Inspecting End-to-End Encrypted Communication Differentially for the Efficient Identification of Harmful Media

Tengfei Zheng, Tongqing Zhou, Kai Lu, and Zhiping Cai

***Abstract*— Due to the immense benefits of guaranteeing user privacy, popular messaging platforms have shown enthusiasm for deploying End-to-End Encryption (E2EE). However, E2EE could be misused for bypassing media moderation, opening a shortcut for the viral spreading of harmful media. Private hash-matching techniques are proposed to identify harmful content in E2EE. Unfortunately, the pioneering solution incurs prohibitively high latency due to redundant user-cloud interactions for a private inspection. In this paper, we design *Entbergen* for efficient inspection of E2EE media by differentially handling harmless and harmful ingredients. For this, a novel Private-2D BloOm filter with Fuzzy Query (PBO-FQ) is designed for local, agile, and private media hash matching. It is proposed as the first structure that adapts inverted index and differential privacy (DP) towards seamless integration of sketch and mask encoding. With PBO-FQ, *Entbergen* can instantly filter out harmless media and only pays attention to the small-scale counterparts by scrutinizing them privately based on homomorphic encryption. Security analysis shows that *Entbergen* can effectively fulfil the desired privacy requirements. Extensive evaluations demonstrate that *Entbergen* is sufficiently efficient (w.r.t. computation and communication overhead) for working on mobile devices and can easily scale to real-world inspection with a large database.**

***Index Terms*— End-to-end encryption, privacy-preserving, data abuse, harmful media identification.**

I. Introduction

END-TO-END encryption (E2EE) empowers and limits the decryption right of messages to the expected recipients. It provides an effective privacy defence against illegal surveillance and eavesdropping as well as other human rights abuses [1]. In recent years, E2EE has been adopted in many popular messaging platforms, such as WhatsApp [2], Apple iMessage [3], Facebook Messenger [4], and Telegram [5], serving over 1 billion active users [6].

However, deploying E2EE poses significant challenges to public safety, especially to the highly vulnerable members of our societies, like sexually exploited children. In 2020, over 21 million online child sexual abuse materials (CSAM) were reported to the US National Center for Missing & Exploited Children (NCMEC) [7]. This harmful content is no niche phenomenon and must be carefully identified at the source to prevent it from viral spread on the Internet. One method widely used on social platforms, such as Facebook [8] and Youtube [9], is to check whether the user content is similar to the known-harmful content. Nevertheless, such similarity testing-based content checking would be ineffective in front of E2EE traffic.

In 2020, the UK and EU presented their laws/strategies to become more effective in fighting harmful online content [10], [11]. They especially require messaging platforms to be able to detect and report harmful content transferred in E2EE communication systems. Hence, it is imperative to design technical solutions for enabling the inspection of harmful media, especially without disclosing user privacy; otherwise, these policies would cripple the hard-won privacy victories of E2EE. Such a contradiction led to recent research interests in the automated detection of harmful media in E2EE communication.

Unfortunately, the pioneering solution of Kulshrestha and Mayer [12] attains privacy-preserving inspection at significant efficiency cost. At its core, the proposed private Hamming distance computation technique incurs redundant server-client communications for matching encrypted messages with the harmful hash set. In practice, it will generally cost 27.5 seconds to check one image against a moderate-size hash set (i.e., 2^{20}), easily exhausting user patience with degraded experiences. The follow-up work [13] proposes disclosing certain information in user messages for efficiency gain, which violates the privacy tenet of E2EE.

Intuitively, offloading media moderation from server to distributed clients[1] is helpful to avoid redundant interaction with the server. For this, a straightforward way is to send the homomorphically encrypted hash set[2] for the local check. However, it is still problematic in efficiency because checking one by one involves significant computation. At this point, we hypothesize that **the de facto cause of inefficiency in existing E2EE media moderation is that they indiscriminately check all the traffic**. In practice, although facing a growing amount of

Manuscript received 29 November 2022; revised 7 July 2023 and 30 August 2023; accepted 6 September 2023. Date of publication 13 September 2023; date of current version 25 September 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFF1203001, in part by the National Natural Science Foundation of China under Grant 62072465, Grant 62102425, and Grant U22B2005, and in part by the Science and Technology Innovation Program of Hunan Province under Grant 2022RC3061 and Grant 2023RC3027. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Pedro Comesana. *(Tengfei Zheng and Tongqing Zhou contributed equally to this work.) (Corresponding author: Zhiping Cai.)*

The authors are with the College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China (e-mail: zhengtengfei@nudt.edu.cn; zhoutongqing@nudt.edu.cn; kailu@nudt.edu.cn; zpcai@nudt.edu.cn).

Digital Object Identifier 10.1109/TIFS.2023.3315067

[1]We specify client as the social application, working under the privacy policy, on user devices.

[2]The harmful hash set includes sensitive individual information, which should not be directly disclosed to the public.

harmful content, harmless media dominates social platforms. Performing tedious inspections on tons of healthy media brings no gain to either user privacy or public moderation. Hence, we are motivated to attain efficiency by putting more effort into the harmful ingredient and agilely bypassing the harmless ones during the inspection.

This work then focuses on a **differential inspection** solution for efficient identification of harmful media with a dedicated framework *Entbergen*.[3] Basically, *Entbergen* adapts the linear matching and no-false-negative property of probabilistic data structure (e.g., Bloom filter) to agilely filter out harmless media and perform a deterministic and complex inspection on the small portion of matched (suspicious) contents. Nevertheless, to jointly accommodate efficiency and privacy, we have to handle two aspects of challenges: **1)** The support for approximate matching. Existing probabilistic data structures are only designed for exact matching between a query and the hash set. **2)** Privacy preservation of the harmful hash set that is supposed to be unrevealed. Probabilistic data structures like Bloom filter are susceptible to differential attacks. Namely, given specific auxiliary information, an attacker is able to infer some hash values that have been inserted into the filter [14], partially leaking the harmful hash set of sensitive images.

To address the challenges, *Entbergen* is constructed on a novel structure of Private-2D BloOm filter with Fuzzy Query (PBO-FQ). First, PBO-FQ uses bit-wise masks to statistically ignore the certain number of bits in an image's hash (likened to sampling) and maps a set of random-masked hashes for each image into PBO-FQ. In this way, the filter is enabled to query approximately matched hashes (*Challenge #1*). Second, differential noises are injected into the constructed filter to resist the differential attacks from revealing the harmful hash set and its sensitive pre-images, theoretically ensuring its differential privacy (*Challenge #2*). Third, we attach the encoding indices of the masked hashes (inverted index) to the filter, in the form of a link list, for quickly locating the matched harmful images. Fourth, in order to further minimize costs, our system utilizes a probability data structure called the 2D Bloom filter (2DBF), since it is independent of the number of hash functions. Basically, PBO-FQ integrates the encoding technique [15] into 2D Bloom filter [16] towards privately offloading sensitive hash set to the local zone for efficient media matching. To further enable robust and secure identification in the system view, PBO-FQ introduces the inverted index on encoded hash strings for later online identification and adds differential noise with controllable strength to yield provable privacy protection. Such adaptation and calibration on the initial integration make PBO-FQ a holistic pipeline.

Based on the response of the PBO-FQ, *Entbergen* can instantly judge whether a ready-to-send image is suspicious or not. Latency-aware homomorphic encryption [17] and randomization means are then used to determine whether a suspect is actually a harmful one. In summary, this paper makes the following contributions:

- We design a novel probabilistic data structure PBO-FQ dedicated to harmful media moderation on E2EE data. It supports fuzzy query with tunable matching accuracy and satisfies ϵ-differential privacy.
- We propose the *Entbergen* framework by using PBO-FQ for differential inspection and adapting the Paillier cryptosystem [17] for harmful media identification. *Entbergen* is realized as the result-revealed and result-unrevealed protocols, which provide users with different levels of privacy protection to meet discrepant requirements of harmful media identification in practical deployment.
- We implement and evaluate *Entbergen* in a mobile computing environment. Experimental results indicate that the proposed schemes provide an average of 36x speedup in the execution time, compared with the baseline, and require 0 additional communication costs in cases where the shared media is harmless. Otherwise, for cases where a user sends a harmful image, the proposed schemes can also provide a 9-15x speedup in the execution time and significantly reduce the communication cost by 176-243 times.

[3]The word Entbergen comes from Martin Heidegger's exposition of truth, which means bringing something from concealment into un-concealment.

II. Background and Existing Approaches

A. Harmful Content Moderation

For over a decade, law enforcement and civil society stakeholders worldwide have collected and constructed several known-harmful media hash sets, which embody CSAM [18], [19], [20], missing children material [21], and extremist material [22], totalling millions of images and videos. These hashes are obtained by proprietary perceptual hash functions (PHFs) like PDQHash [23] and PhotoDNA [24], which can map similar media to unique hash representations that have small distances.

Currently, the predominant approaches for inspecting harmful media in large messaging platforms (e.g., Facebook and YouTube) rely on the known harmful hash set. A content moderation party can quantify the similarity between a harmful hash set $\mathscr{B}$ and the shared media's hash w relying on the Hamming distance metric $d_H(\cdot,\cdot)$. The media is identified as harmful if the Hamming distance is less than a fixed similarity threshold δ_H. A typical application of this method is Facebook's ThreatExchange service [25].

However, sending the hash value w to a third party has privacy risk, as w may have a known preimage, and PHF constructions usually leak information about the media content [26]. An adversary at the third party could brutally match a received hash with a pre-image-hash database to guess the original media. Hence, these hash values and the media plaintext are equivalently sensitive in terms of privacy, making hash-based approaches stand in tension with E2EE technologies.

