	> 500	> 1000

The results indicate that, except for non-blurred face images where the rate of illegitimate collisions decreased, all other types showed a significant increase in illegitimate collision rates, particularly for medium and highly blurred face images. Therefore, the use of this new model does not alter the conclusions. On the contrary, it tends to exacerbate the issues highlighted.

VI. Collision with CSAM Databases

This section applies our results to estimate the expected number of false positives when CSAM content would be detected at scale using the current CSAM databases.

a) CSAM Hash Database Scale.: According to the NCMEC report [42], *"As of December 31, 2023, NCMEC has added 7,705,865 hashes to the Non-Governmental Apparent Child Pornography Hash-Sharing Initiative. Other non-governmental organizations have submitted an additional 9,802,435 hashes."* Furthermore, the report [52] from the organization Torn states, *"Safer, our all-in-one solution for CSAM detection, uses hashing and matching as a core part of its detection technology. With the largest database of verified hashes (29 million hashes) to match against, Safer can cast the widest net to detect known CSAM."*

From these reports, we can deduce that at least these two organizations have access to databases containing 17.5 million and 29 million hash values, respectively. This indicates that CSAM hash databases contain between 2^{24} and $2^{24.8}$ hash values; in order to be on the safe size, we will use the value 2^{24} in our estimates. Unfortunately, it can be expected that these databases will become larger in the future. For our research, we did not have access to this database of highly sensitive data. Table IV shows that for images containing human faces, the maximum effective size of a NeuralHash string, assuming uniformly distributed and independent bits, is reached for the blurred face only type. Therefore, to remain conservative, we will assume that images with only the face blurred are representative of the images in these databases.

b) Probability of Collision with CSAM Databases.: Let p denote the probability that the hash value of a given image matches one of the stored CSAM hash values. Given that we consider the maximum effective size of a NeuralHash string (32 bits), we have $p < \frac{2^{24}}{2^{32}}$.

Under the assumption that N images are "independent" (which is not completely realistic as many users take several variants of the same image), the probability $\mathbb{P}(X = k)$ of exactly k collisions between user images and the CSAM database follows a binomial distribution:

$$\mathbb{P}(X = k) = \binom{N}{k} p^k (1-p)^{N-k} .$$

The EU Council draft proposal of July 2024 [21] and Apple's reports [2], [8] suggest increasing the number of required flagged images to 2-3 before reporting a user. Given this, the probability of illegitimately reporting a user based on k required flagged images can be expressed as $1 - \sum_{i=0}^{k} \mathbb{P}(X = i)$.

In 2019, European citizens took an average of $N = 597$ selfies per year [24]. Table VII provides an estimate of the number of EU citizens that would be illegitimately reported in the first year of using NeuralHash, based on k, the number of flagged images required before reporting a user. It is important to note that these estimates account only for selfies and apply only to the first year. As users generally accumulate more images over time, the number of flagged citizens would rise significantly in subsequent years.

TABLE VII: Number of EU citizens illegitimately reported

# images to flag before reporting	1	2	3	5	8	10
# reported EU citizens	406M	304M	185M	39M	1.2M	69K

M stands for million K stands for thousand

Our results suggest that the number of false positives in the context of face image hashing is very high. The rate of false positives and the expected number of matches are so high that the proposition of increasing the required number of flagged images to 2 or 3, will not effectively prevent legitimate users from being reported as CSAM owners. Even with this threshold, hundreds of millions of EU citizens would still be falsely reported. Increasing the hash size to 128 bits is also insufficient to reduce the probability to an acceptable level.

c) Conclusion.: Using NeuralHash as an example proposed for a real-world use case, our work demonstrated significant flaws in the application of perceptual hashing for CSAM detection. Specifically, we found that the applying NeuralHash to human faces (blurred or not) results in a very high false positive rate. Given that CSAM content often contains faces, we expect the false positive rate in this context to be similarly high, leading to many users being incorrectly flagged as CSAM owners. Furthermore, NeuralHash was also found to be inaccurate in detecting the same image of a face, exhibiting a very high false negative rate, even for images

without any blurring. These false negatives rate suggest that NeuralHash is not reliable for its intended purpose either.

Our work lead us to conclude that using current perceptual hash function designs similar to NeuralHash for CSAM detection using large-scale Client Side Scanning, as implied by the EU draft proposal for CSAM detection, will result in a huge number of citizens being illegitimately flagged. The current design of NeuralHash is vulnerable to black-box attacks and does not provide the necessary accuracy and reliability to be deployed on a large scale for such a sensitive task.

Therefore, we strongly recommend against using NeuralHash or a similar perceptual hash function in the context of Client Side Scanning for CSAM detection. The potential for harm and the infringement on privacy far outweigh the intended benefits of such systems.