B. Inspecting Media in E2EE

Drawing upon an extensive range of cross-disciplinary literature, Scheffler and Mayer [27] systematically investigate the domain of content moderation in end-to-end encrypted (E2EE) systems. They present a comprehensive content moderation pipeline process and offer valuable insights into potential system designs as well as open challenges in the E2EE context. Their work serves as a valuable resource for researchers seeking guidance in this area. As mentioned in [27], when

examining media in an E2EE context, it is essential to consider the privacy not only of users' shared content but also of the harmful hash set $\mathscr{B}$. The disclosure of $\mathscr{B}$ would cause sensitive content evasion, moderation techniques disclosure, and legal liability to the administrative institutes. To this end, a straightforward method that quantifies the similarity between hashes to alleviate privacy risks is using the 2PC protocols. Existing 2PC protocols [28] for similarity matching can ensure that both the privacy of client media and the harmful hash set is not compromised. Unfortunately, these protocols can not efficiently scale to large databases.

1) Private Membership Computation: The First Scheme: In 2021, Kulshrestha and Mayer described a client-server protocol to explore the technical feasibility of inspecting harmful media for E2EE services for the first time [12]. At the beginning of the protocol, they map similar items in $\mathscr{B}$ to the same bucket using locality-sensitive hashing (LSH) techniques. Private information retrieval is employed to assist the client in privately retrieving a set of encrypted buckets that contain hashes similar to w from the server. The client then performs computation on the encrypted hashes and returns ciphertexts to the server. Finally, the server can learn the result of the matching (equality) tests.

However, the scheme is prohibitively expensive, which requires 27.5 seconds to check the similarity of one image against a database with the size of 2^{20}. While the scheme skips 2PC's drawbacks, it still cannot meet the needs of actual deployment.

2) Bucketing Technique: The Gain And The Loss: An alternative solution focuses on client-side harmful detection [13]. Inspired by the previous work [12], this scheme first gathers possibly relevant images to a bucket and then performs a similarity testing protocol over the bucket. The basic workflow can be described as follows: apply a standard PDQHash to an image, choose a designated number of bit indices randomly, flip each selected bit with a certain probability, and then send these bits and their indices to the server. The server then returns those harmful hashes with a distance smaller than a coarse threshold δ_H to the client. Experimental results show that this proposal significantly reduces the latency and required bandwidth.

Nevertheless, this scheme is insufficiently private, as it explicitly leaks some client data information to the platform to trade off client privacy for efficiency. In front of the advanced attacks (e.g., membership inference) on data sharing, such leakage of clients' information constitutes a series of vulnerabilities on E2EE, breaching its security guarantee to clients.

3) Apple PSI: Detecting Encrypted CSAM in iCloud: Apple recently announced a CSAM detection system based on its deep perceptual hashing algorithm NeuralHash [29]. To detect CSAM, Apple first encodes images in user files uploaded to Apple's iCloud service into perceptual hashes using NeuralHash. A private set intersection will then be performed between the encoded hashes and a database containing CSAM hashes. The matched images are revealed to Apple when the number of matches exceeds a certain threshold. Intuitively, Apple's work can be applied to inspect harmful media in the E2EE communication system by setting the threshold to 1.

However, the protocol only sustains exact hash matching. Namely, the criterion of identifying an image as harmful is $d_H(w, b) = 0$, where b is an item in $\mathscr{B}$. Instead, we focus on approximate hash matching, a more robust criterion that allows $d_H(w, b)$ to be less than a fixed threshold δ_H ($\delta_H \geq 0$). For actual deployment, due to the official NeuralHash model being inaccessible, its latency remains a question to explore in evaluation.

4) Passive Inspection: Another line of work inspects E2EE media passively. It only inspects content reported by the recipients [30]. However, such solutions may not be suitable for CSAM, where the message's recipients can also be the bad actors. Unlike this work [30], *Entbergen* focuses on a proactive inspection strategy to handle general harmful content.

5) Trust In the Hash Set and Implementation: In light of the trust concerns in the hash set and system implementation, Scheffler et al. [31] contribute three novel cryptographic protocols by integrating standard cryptographic primitives such as advanced encryption standard and zero-knowledge proof. These protocols enable three types of public verification for perceptual hash matching systems: 1) prove that each element of the harmful hash set was built from safety groups, 2) demonstrate the absence of specific lawful content in hash sets, and 3) notify users of any false positive matches. Given that our proposal intends to identify harmful media in E2EE efficiently, we state that further orthogonal efforts could be integrated for a more robust system, while the trust concerns are out of the scope of our focused context.

In summary, all these existing approaches use media's similar representations (hash value) as client requests and employ an interactive client-server protocol to calculate the Hamming distance between the shared media's hash and the harmful hashes privately. They shared two drawbacks in practice: On the one hand, these schemes require multiple rounds of interaction between the client and the server to identify a piece of media, resulting in high latency for users to send data, especially considering unstable network delays. On the other hand, they waste lots of computation for weak mobile devices to perform the tedious inspection for every piece of media, most of which are harmless.

C. Probabilistic Data Structure

Bloom filter (BF) [32] is a probabilistic data structure used to represent set membership. Based on the standard BF, researchers have developed a series of variants applied to a wide range of domains. For details, we refer readers to [33], [34], and [35]. The BF variant most relevant to our work is 2-Dimensional Bloom Filter, which is characterized by faster query time, lower memory size, and higher accuracy [16], [36].

1) About Fuzzy Query: Standard BF and its variants do not support fuzzy queries. They ignore the potential hits of approximate items, since uniform and independent hash functions (e.g., MD5 and SHA-1) are used to provide boolean-based answers. Hua et al. [37] propose an LSBF structure that replaces conventional random and independent hash functions with locality-sensitive hash functions [38]. However, it shows poor performance on query time and a false negative rate in

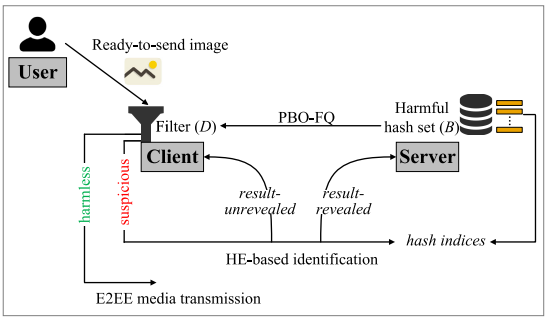


Fig. 1. Overview of the *Entbergen* framework.

high-dimensional Hamming space. Thus, it is unsuitable for harmful media identification with a set of high-dimensional hash vectors as input.

2) About Private BF: Some schemes utilize the false positives caused by potential hash collision to provide privacy protection for the original data [39]. However, an attacker, with sufficient computational resources and auxiliary information, could compromise the protection provided by the collision with high possibilities through reconstructing the filter's content. As a remedy, Xue et al. [14] propose a BF perturbation method by using the DP technique to relieve the weakness of deterministic probability in BF encoding techniques. Their scheme can satisfy the strong guarantee of ϵ-DP. Inspired by their scheme, we adopt DP in PBO-FQ to prevent hash-preimage inference but inject random noise on a 2D BF.

III. System Overview and Preliminaries

A. System Model

Entbergen is built on top of an E2EE messaging system and is invoked when users transmit media through the E2EE channel. The framework of *Entbergen* is shown in Fig. 1, which involves three kinds of entities: a user who communicates with others through E2EE service, a client installed on the user's smartphone device, and a server that provides the content moderation service, which could be the messaging platform or a third party. The server holds a known harmful hash set $\mathscr{B}$. $\mathscr{D}$ denotes the designed filter PBO-FQ, which is the private and compact representation of $\mathscr{B}$ offloaded to the client.

From a high-level view, when a user intends to send media, the client on it first performs a local check by querying the media's hash w to the downloaded filter PBO-FQ, which has the harmful hash set privately encoded. If the media is identified as harmless, it can then be normally transferred to the recipient through the E2EE service. In this case, our framework with the instant local check is believed to incur small latency and will generate no additional communication for inspecting the media. Otherwise, the ready-to-send media is identified as suspicious by the filter and needs to go through a deterministic inspection. Note that we use suspicious media to denote those that do not pass the filter, for that false positives are the by-products of our filter. **The crux here is to instantly categorize a media as harmless and suspicious to avoid the tedious inspection for definitely health media, namely, the differential property of *Entbergen***. Finally, the client and the server 'sentence' the suspects using homomorphic encryption (HE).

The DP and HE means are jointly utilized in *Entbergen* for differentially inspecting harmful media, i.e., benign content merely undergoing local check with the DP-perturbed filter and suspicious content further being scrutinized by HE-based matching. On one hand, DP is used to preserve the privacy of harmful hash set against differential attacks during local check (Algorithm 2 in Sec. IV). For this, utilizing HE in this stage not only cannot resist the differential attacks, but also ruins the intention of using Bloom filter for efficiency. On the other hand, HE is adopted to keep the to-be-verified suspicious hashes unclosed during server-client interactive verification (Fig. 4 in Sec. V). Adding DP noise cannot secure the privacy of a single hash and would mislead the exact distance calculation, thus is not suitable at this stage. Therefore, DP and HE are not replaceable for each other in the design.

We emphasize that by relying on local check and differential inspection, *Entbergen* shows superior scalability regarding media sharing actions and the scale of harmful hash sets. The underlying reason is that the number of harmful user media is always relatively small compared with users' daily and healthy communications. Provided with limited resources on mobile devices, we should pool them on the principal moderation tasks.

B. Security Definition

In this work, we follow the static non-colluding and semi-honest security model and privacy requirements in [12] for harmful media identification.

1) Privacy Goals: Inspecting harmful media in E2EE communications has the following privacy requirements.

a) Content privacy: In light of the hash values of media and media plaintext are equivalently sensitive in terms of privacy, the shared media content and its hash w should be confidential. Revealing these plaintexts to the server stands in tension with E2EE communications.

b) Database privacy: The known harmful hash set $\mathscr{B}$ should be confidential. It is important to keep database privacy as it embodies sensitive and illegal contents, for example, CSAM. Disclosing such content could lead to prejudice, enable malicious users to evade supervision, or even cause legal liability.

c) Result-revealed and unrevealed privacy: We consider two privacy levels for the identification result: result-revealed and result-unrevealed. When the shared media is identified as suspicious, a result (i.e., whether w is similar to an item in $\mathscr{B}$) is presented. In some situations where the sender of media content can be a bad actor, the result should be revealed to the server to assist in taking vigorous actions to help victims and combat criminals (result-revealed). However, revealing the result provides a backdoor to the server to monitor whether clients shared any media on a chosen watch list. To this end, an alternative scheme is to have only the client learn the result, enabling client-side notifications, warning or informing users when they share harmful media (result-unrevealed).

For the above considerations, two schemes are proposed in this paper. However, we do not take a position on which scheme should be deployed given the user privacy and security, human rights, and legal and ethical requirements.

2) Security Model: We clarify the security boundary of our work from the following aspects.

a) The behavior of the entities involved: The involved entities in the proposed schemes are honest-but-curious. Namely, the entities will faithfully perform the designated protocol. However, they could launch passive attacks, i.e., the client and the server would take a stab at accessing the harmful database and the shared media content, respectively. Moreover, the client and the server are non-collusion. The client would not reveal users' media content and its corresponding hash to the server.

Like prior work [12] and [13], *Entbergen* essentially uses the perceptual hashing of media content, so its security boundary is fundamentally limited to semi-honest security. *Entbergen* would fail when a user maliciously damages the perceived characteristics of the media to eliminate the similarity in Hamming space. While the security model appears lax, perceptual hashing matching (PHM) has been proven valuable for identifying harmful content because of its high efficiency and ease of deployment. PHM is widely used by communication services, law enforcement agencies, and child safety groups today [7], [29], [40].

b) The local-check procedure: We highlight that the local-check procedure is encapsulated and executed by default on the client (i.e., the software of the end mobile device), which can be regarded as a secure enclave. Our protocol is agnostic to how to protect the client's security, but a client should be designed to possess the capability of preventing user tampering. Moreover, the client would not disclose any information about users' shared content to the server, as the provisions of privacy policies ban it.

c) Intermediate information of system operation: To check whether a shared image is harmful, the Hamming distance between the shared image and the corresponding matched image is examined, which would be either revealed to the server (in the result-revealed scheme) or the client (in the result-unrevealed scheme).

First and foremost, it is noticed that the vast majority of harmless images can be filtered by local check without Hamming distance producing. Only shared images categorized as suspicious, including true positives and about 0.1% false positives, would undergo the deterministic inspection that produces the Hamming distance.

The intermediate information, i.e., the Hamming distance, increases the system's attack surface. First, membership inference attacks can be conducted by the client/server based on the knowledge of the Hamming distance. Given that this paper aims to infer whether an image belongs to the harmful dataset, a.k.a. membership inference, such attacks are inevitable in the proposed scheme. Second, the client/server can obtain the original harmful/shared image content by inferring the raw hash value. We analyze this attack's difficulty based on scenarios involving true and false positives.

For images Img_a and Img_b, suppose their perceptual hash values are x and y, respectively, and the Hamming distance between x and y is $\triangle$. In practice, a similarity threshold δ is required to be set to determine if two images are similar. Assuming there is a curious client or server $\mathcal{A}$, possessing the information about Img_a, x, $\triangle$, and δ, attempts to obtain the information about y to infer Img_b.

- When Img_a is a true positive, $\triangle \leq \delta$. $\mathcal{A}$ then can immediately assert that $Img_b = Img_a$, due to the perceptual hash function's robustness.
- When Img_a is a false positive, $\triangle > \delta$. If adversary $\mathcal{A}$ tries to infer the context of Img_b, $\mathcal{A}$ requires obtaining the exact y. Let P be the probability of guessing y, then P can be computed as follows.

$$P = \frac{1}{C_l^{\triangle}} = \frac{\triangle!(l-\triangle)!}{l!}, \tag{1}$$

where ! denotes factorial operation, l denotes the length of perceptual hash. Since $\triangle > \delta$, then, we can further obtain,

$$P < \frac{\delta!(l-\delta)!}{l!}. \tag{2}$$

We acknowledge that in cases where l and δ are small, $\mathcal{A}$ can guess y through the Hamming distance. However, in light of accuracy, l is typically greater than or equal to 128, δ is usually set to 10% of l. In such a parameter setting, P is almost zero.

Based on the above analysis, we can deduce that inferring the content of Img_b solely based on the Hamming distance is challenging unless Img_a and Img_b exhibit inherent similarity.

C. Technical Preliminaries

The technical tools employed to build the proposed scheme consist of sampling, the 2-dimensional Bloom filter, DP, and the Paillier cryptosystem. We present the implementation details of the above techniques in Appendix to make this paper self-contained.

IV. PBO-FQ for Local Check

By incorporating encoding with 2D Bloom filter and introducing Laplace-based DP and inverted index, PBO-FQ is proposed that reconciles fuzzy query and privacy properties at low cost. Designing PBO-FQ as a cohesive system necessitates tightly integrating all these components into a functional pipeline, making it challenging. This section describes the design rationale and detailed construction process of a novel probabilistic structure of PBO-FQ.

A. Basic Idea

1) For Fuzzy Query: In inspecting media in E2EE, the criterion for a media M to be considered harmful is $d_H(w, b_i) \leq \delta_H$. Hence, PBO-FQ should support fuzzy query, i.e., "*Is w approximate to an item in $\mathscr{B}$?*". Knowing that BF does not support approximate matching, we propose to soften the exact matching of a media to a set of approximate matching of its subsampling one. To this end, we first encode each item b_i in $\mathscr{B}$ to a set $\mathscr{S}_i \leftarrow \mathsf{Encode}(b_i)$, and then combine each generated set to attain $\mathscr{S}$ (where $\mathscr{S} = \mathscr{S}_1 \cup \mathscr{S}_2 \ldots \cup \mathscr{S}_n$) to a BF. Similarly, for an incoming item w, we can encode it to a set $\mathscr{S}'$. Then we determine whether w is similar to any item in $\mathscr{B}$ by checking how many items of $\mathscr{S}'$ are in the BF.

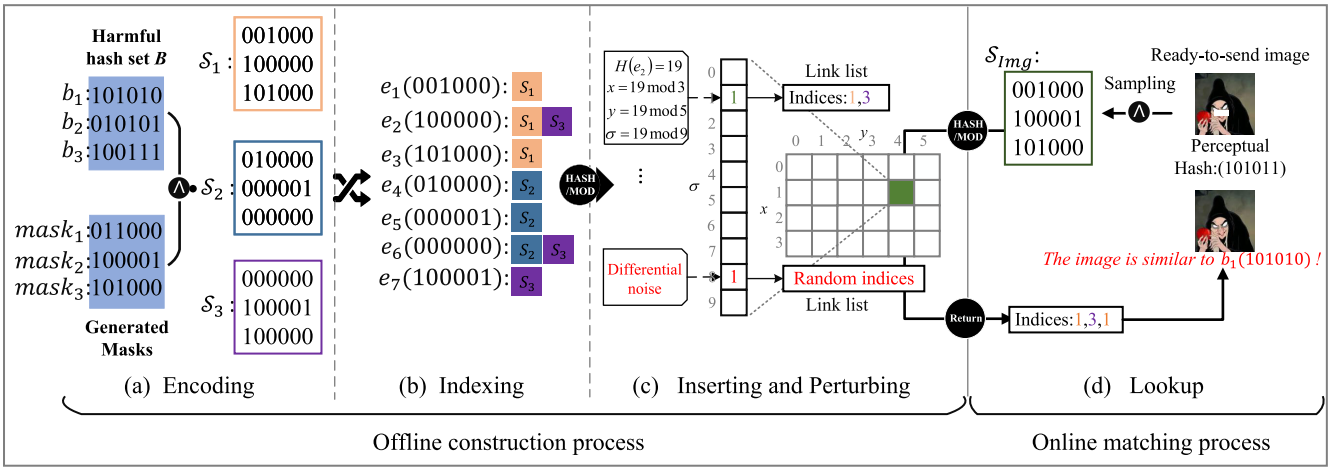


Fig. 2. An illustrative example on the offline construction ((a)-(c)) and the online using process ((d)) of PBO-FQ with $|\mathscr{B}| = 3$, $t = 2$, $T = 3$, $x = 3$, $y = 5$, and $\sigma = 9$. $H(\cdot)$ is a hash function (e.g., SHA256), and mod denotes the modular operation.

If more than t items of $\mathscr{S}'$ are in the BF, then we denote w and a certain item $b_i \in \mathscr{B}$ is hamming-close, similar to the criterion used in [15]. Hitherto, we can achieve the fuzzy query function.

2) For Privacy Preservation: Such a manner cannot meet database privacy introduced in Section III-B. Although the probabilistic data structure of BF can provide certain privacy protection by sacrificing matching accuracy, the privacy guarantee is limited [14]. W.h.p., the filter's content (i.e., hash values) could be reconstructed with specific auxiliary information, then subsequently the confidential pre-images of these hashes could be compromised by brute force if an attacker has sufficient computational resources. Targeting this privacy requirement, a DP perturbation method [14] is applied to ensure that the filter satisfies ϵ-DP.