It remains an open problem whether new perceptual hash functions with longer hash values can be designed that provide a better distribution of outputs for relevant inputs such that they resist black-box attacks. The design of perceptual hash functions that can resist white-box attacks is a much harder problem, in particular because Client Side Scanning means that the design cannot be kept secret.

Even if the accurate and reliable perceptual hash functions would be available, the use of Client Side Scanning remains highly problematic due to the risk of function creep and abuse.

References

[1] Harold Abelson, Ross J. Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Jon Callas, Whitfield Diffie, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Vanessa Teague, and Carmela Troncoso. Bugs in our pockets: the risks of client-side scanning. *J. Cybersecur.*, 10(1), 2024.

[2] Apple. CSAM Detection – Technical Summary 2021. Technical report, Apple, 2021. accessed on July 8, 2024.

[3] Apple. Expanded protections for children, frequently asked questions, 2021. accessed: July 8, 2024.

[4] Yuki M. Asano, Christian Rupprecht, Andrew Zisserman, and Andrea Vedaldi. PASS: An ImageNet replacement for self-supervised pretraining without humans. *NeurIPS Track on Datasets and Benchmarks*, 2021.

[5] Anish Athalye. Inverting PhotoDNA. https://www.anishathalye.com/2021/12/20/inverting-photodna/, December 2021.

[6] Anish Athalye. NeuralHash Collider, 2021. accessed: July 7, 2024.

[7] Jiawang Bai, Bin Chen, Yiming Li, Dongxian Wu, Weiwei Guo, Shu-Tao Xia, and En-Hui Yang. Targeted attack for deep hashing based retrieval. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 618–634. Springer, 2020.

[8] Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. The Apple PSI System. Technical report, Apple, Inc., July 29, 2021.

[9] Dominique Brunet, Edward R. Vrscay, and Zhou Wang. On the mathematical properties of the structural similarity index. *IEEE Trans. Image Process.*, 21(4):1488–1499, 2012.

[10] Elie Bursztein, Einat Clarke, Michelle DeLaune, David M. Elifff, Nick Hsu, Lindsey Olson, John Shehan, Madhukar Thakur, Kurt Thomas, and Travis Bright. Rethinking the Detection of Child Sexual Abuse Imagery on the Internet. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2601–2607. ACM, 2019.

[11] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[12] Damon M. Chandler and Sheila S. Hemami. VSNR: A wavelet-based visual signal-to-noise ratio for natural images. *IEEE Trans. Image Process.*, 16(9):2284–2298, 2007.

[13] Microsoft Corporation. New technology fights child porn by tracking its "PhotoDNA"', December 15, 2009. accessed: July 8, 2024.

[14] Counter Extremism Project. How CEP's eGLYPH Technology Works, Dec 08, 2016. accessed: July 8, 2024.

[15] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2196–2205. PMLR, 2020.

[16] Janis Dalins, Campbell Wilson, and Douglas Boudry. PDQ & TMK + PDQF - A test Drive of Facebook's Perceptual Hashing Algorithms. *CoRR*, abs/1912.07745, 2019.

[17] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9185–9193. Computer Vision Foundation / IEEE Computer Society, 2018.

[18] Andrea Drmic, Marin Silic, Goran Delac, Klemo Vladimir, and Adrian Satja Kurdija. Evaluating robustness of perceptual image hashing algorithms. In *40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO, Opatija, Croatia*, pages 995–1000. IEEE, 2017.

[19] Ling Du, Anthony T. S. Ho, and Runmin Cong. Perceptual hashing for image authentication: A survey. *Signal Process. Image Commun.*, 81, 2020.

[20] Cross-Party Letter of Members of the European Parliament Against General Monitoring, 2021. Patrick Breyer, Alviina Alametsä, Rosa D'Amato, Pernando Barrena, Saskia Bricmont, Antoni Comín, Gwendoline Delbos-Corfield, Francesca Donato, Cornelia Ernst, Claudia Gamon, Markéta Gregorová, Francisco Guerreiro, Svenja Hahn, Irena Joveva, Petra Kammerevert, Marcel Kolaja, Moritz Körner, Karen Melchior, Clara Ponsatí, and Mikuláš Peksa.

[21] Proposal for a Regulation of the European Parliament and of the Council laying down rules to prevent and combat child sexual abuse, 2024. accessed on July 6, 2024.

[22] Facebook. The TMK+PDQF video-hashing algorithm and the PDQ image-hashing algorithm, 2020. accessed: July 8, 2024.

[23] Hany Farid. An Overview of Perceptual Hashing. *Journal of Online Trust and Safety*, 1(1), Oct. 2021.