3) Questioning the Suspects: Adding perturbation noise into the PBO-FQ filter results in an unexpected increase in false positives (i.e., identifying health media as suspicious). Notifying users or blocking sharing actions when they transmit harmless will seriously damage users' online social experience. PBO-FQ is then responsible for providing a way to verify whether w is actually harmful (i.e., questioning) when it is identified as suspicious. For this, we record each item's indices in a linked list corresponding to the bit-position where the item is mapped for locating the matched item, thus quickly questioning the suspects in the follow-up operation (See Section V). However, our experiments found that the filter may return multiple indices, making hash-locating errors. This is caused by cross-overlap between sampling sets $\mathscr{S}_i$, as illustrated in the following toy example. To address this problem, we further compare the number of occurrences of each returned index with the threshold value t to locate the matched item correctly.

Example. Suppose hashes $b_1 = 10101$ and $b_2 = 01010$ are the items in $\mathscr{B}$, while $mask_1 = 01100$, $mask_2 = 10001$, and $mask_3 = 01010$ are three generated masks. b_1 and b_2 can be encoded to hash set $\mathscr{S}_1 = \{00100_1, 10001_1, 10000_1\}$ and $\mathscr{S}_2 = \{01000_2, 00000_2, 01010_2\}$, respectively. The subscript represents the index of each item. Then, the items in $\mathscr{S}_1$ and $\mathscr{S}_2$ are mapped to a BF. Assume $\delta_H = 1$ and $t = 2$ in the filter settings. When receiving a query with item $w = 00110$, it can be encoded to the set $\mathscr{S}' = \{00100, 00000, 00010\}$ using the identical masks. We will locate b_1 and b_2 after querying each item in $\mathscr{S}'$, but w is not similar to either of them.

4) Reducing Cost: Encoding an item to a set $\mathscr{S}_i$ results in an increase in the number of time-consuming hash computations to query an item, i.e., from $O(k)$ to $O(kT)$ (suppose k is the number of hash functions used in BF). To further reduce the query cost, our final design replaces the BF with 2DBF [16] as 2DBF is independent of the number of hash functions compared to any other variants. We thus can reduce the number of time-consuming hash computations from $O(kT)$ to $O(T)$.

B. Construction of PBO-FQ

We use $\mathscr{D}$ to denote an instantiated PBO-FQ. $\mathscr{D}$ is organized by a tuple $(\mathbb{B}_{x,y}, \mathbb{L})$, where $\mathbb{B}_{x,y}$ is a 2-dimensional array and $\mathbb{L}$ is a linked list set. $\mathbb{B}_{x,y}$ contains $x \times y$ cells and each cell is σ-bits. $\mathbb{L}$ is a set of empty linked lists, and the header of each linked list corresponds to a bit in $\mathbb{B}_{x,y}$.

$\mathscr{D}$ is composed of three subroutines: Inserting (Algorithm 1) is invoked upon arrival of each item to insert it into $\mathscr{D}$; Perturbing (Algorithm 2) focuses on protecting the privacy of $\mathscr{B}$; Lookup (Algorithm 3) is invoked to query whether a new item is similar to any item in set $\mathscr{B}$. An example of the offline construction of PBO-FQ ((a)-(c)) and the online using ((d)) process with $|\mathscr{B}| = 3$, $t = 2$, $T = 3$, $x = 3$, $y = 5$, and $\sigma = 9$ is shown in Fig. 2, where T is the number of masks, and t is a given threshold value of sampling. The details of each algorithm are described as follows.

1) Inserting: When inserting $\mathscr{B}$ into $\mathscr{D}$, Inserting first processes $\mathscr{B}$ as the following. First, Inserting encodes each item b_i to a set $\mathscr{S}_i \leftarrow \mathsf{Encode}(b_i)$. Second, Inserting builds an inverted index [41] for each item in $\mathscr{S}$ ($\mathscr{S} = \mathscr{S}_1 \cup \mathscr{S}_2 \ldots \cup \mathscr{S}_n$). We use $\mathscr{M}$ to denote the set of an item b_i's inverted indices. The process of encoding with $|\mathscr{B}| = 3$ and $T = 3$ is shown in Fig. 2 (a). Figure 2 (b) illustrates the process of constructing the inverted index for items in $\mathscr{S}_1$, $\mathscr{S}_2$, and $\mathscr{S}_3$.

Upon the arrival of an item e in $\mathscr{S}$, Inserting finds the bit where e is mapped by computing $H(e)$ ($H(\cdot)$ is a hash function) and three modulus operations (lines 5-6 in Algorithm 1).

Algorithm 1 Inserting Subroutine of PBO-FQ

Require: $\mathbb{B}_{x,y} \leftarrow 0$, an empty list set $\mathbb{L}$, $\mathscr{S} \leftarrow 0$

1: **procedure** INSERTING(e)
2: $\mathscr{S} \leftarrow \mathsf{Encode}(\mathscr{B})$
3: $\mathscr{M} \leftarrow \mathsf{InvertedIndex}(\mathscr{S})$
4: **for** each item e in $\mathscr{S}$ **do**
5: $h \leftarrow H(e)$
6: $i \leftarrow h \bmod x$, $j \leftarrow h \bmod y$, $p \leftarrow h \bmod \sigma$
7: acquire index set Ind_e of e from $\mathscr{M}$
8: $\mathbb{B}_{i,j} \leftarrow \mathbb{B}_{i,j} \vee (1 \ll p)$ ▷ set desired bit to 1
9: append Ind_e to $\mathbb{L}_{i,j,p}$
10: **end for**
11: **return** $(\mathbb{B}_{x,y}, \mathbb{L})$
12: **end procedure**

Algorithm 2 Perturbing Subroutine of PBO-FQ

Require: $\mathbb{B}'_{x,y} \leftarrow 0$, an empty list set $\mathbb{L}'$, probability value $\pi \in (0, 1)$

1: **procedure** PERTURBING($\mathbb{B}_{x,y}$, $\mathbb{L}$)
2: Generate the Laplace distribution noise data $\mathscr{L} = \{l_1, l_2, \ldots, l_n\}$ where $n = x * y * \sigma$
3: Let $\mathscr{F}(\delta) = \pi$ to compute the threshold value δ, where $\mathscr{F}$ is the cumulative distribution function of $\mathscr{L}$
4: **for** $m = 1$ to n **do**
5: **if** $l_m \geq \delta$ or $l_m \leq -\delta$ **then**
6: $i = m/(y * \sigma)$ ▷ / denotes quotient operation
7: $j = (m/\sigma) \mod y$
8: $p = m \mod \sigma$ ▷ map l_m to a certain bit
9: $\mathbb{B}'_{i,j} \leftarrow \mathbb{B}'_{i,j} \vee (1 \ll p)$
10: append random indices to $\mathbb{L}'_{i,j,p}$
11: **end if**
12: **end for**
13: $\mathbb{B}_{x,y} \leftarrow \mathbb{B}_{x,y} \oplus \mathbb{B}'_{x,y}$, $\mathbb{L} \leftarrow \mathbb{L} \cup \mathbb{L}'$
14: **return** $(\mathbb{B}_{x,y}, \mathbb{L})$
15: **end procedure**

Algorithm 3 Lookup Subroutine of PBO-FQ

Require: An empty set $\mathbb{L}_w$

1: **procedure** LOOKUP(w)
2: $\mathscr{S}_w \leftarrow \mathsf{Encode}(w)$
3: **if** $\mathbb{B}_{x,y} == 0$ **then**
4: **return** lookup failure
5: **end if**
6: **for** l=1 to T **do**
7: $h_l \leftarrow H(e_l)$
8: $i_l \leftarrow h_l \bmod x$, $j_l \leftarrow h_l \bmod y$, $p_l \leftarrow h_l \bmod \sigma$
9: $Flag_l \leftarrow (\mathbb{B}_{i_l,j_l} \wedge (1 \ll p_l)) \gg p_l$
10: **if** $Flag_l == 1$ **then**
11: $\mathbb{L}_w = \mathbb{L}_w \cup \mathbb{L}_{i_l,j_l,p_l}$
12: **end if**
13: **end for**
14: $Num, Ind = \mathsf{Max.Count}(\mathbb{L}_w)$
15: **return** Num and Ind
16: **end procedure**

After that, Inserting sets the bit to 1 and records the inverted indices of e in a linked list corresponding to the bit. The process of inserting an item into PBO-FQ is illustrated in Inserting (Algorithm 1). Figure 2 (c) gives an illustrative example on the inserting operation (Inserting(e_2)), where $H(e_2) = 19$.

2) Perturbing: The process of perturbing is illustrated in Algorithm 2. To protect the privacy of the original hash in $\mathscr{B}$, a DP perturbation method [14] is introduced to PBO-FQ. The technical pivot of [14] is to convert Laplace noise in floating points to 0 and 1 based on a preset probability value/noise budget π. Specifically, it first generates the Laplace distribution noise data $\mathscr{L}$ with location parameter $\mu = 0$ and scaling parameter $\lambda = 1$, where $|\mathscr{L}| = x * y * \sigma$. Given $\mathscr{L}$, we can obtain its cumulative distribution function $\mathscr{F}$ by computing:

$$\mathscr{F}(x) = \int_{-\infty}^{x} f(u)d(u) = \begin{cases} \frac{1}{2}\exp(x), x < 0 \\ 1 - \frac{1}{2}\exp(-x), x \geq 0 \end{cases}. \quad (3)$$

Based on π and $\mathscr{F}$, we then can select a positive threshold value δ, e.g., suppose $\pi = 0.2$, let $\mathscr{F}(x) = 1 - \pi/2$, then we have $\delta = x = 1.61$. If $l \geq \delta$ or $l \leq -\delta$ ($l \in \mathscr{L}$), the Laplace noise l will be converted to 1. Otherwise, the noise will be converted to 0. However, since such generated noise data is one-dimensional, it cannot be directly mapped to a two-dimensional filter. We then organize the one-dimensional noise data into two-dimensional according to the dimensions of $\mathbb{B}'_{x,y}$ (lines 4-9 in Algorithm 2). Meanwhile, the random indices will be appended to the linked list corresponding to the bit renovated to 1. Finally, Perturbing computes $\mathbb{B}_{x,y} = \mathbb{B}_{x,y} \oplus \mathbb{B}'_{x,y}$ and $\mathbb{L} = \mathbb{L} \cup \mathbb{L}'$ to output the final filter. A simple example of PBO-FQ after Inserting and Perturbing with $x = 3$, $y = 5$, and $\sigma = 9$ is shown in Fig. 2 (c).

Note that other mechanisms, e.g., random response and Gaussian noise, can also be applied to PBO-FQ for privacy protection purpose. Since we focus on constructing a filter that satisfies differential privacy, we only present an instantiation of PBO-FQ based on the Laplace method and prove the privacy property accordingly. Exploring which mechanism is more efficient is not within the scope of this study. We would like to point out that the bit-flipping operation for perturbing Bloom filter in PBO-FQ is also a form of random response.

3) Lookup: The lookup operation is illustrated in Algorithm 3, which fails when $\mathbb{B}_{x,y}$ is empty. When looking up a new item w, Lookup first encodes w to a set $\mathscr{S}_w$ by invoking Encode($\cdot$). Each item in $\mathscr{S}_w$ can be determined whether it has been inserted into the filter by computing $Flag$ for them (lines 6-9 in Algorithm 3). If $Flag_l$ equals 1, Lookup then records the e_l's indices stored in linked list into $\mathbb{L}_w$ by union operation. Finally, Lookup counts the number of occurrences of each index in the set $\mathbb{L}_w$ and returns the max count Num and the corresponding index Ind.[4] If Num is greater than or equal to t, meaning that w is similar to an item b_{Ind}. Otherwise, w is not similar to any item in $\mathscr{B}$. Fig. 2 illustrates the process of Lookup, where the image's hash is 101011. After performing

[4]We use Max.Count($\cdot$) (line 14 in Algorithm 3) to denote this operation.

Lookup, we can obtain $Num = 2$ and $Ind = 1$. Since $Num \geq t$, the image will be identified as similar to b_1.

C. Analysis on PBO-FQ

1) Differential Privacy: The designed filter PBO-FQ can provide ϵ-DP assurance for original data $\mathscr{B}$. The privacy of $\mathscr{B}$ is preserved by a two-step process in our design. First, each item in $\mathscr{B}$ is hash-mapped to compose the filter, which avoids exposing plaintext hash directly to the client. Although the collision of different elements mapped to the same bit position can provide privacy protection for $\mathscr{B}$, it is limited and vulnerable to membership attacks and differential attacks. For this, the second step is adding noise generated from Laplace distribution data to the output of PBO-FQ.

Assume $\mathscr{D}$ is the original filter without perturbing, the probability of $\mathscr{D}$ outputting 1 is f, and $1-f$ is the probability of $\mathscr{D}$ outputting 0. To quantify the noise impact of π on the output of the filter, we introduced the following theorem.

Theorem 1: The constructed PBO-FQ satisfies ϵ-differential privacy where $\epsilon = \ln \frac{f}{(1-f)\pi}$.

Proof: Let the output of $\mathscr{D}$ be $d = \{d_1, d_2, \ldots, d_i, \ldots, d_n\}$ (n is the length of $\mathscr{D}$), $\mathscr{D}^*$ be the perturbed $\mathscr{D}$ with Laplacian noise added, and $d^* = \{d_1^*, d_2^*, \ldots, d_i^*, \ldots, d_n^*\}$ denotes the output of $\mathscr{D}^*$.

For each element in the filter, the probability of observing $\mathscr{D}^*$ output 1 when $\mathscr{D}$ output 1 is:

$$P(d_i^* = 1 | d_i = 1) = f \cdot (1 - \pi) + f \cdot \pi = f. \tag{4}$$

The probability of observing $\mathscr{D}^*$ output 1 when $\mathscr{D}$ output 0 is:

$$P(d_i^* = 1 | d_i = 0) = (1 - f) \cdot \pi. \tag{5}$$

Based on the above two cases, we can obtain:

$$\exp(\epsilon) = \frac{P(b_i^* = 1 | b_i = 1)}{P(b_i^* = 1 | b_i = 0)} = \frac{f}{(1 - f)\pi}. \tag{6}$$

Then, we have $\epsilon = \ln \frac{f}{(1-f)\pi}$. □

The proof indicates that the perturbation method of adding 0 or 1 in $\mathscr{D}$ based on threshold-Laplace distribution noise satisfies the DP. From Eq. 6, we find that larger π leads to smaller ϵ under a given f, which means that more noise inserted facilitates stronger privacy protection strength. On the other hand, larger f leads to larger ϵ under a given π, which means that excessive numbers of 1 in $\mathscr{D}$ will weaken the effect of perturbation caused by noise data, thus hindering privacy protection.

In order to strike a balance between accuracy and privacy, we adopt a strategy that prioritizes privacy while still maintaining a reasonable level of accuracy to choose the optimal π. Namely, we first establish a bound on the desired accuracy, e.g., in the proposed scheme, we set a requirement to keep the FPR below 0.1%. With this accuracy threshold defined, the optimal π is the maximum value of π that could satisfy these accuracy requirements (larger π, better privacy in Theorem 1).

2) Accuracy: The false positive (FP) and false negative (FN) are the main yardsticks of the accuracy of the BF and its variants, which also are optimization objectives in the solutions proposed to engineer high-accuracy filters. In contrast to them, we trade off accuracy for privacy in the design of our filter.

FP events enable the privacy protection feature, where the higher the false positive rate (FPR), the better the privacy protection effect. In our design, FP is mainly caused by adding noise to the output of the filter. The internal link between privacy protection and noise budget π is explored in Section VI-C.1. The standard 2DBF does not produce FNs, but PBO-FQ possibly produces FNs caused by the sampling operation introduced for the fuzzy query. We continuously optimize the sampled bit, t and T to eliminate FNs to reduce false negative rate (FNR). Section VI details the results on the FNR and FPR caused by sampling operation with different T and t.

V. Design of Entbergen

A. Syntax

The proposed *Entbergen* consists of four algorithms: Setup, NewUser, LocalCheck and Verification. When users share a media through E2EE, the client first checks whether there exists $b \in \mathscr{B}$, such that $d_H(w, b) \leq \delta_H$ depending on the designed filter PBO-FQ. However, perturbation data added into PBO-FQ results in some false positives. Thus, we further design a Verification algorithm to eliminate false positives. With these four algorithms, *Entbergen*'s false positive rate is equal to 0.

The syntax of *Entbergen* is defined as follows:

- Setup$(\mathscr{B}, pp) \rightarrow (\mathscr{D}, pk_s, sk_s)$. The server runs this algorithm at the system setup. With the input of public parameters pp and harmful media dataset $\mathscr{B}$, it first constructs the PBO-FQ $\mathscr{D}$ by invoking Inserting and Perturbing. It then generates a server key pair (pk_s, sk_s).
- NewClient$(pp) \rightarrow (\mathscr{D}, pk_c, sk_c)$. NewClient is a protocol between a new client and the server to register that client in the system. On success, the server should offload $\mathscr{D}$ to the client, and the client generates a client key pair (pk_c, sk_c).
- LocalCheck$(\mathscr{D}, w) \rightarrow (Num, Ind)$. The client runs this algorithm as the user shares media. With the input of $\mathscr{D}$ and the shared media's hash w, the client checks whether the w is similar to the item in $\mathscr{B}$ by invoking Lookup. Upon successfully completing the algorithm, the client gets Num and Ind, with Num bigger than the pre-defined threshold, asserting that w is harmful media. If the media is identified as suspicious, the client would invoke Verification algorithm. Otherwise, the media would be transferred to the recipient.
- Verification$(Ind, w, b_{Ind}) \rightarrow d_H(w, b_{Ind})$. Verification is an interactive algorithm between the client and the server and would be awakened when the w is identified as suspicious by LocalCheck. This algorithm eliminates the FP by privately computing the Hamming distance $d_H(w, b_{Ind})$ between w and its similar item. The server can learn $d_H(w, b_{Ind})$ in the result-revealed scheme, but only the client can learn $d_H(w, b_{Ind})$ in the result-unrevealed scheme.

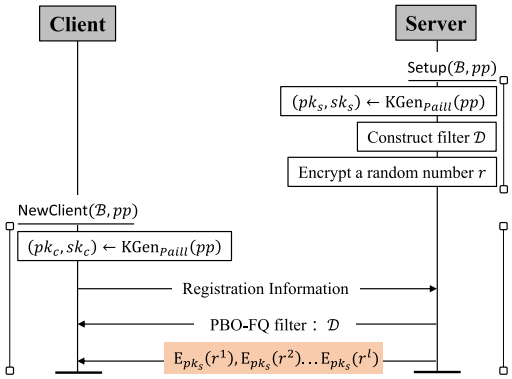


Fig. 3. The offline pre-processing phase of *Entbergen*.