[24] GingerComms for HONOR, December 9, 2019. Research carried out by GingerComms on behalf of HONOR on 2,053 UK adults and 750 adults in France, Germany, Spain, Italy and the Netherlands between 14 – 18 November 2019.

[25] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 518–529. Morgan Kaufmann, 1999.

[26] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[27] Google Inc. How Content ID works, 2007. accessed: July 8, 2024.

[28] Qingying Hao, Licheng Luo, Steve T. K. Jan, and Gang Wang. It's not what it looks like: Manipulating perceptual hashing based applications. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 69–85. ACM, 2021.

[29] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 604–613. ACM, 1998.

[30] Shubham Jain, Ana-Maria Cretu, Antoine Cully, and Yves-Alexandre de Montjoye. Deep perceptual hashing algorithms with hidden dual purpose: When client-side scanning does facial recognition. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 234–252. IEEE, 2023.

[31] Shubham Jain, Ana-Maria Cretu, and Yves-Alexandre de Montjoye. Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 2317–2334. USENIX Association, 2022.

[32] Malcolm Hall Khaos Tian. nhcalc, 2021. accessed: July 23, 2024.

[33] Yannic Kilcher. Neural hash collision creator, 2021. accessed: July 7, 2024.

[34] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Vancouver, British Columbia, Canada*, pages 1042–1050. Curran Associates, Inc., 2009.

[35] Anunay Kulshrestha and Jonathan R. Mayer. Identifying Harmful Media in End-to-End Encrypted Communication: Efficient Private Membership Computation. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 893–910. USENIX Association, 2021.

[36] Ian Levy and Crispin Robinson. Thoughts on child safety on commodity platforms, 2022.

[37] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2475–2483. IEEE Computer Society, 2015.

[38] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. *Int. J. Comput. Vis.*, 127(9):1217–1234, 2019.

[39] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[41] NCMEC. NCMEC's statement regarding end-to-end encryption. Technical report, National Center for Missing and Exploited Children, 2019.

[42] NCMEC. OJJDP-NCMEC-Transparency-CY-2023-Report. Technical report, NCMEC, 2024. accessed: July 22, 2024.

[43] Joint statement of scientists and researchers on EU's proposed Child Sexual Abuse Regulation, 4 July 2023. accessed: July 7, 2024.

[44] An open letter against Apple's privacy-invasive content scanning technology, 2021. accessed: July 23, 2024.

[45] Joint statement of scientists and researchers on EU's new proposal for the Child Sexual Abuse Regulation, 2nd May 2024. accessed: July 7, 2024.

[46] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 372–387. IEEE, 2016.

[47] Evan Klinger & David Starkweather. pHash. The open source perceptual hash library, 2010. accessed on July 22, 2024.

[48] Martin Steinebach. An Analysis of PhotoDNA. In *Proceedings of the 18th International Conference on Availability, Reliability and Security, ARES 2023, Benevento, Italy, 29 August 2023- 1 September 2023*, pages 44:1–44:8. ACM, 2023.

[49] Lukas Struppek, Dominik Hintersdorf, Daniel Neider, and Kristian Kersting. Learning to Break Deep Perceptual Hashing: The Use Case NeuralHash. In *FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022*, pages 58–69. ACM, 2022.

[50] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.*, 23(5):828–841, 2019.

[51] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[52] Torn. How hashing and matching can help prevent revictimization, 2023. accessed: July 22, 2024.

[53] UK Department for Science, Innovation & Technology. Online Safety Act: explainer, 8 May 2024.

[54] Xunguang Wang, Zheng Zhang, Guangming Lu, and Yong Xu. Targeted attack and defense for deep hashing. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada*, pages 2298–2302. ACM, 2021.

[55] Xunguang Wang, Zheng Zhang, Baoyuan Wu, Fumin Shen, and Guangming Lu. Prototype-supervised adversarial network for targeted attack of deep hashing. In *IEEE CVPR*, pages 16357–16366. Computer Vision Foundation / IEEE, 2021.

[56] Yongwei Wang, Hamid Palangi, Z. Jane Wang, and Haoqian Wang. Revhashnet: Perceptually de-hashing real-valued image hashes for similarity retrieval. *Signal Process. Image Commun.*, 68:68–75, 2018.

[57] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.

[58] Dayan Wu, Zheng Lin, Bo Li, Mingzhen Ye, and Weiping Wang. Deep supervised hashing for multi-label and large-scale image retrieval. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR 2017, Bucharest, Romania, June 6-9, 2017*, pages 150–158. ACM, 2017.

[59] Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao. Adversarial examples for hamming space search. *IEEE Trans. Cybern.*, 50(4):1473–1484, 2020.

[60] Asuhariet Ygvar. AppleNeuralHash2ONNX, August 2021. accessed: July 12, 2024.