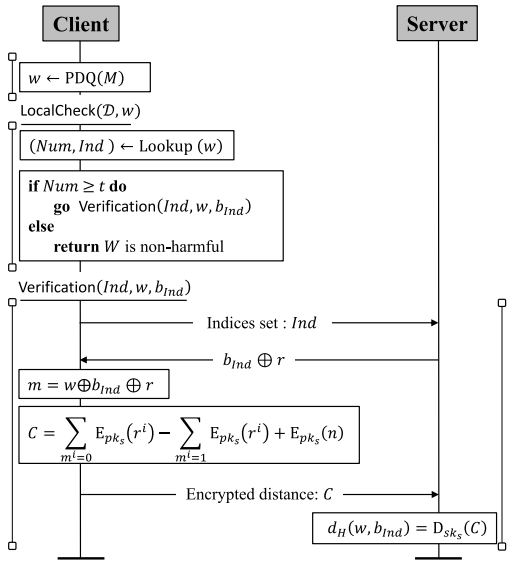


Fig. 4. The online identification phase of the result-revealed scheme.

We emphasize that the differential property of *Entbergen* is as: the harmless media dominating social networks can be filtered by LocalCheck at little cost (only perform the hash operation and need 0 communication overhead), while the remaining suspicious media requires invoking a slightly complex algorithm, i.e., Verification, to be further inspected.

In the Verification phase, a major tool that we use is an additively homomorphic cryptosystem. Given two ciphertexts $\mathsf{E}(m_1)$ and $\mathsf{E}(m_2)$, this tool enables to compute $\mathsf{E}(m_1 + m_2)$, or compute $\mathsf{E}(c \cdot m_1)$ for any known constant c, without the knowledge of the private key. By sending an encrypted random value to the client in advance for further optimization, we can free the client and server from executing the computationally expensive encryption algorithm. W.l.o.g., we use the Paillier algorithm [17] to instantiate the proposed schemes. Note that, although the GM algorithm [42] can be directly applied to privately compute the Hamming distance using XOR homomorphic property, its encryption operation must be performed by the client and server, resulting in additional latency.

B. Instantiation of Result-Revealed Scheme

The result-revealed scheme consists of the offline pre-processing and online identification phases, described in Fig. 3 and Fig. 4. In the offline pre-processing phase, Setup and NewClient are conducted by the server and the client, respectively. Before establishing the system, the server constructs the PBO-FQ $\mathscr{D}$. Upon successful user registration, the server offloads $\mathscr{D}$ to the client. The server and the client generate their own key pair (pk_s, sk_s) and (pk_c, sk_c) using the Paillier cryptosystem [17], respectively.

In the online identification phase, LocalCheck is first conducted on the client side. When a user shares media M, the client computes its perceptual hash $w = PDQ(W)$ utilizing PDQHash [23]. Note that $\mathscr{B}$ is constructed by the same perceptual hash function, and w and $b \in \mathscr{B}$ have the same length l. To check whether w is similar to an item in the known harmful hash set $\mathscr{B}$, the client invokes Lookup and then obtains Num and Ind. If Num is smaller than t, M is identified as harmless and can be transmitted to the recipient through E2EE. If Num is greater than or equal to t, M is identified as suspicious. As the perturbation is added into $\mathscr{D}$, the client cannot distinguish whether it is truly harmful or harmless media falsely reported as harmful media. To this end, $d_H(w, b_{Ind})$ is computed through Verification ran between the client and the server.

In the context of plaintext, $d_H(w, b_{Ind})$ can be obtained by bit-addition, i.e., $d_H(w, b_{Ind}) = w^i(1 - b^i_{Ind}) + (1 - w^i)b^i_{Ind}$, where w^i and b^i_{Ind} represent the i-th bit of the binary hash string w and b_{Ind}, respectively. A naive method could have the server calculate $\mathsf{E}_{pk_s}(b^i_{Ind})$ ($\mathsf{E}_{pk}(\cdot)$ is an encryption algorithm under pk) for each bit location i utilizing the Paillier cryptosystem [17]. After receiving the encrypted representation $\{\mathsf{E}_{pk_s}(b^1_{Ind}), \mathsf{E}_{pk_s}(b^2_{Ind}) \ldots \mathsf{E}_{pk_s}(b^l_{Ind})\}$, the client can compute $C = \sum_{i=1}^{l} \mathsf{E}_{pk_s}(b^i_{Ind}) \cdot_{pk_s} (1 -_{pk_s} w^i) +_{pk_s} (1 -_{pk_s} \mathsf{E}_{pk_s}(b^i_{Ind})) \cdot_{pk_s} w^i$. Finally, the server can get the Hamming distance $d_H(w, b_{Ind}) = \mathsf{D}_{sk_s}(C)$ ($\mathsf{D}_{sk}(\cdot)$ is a decryption algorithm under sk). However, in our experiments, we found that the encryption algorithm of the Paillier cryptosystem is computationally expensive and leads to high latency.

To reduce the online overhead, we apply the optimization means proposed in SCiFI [43]. First, the server chooses a random l bit binary string r and sends the encryption of these bits to the client upon successfully registering a user in the offline pre-processing phase. In the online identification phase, when the server receives Ind, it sends $r \oplus b_{Ind}$ to the client. The client then computes $m = r \oplus b_{Ind} \oplus w$. Second, the computation of the Hamming distance of w and b_{Ind} is converted to homomorphic addition or subtraction on $\mathsf{E}_{pk_s}(r^1), \ldots, \mathsf{E}_{pk_s}(r^l)$. For each bit location i, the client adds $\mathsf{E}_{pk_s}(r^i)$ to the sum, if $m^i = 0$, or adds $1 - \mathsf{E}_{pk_s}(r^i)$ if $m^i = 1$. Suppose n denotes the number of bits in m which are equal to 1, the encryption of the Hamming distance can be computed as $C = \sum_{m^i=0} \mathsf{E}_{pk_s}(r^i) - \sum_{m^i=1} \mathsf{E}_{pk_s}(r^i) + \mathsf{E}_{pk_s}(n)$.

The server then can compute $\mathsf{D}_{sk_s}(C)$ to obtain the Hamming distance $d_H(w, b_{Ind})$. If $d_H(w, b) \leq \delta_H$, the server would warn users against abusive content. In extreme cases, where users share criminal media, such as CSAM and drug promotional materials, the server could report the information to the law enforcement agency to combat crimes.

In the result-revealed scheme, b_{Ind} is hidden in $r \oplus b_{Ind}$. Without secret key sk_s, the client cannot decrypt $\mathsf{E}_{pk_s}(r)$ and thus cannot learn any information about b_{Ind}. The Paillier algorithm's semantic security ensures that $\mathsf{E}_{pk_s}(r)$ reveals no

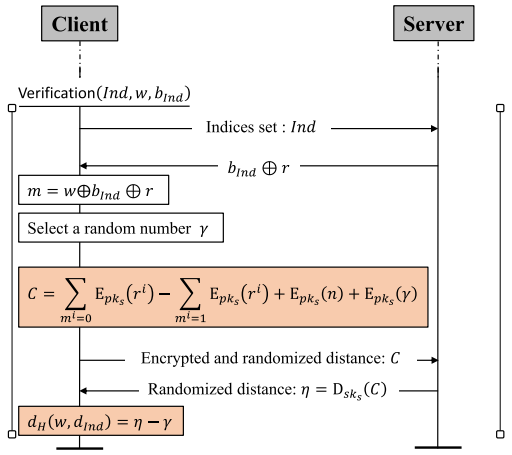


Fig. 5. The online identification phase of the result-unrevealed scheme.

information about r. Therefore, the result-revealed scheme reveals no information about $\mathscr{B}$ to a client. Besides, the server only obtains the Hamming distance between w and b_{Ind} after participating in Verification. As analyzed in Section III-B, it is difficult to speculate on w-related information by virtue of Hamming distance alone, especially when w and b_{Ind} are not similar. Therefore, in the result-revealed scheme, the server cannot recover w and learn nothing about the shared media if the media is harmless.

C. Instantiation of Result-Unrevealed Scheme

Although the result-revealed scheme protects content privacy, it provides a backdoor to the server to implement illegitimate surveillance by appending a chosen watch list to $\mathscr{B}$. Hence, a more strict form of user privacy is considered in this section, where the result is confidential for any entity other than the client.

Intuitively, we could achieve result-unrevealed in the scheme described in Section V-B, if there is a way to randomize C to hide the Hamming distance from the server. We achieve a more strict form of user privacy with minor modifications to the result-revealed scheme.

Our result-unrevealed construction preserves the basic framework of the result-revealed scheme in that the server constructs the private filter and offloads it to the client in the offline pre-processing phase. The client first operates LocalCheck when the user shares media in the online identification phase. Modifications occur in the last two steps of the protocol, as shown in Fig 5. Before the client computes the encryption of the Hamming distance C, it selects a secret random number γ as the randomization factor. Subsequently, the client computes $C = \sum_{m^i=0} \mathsf{E}_{pk_s}(r^i) - \sum_{m^i=1} \mathsf{E}_{pk_s}(r^i) + \mathsf{E}_{pk_s}(n) + \mathsf{E}_{pk_s}(\gamma)$ and sends it to the server. With this, the Hamming distance is hidden in the randomized value $\gamma + d_H(w, b_{Ind})$.

Since only the client knows γ, the actual Hamming distance of w and b_{Ind} can be merely decrypted by the client. If $d_H(w, b) \leq \delta_H$, the client will notify and warn the user at its side. In contrast to the result-revealed scheme, the result-unrevealed scheme targets client-side notification, not platform notification.

In the result-unrevealed scheme, the server only obtains $\gamma + d_H(w, b_{Ind})$. Without knowledge of the client randomization factor γ, the server cannot undo the offset to learn $d_H(w, b_{Ind})$. The security of the Paillier cryptosystem ensures that the server cannot learn γ. Therefore, the server learns nothing about $d_H(w, b_{Ind})$ and w in the result-unrevealed scheme.

In the result-unrevealed scheme, the client obtains $d_H(w, b_{Ind})$ and knows the match result. Note that revealing the Hamming distance to the client admits to an attack on server privacy. In the case where $d_H(w, b_{Ind}) = 0$, a malicious client can recover r by computing $w \oplus r \oplus b_{Ind}$. They can then iteratively learn the rest of the harmful hash set. To combat this issue, an update policy of r is introduced in the result-unrevealed scheme. In general, the server sends a new encrypted r to the client regularly to update it. For the users whose shared media has been identified as suspicious frequently in a short time, the server sends new r to these users to update it in real-time.

The semantic security of the Paillier algorithm has been rigorously proved in previous works [17], [43]. In the proposed schemes, the unmodified Paillier algorithm is employed, thereby maintaining its inherent security. Please kindly refer to these works for a detailed description of its security proof.

VI. Evaluation

A. Experimental Setup

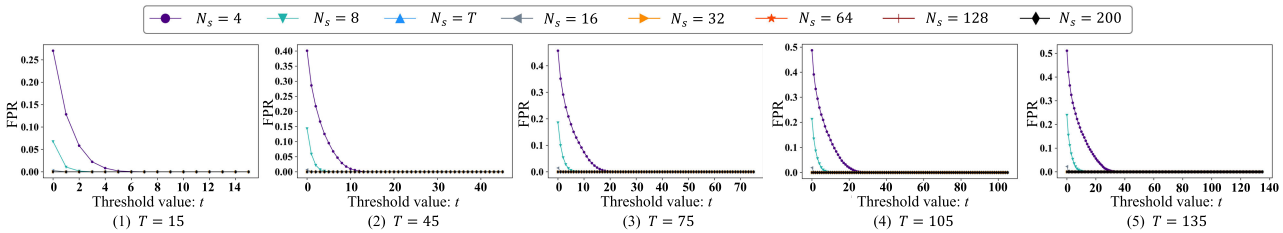
1) Data Collection: An ideal database $\mathscr{B}$ should contain various types of harmful materials. However, since potential ethical and privacy concerns, we cannot obtain them. Hence, just like other schemes of detecting harmful media in E2EE communications [12], [13], we aggregate images from the public datasets and encode each image with PDQHash to form varying sizes of $\mathscr{B}$. Specifically, the public databases consist of common objects [44], [45], faces [46], and social media [47]. For shared images at the client side, half of them are collected from the Flickr30k [48] to simulate the shared images that are not similar to any image in $\mathscr{B}$, and the rest are generated by modifying $\mathscr{B}$ to simulate a match between shared images and harmful dataset. It is noteworthy that lacking harmful content will not affect the evaluation of the scheme's performance.

2) Implementation: We use a workstation equipped with an Intel CPU@i9-10900K and 64 GB RAM running on Linux for server-side computation. For the client side, we use a lightweight mobile device, Xiaomi 10, with the Android operating system v10 to execute the proposed protocols.

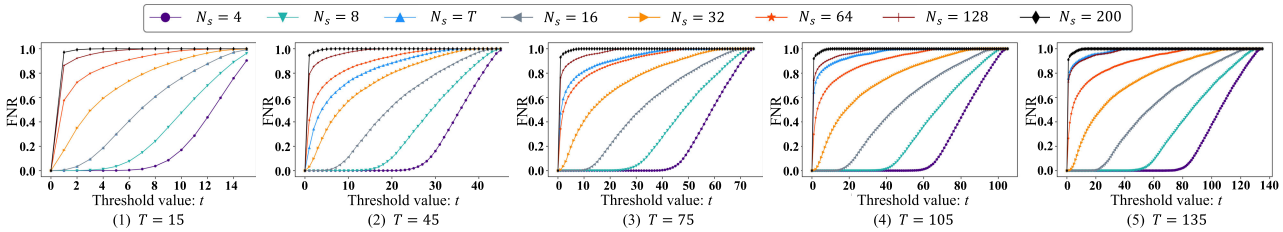
B. Performance of Encoding

Recall that at the beginning of building the filter, each item in $\mathscr{B}$ is encoded to a sampled set to achieve a fuzzy query. In this section, we mainly present the experimental results on how to construct a set of efficient masks and the sensitivity of encoding.

1) Masks Construction: The performance of masks is closely related to the number of masks, threshold value, and the number of sampled bits (T, t, N_s). A set of efficient masks should preserve both small FNR and FPR with a given threshold value t, and t should also be as small as possible.



(a) FPR of encoding operation with different parameter combinations.



(b) FNR of encoding operation with different parameter combinations.

Fig. 6. Performance of encoding operation with different parameter combinations. t is the threshold value, T is the number of masks, and N_s is the number of sampled bits.

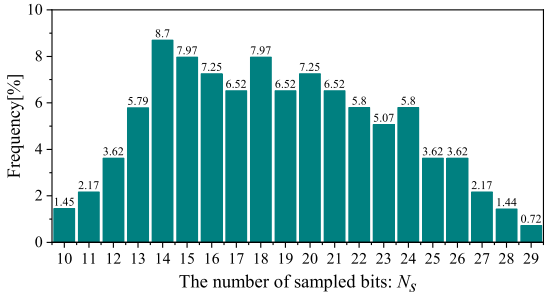


Fig. 7. Stats of the number of sampled bits N_s in masks.

Tuning all parameters together has its own challenges because this is a big search space to explore. In this section, we use the grid-search method to explore which parameter combination preserves better performance. Specifically, we first give a range $[10, 150]$ for T, and then select T from the range with step 10. For each T, we let t vary from 1 to T and N_s vary from 1 to 255 (the hash string's length is 256-bit). Based on the above parameter combinations, 286,720 sets of masks are constructed. We then evaluate these masks' performances by computing their FNRs and FPRs on the same dataset. Fig. 6 shows a portion of the experimental results. Some observations about our results are as follows:

a) Better performance when N_s accounts for about 5%-10% of the hash string's length: As indicated in Fig. 6, masks with a lot of sampled bits possess lower FPR, but perform poorly on FNR. A small number of sampled bits lead to an opposite performance. To explore the efficacious number of sampled bits, we count the convergence rates of the constructed masks' FNR and FPR. Specifically, we set a small range for threshold value, i.e., $t \in [1, 5]$, and count what N_s can make FNR and FPR become 0 simultaneously within this interval. The statistics are reported in Fig. 7. As shown, 90% of masks' sampled bits are 13-26, which accounts for 5%-10% of the total length of 256.

b) Larger T, better fault tolerance: On one hand, larger T contains more information on the raw data, thus perceiving more similar and dissimilar data. On the other hand, larger T means more choices for t (note that $t \in [1, T]$). As shown in Fig. 6, when $T = 15$, we cannot tune t and N_s to achieve both $FPR = 0$ and $FNR = 0$. A large T can provide more diverse trade-offs between FNR and FPR. In addition, empirical results show that for datasets with sizes of 10 thousand, 100 thousand, and 1 million, the encoding has better performance when T is greater than 25, 35, and 55, respectively.

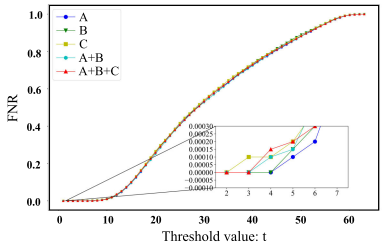
To save space, we only visualize a portion of the experimental results in this part. The whole empirical data is publicly available at Aliyundrive[5] in the hope of providing valuable insights on constructing masks with users of *Entbergen*.

2) Sensitivity of Encoding on Different Datasets: Based on the above insights, we can construct a set of masks that satisfy the desired requirements for a given dataset. In this section, we extend our investigation to examine the sensitivity/generalization of the encoding. Specifically, we explore whether the performance remains satisfactory when applying the encoding operation to different datasets using the same pre-constructed masks.

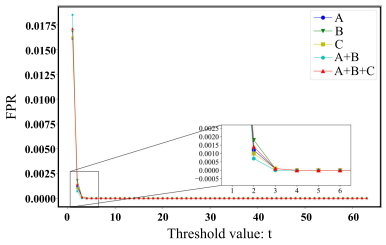
We initially construct a set of masks on benchmark dataset A, which achieves $FNR = 0$ and $FPR = 0$ under the parameters (64, 4, 16). Subsequently, we apply the encoding operation using the same pre-constructed masks on four additional testing datasets to assess the sensitivity of the constructed masks. The testing datasets consist of two distinct datasets B and C, as well as two mixed datasets A+B and A+B+C (where "+" denotes incremental update). The datasets of A, B, and C are formed by randomly sampling 100,000 images from the COCO dataset. The FNRs and FPRs of encoding on these datasets are depicted in Fig. 8.

It is observed that the FNRs on the testing datasets when the threshold value $t = 4$ are slightly higher than that on the benchmark dataset (about 0%-0.015%), but they exhibit consistent behavior. Besides, the FPRs on the testing and

[5]https://www.aliyundrive.com/s/swjv6qQM7g5



(a) FNR of encoding on different datasets.



(b) FPR of encoding on different datasets.

Fig. 8. Sensitivity testing of the same set of masks on different datasets. "+" denotes the incremental update.