[61] Christoph Zauner. Implementation and benchmarking of perceptual image hash functions, MsC Thesis, University of Applied Sciences, Hagenberg, Austria, 2010.

[62] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1556–1564. IEEE Computer Society, 2015.

[63] Mingkang Zhu, Tianlong Chen, and Zhangyang Wang. Sparse and imperceptible adversarial attack via a homotopy algorithm. In *Proceedings of the 38th International Conference on Machine Learning, ICML, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12868–12877. PMLR, 2021.

APPENDIX

Examples of each type of image

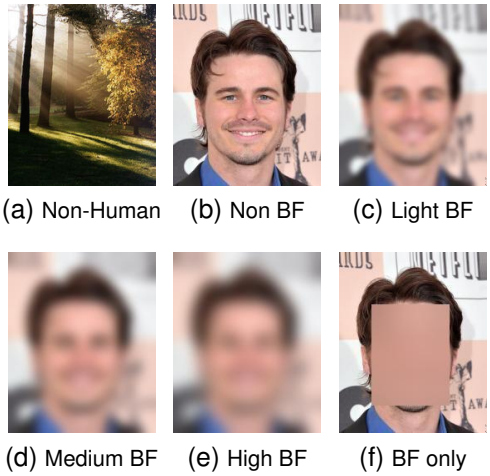


Fig. 7: Example of the 6 types of images used

Distribution of Bit Values

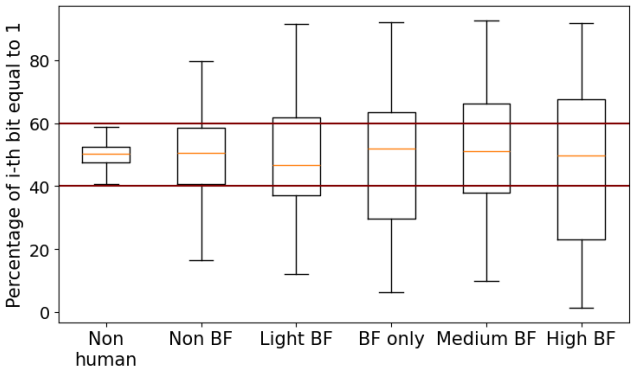


Fig. 8: Disparity of bit values for each type of image

SMM Color Visualization

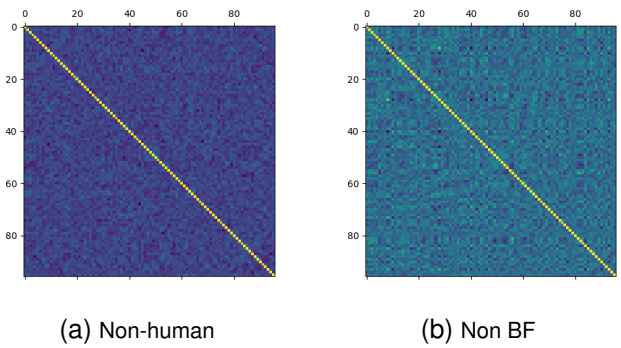


Fig. 9: *SMM* for Non-human and Non BF type

Approximation on 96 bits

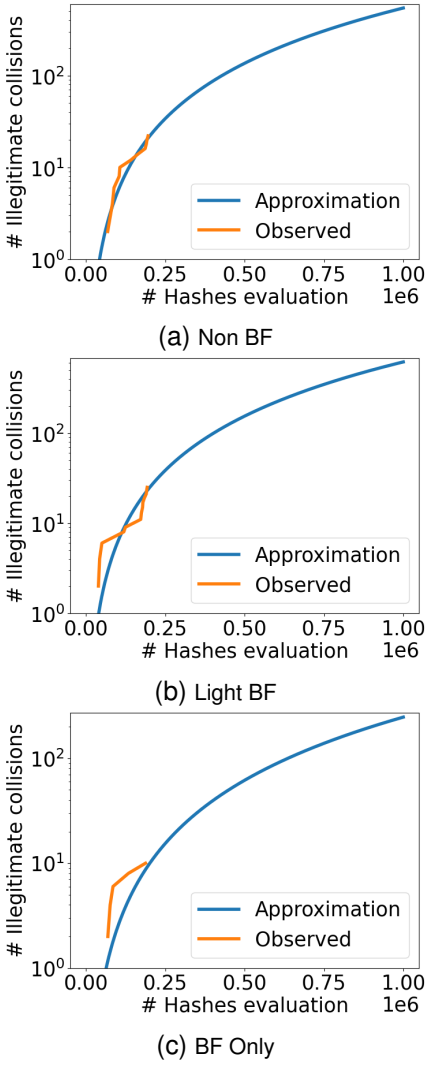


Fig. 10: Approximated and observed number of illegitimate collisions on 96 bits