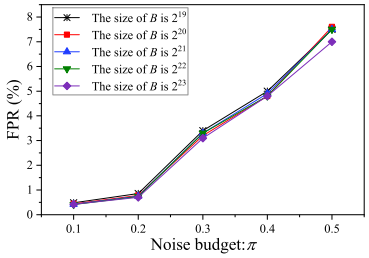


Fig. 9. FPR with noise budget π varying from 0.1 to 0.5, and different size of $\mathscr{B}$.

benchmark datasets are equal to 0 when the threshold value $t = 4$.

Based on the above results, we can conclude that the encoding operation is robust to loose conditions, i.e., the margin of error caused by encoding using the same masks on different datasets is tolerable. However, the performance of encoding using the same masks on different datasets may not always satisfy strict hypothetical conditions (e.g., keeping $FNR = 0$). For the latter condition, one can try to decrease the threshold value t or reconstruct new masks.

C. Performance of the PBO-FQ

1) Accuracy and Privacy Assessment: This section investigates the accuracy and privacy of PBO-FQ. The accuracy of PBO-FQ is measured by FNR and FPR. The FNR of PBO-FQ is only affected by encoding, namely, $FNR_{PBO-FQ} = FNR_{sampling}$, which is evaluated in Fig. 6. The FPR of PBO-FQ is mainly affected by noise added in. Hence, we explore the impact of different noise budgets on the FPR of PBO-FQ. To evaluate the FPR,[6] we first establish original filters for varying set sizes $\mathscr{B}$ with the same initial FPR = 0.00001 and $\pi = 0$. Subsequently, PBO-FQs with noise budget π ranging from 0.1 to 0.5 are constructed. Fig. 9 demonstrates

[6]In this part, FPR refers specifically to the FPR of PBO-FQ.

TABLE I

MEMORY CONSUMPTION OF PBO-FQ ($\mathbb{B}_{x,y}$, $\mathbb{L}$) WITH $T = 32$ FOR VARYING SET SIZES $|\mathscr{B}|$ AND NOISE BUDGET π

$\lvert\mathcal{B}\rvert$	Memory consumption (Mb)				
	$\pi = 0.1$	$\pi = 0.2$	$\pi = 0.3$	$\pi = 0.4$	$\pi = 0.5$
2^{19}	41.1	43.1	45.2	47.3	49.4
2^{20}	86.1	90.3	94.4	98.6	102.7
2^{21}	174.3	183.6	191.8	199.1	207.4
2^{22}	376.5	391.1	407.7	424.2	440.8
2^{23}	781.2	814.2	847.4	880.5	909.6

that the FPR increases when π rise. As for privacy, we follow the prior works [14], FPR is used as a metric for the privacy protection capability of PBO-FQ. As shown in Fig 9, a larger π leads to a higher FPR, which increases privacy. The designed PBO-FQ provides a method for the moderation party to protect the privacy of the harmful hash set by flexibly tuning the noise budget.

2) Memory Assessment: The proposed filter PBO-FQ consists of a 2-dimensional array $\mathbb{B}_{x,y}$ and a linked list $\mathbb{L}$, which occupies the most memory. Table I shows the memory consumption of $\mathbb{B}_{x,y}$ and $\mathbb{L}$ for varying set sizes $|\mathscr{B}|$ and noise budget π. The memory consumption will increase significantly with the rise of the set sizes $|\mathscr{B}|$ as shown in Table I. Although the memory size of $\mathbb{B}_{x,y}$ remains constant with the increase of π, the total memory size of PBO-FQ would increase as the random indices are appended to the linked list. We investigated the mobile phones sold top 100 on Amazon and JD.com. Among them, the minimum memory provided is 64 GB. Hence, the memory consumption of the proposed schemes is acceptable for current mobile phones.

D. Performance of Entbergen

This section presents the performance results of *Entbergen* and compares them with Kulshrestha and Mayer's scheme [12]. Given the differences between *Entbergen* and the schemes of Hua et al. [13] and Apple [29] in security definitions and functions, it is biased to directly compare with these schemes' performances.

1) Accuracy of Entbergen: The deterministic inspection ability of Verification algorithm (i.e., using homomorphic encryption) in *Entbergen* could eliminate all the FP raised in local check of PBO-FQ. However, there is still an accuracy loss due to FN in *Entbergen*. The FNR of *Entbergen* is equivalent to that of the plaintext algorithm, because FN is totally caused by the utilized perceptual hash functions.

This part focuses on evaluating the FNR of the PDQHash employed in *Entbergen* to evaluate its robustness against image transformations. We conduct the evaluation by compiling an image dataset, then examining the perceptual hash Hamming distance between each original image and its similar image. We randomly sample 10,000 images from the COCO 2017 unlabeled image dataset [44] and apply 3 transformations to each image. The transformations reflect possible quality loss (quality compression) and noise impacting (Gaussian noise addition) in transmission, and possible user action to make images visually appealing (gamma correction). We test six extents for each transformation. Specifically, 1) the extents of quality compression are sampled as $c = \{1, 10, 20, 30, 50, 80\}$ from [0%, 100%], where smaller c indicates more severe

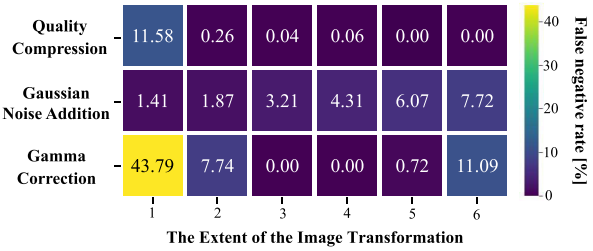


Fig. 10. Heatmap of FNR of PDQHash under different transformations. The color bar on the right shows how the colors and FNR values are mapped, and the numbers on the bottom correspond to 6 different extents of transformation (the transformation extent is not proportional to the number).

TABLE II
OFFLINE CPU EXECUTION TIME OF CONSTRUCTING MASKS FOR VARYING SET SIZES $|\mathscr{B}|$

T	CPU execution time (h)				
	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}
65	0.35	0.60	0.69	1.62	4.38
75	0.41	0.91	1.71	3.93	9.04
85	0.47	1.03	2.14	4.49	10.32

quality loss, 2) the extents of Gaussian noise addition are sampled as $\theta = \{0.01, 0.02, 0.04, 0.06, 0.08, 0.10\}$, where larger θ indicates adding more noise, 3) the extents of gamma correction are sampled as $\gamma = \{0.3, 0.5, 0.8, 1.3, 1.5, 1.8\}$ from $[0, 2]$, where γ less than 1 can enhance the brightness of the dark parts of the image and γ greater than 1 can improve the contrast.

Fig. 10 illustrates the impact of different types of transformations on the FNR of PDQHash when dealing with perceptually similar images. The results demonstrate that PDQHash could exhibit robustness when facing moderate levels of transformations. However, when subjected to more substantial degrees of certain transformations (such as quality compression and gamma correction), PDQHash's ability to accurately detect similar images degrades. Similar to previous schemes [12], [13] that rely on perceptual hash functions, the proposed scheme is unavoidably susceptible to false negatives.

2) Offline Masks Construction Cost: In this part, we evaluate the cost of constructing a set of expected masks offline. We set the number of sampled bits N_s in range 15-20, and the termination condition for the mask construction procedure is $FNR = 0$ AND $FNR \leq 0.005\%$ at the same threshold value t. Table II presents the CPU execution time of constructing several sets of masks with different T (number of masks) for varying harmful hash set sizes. As shown, for different T and different data sizes, the server needs 0.35-10.32 hours to construct the desired masks. It is worth-noting that the mask construction procedure occurs offline and does not affect the efficiency of online identification of harmful images. In practice, hash set is not frequently updated, given the period for collecting and determining harmful media.

3) Online Identification Cost: Before presenting the computation and communication costs of the online identification phase, we specify the following parameters: $l = 256$, $T = 64$, $t = 2$, $N_s = 16$ and $\pi = 0.2$, which are used to construct the filter. These values were selected to balance memory consumption and privacy, in line with studies by [14] and [15]. We present benchmark results measured on a lightweight mobile client in Table III.

a) Online computation cost: We present the average total execution time for varying $|\mathscr{B}|$ as shown in Table III. In the proposed schemes, the execution time varies much between shared images that match an item in $\mathscr{B}$ and those that do not, due to *Entbergen*'s capability of inspecting differentially. The execution times of the result-revealed scheme and the result-unrevealed scheme are not much different. In both result-revealed and unrevealed schemes, when the user shares harmless media (do not match with any item in $\mathscr{B}$), the client returns the check results in real-time, under 1 second. Even in the user share a suspicious media scenario (match with items in $\mathscr{B}$), the proposed schemes also achieve promising performance (3.04 seconds for large-scale database $|\mathscr{B}| = 2^{23}$).

b) Online communication cost: As shown in Table III, the online communication cost of the proposed schemes varies across different types of shared media. Especially when the shared images are harmless, the communication cost is **0** benefiting from the client's local-check capability. When the shared images are suspicious, the proposed schemes achieve at most 4.33 KB communication cost, which indicates that our protocols can still operate effectively in poor network communication environments.

4) Comparison With Prior Art: We compare the performance of *Entbergen* with the state-of-the-art work [12], which has a consistent security model as this work. Kulshrestha et al. propose to initially map similar items in the harmful hash set to the same bucket using locality-sensitive hashing (LSH) techniques, and then use private information retrieval to assist the client in privately retrieving a set of encrypted buckets that contain hashes similar to the shared image from the server. After that, the client could perform matching tests on ciphertexts and return the result to the server for decryption. The computation times and communication costs are reported in Table III. As shown, *Entbergen* provides an average of 36x, compared with that of [12], speedup in execution time in the scenario where the shared media do not match an item in $\mathscr{B}$. In another scenario where the shared media match an item in $\mathscr{B}$, *Entbergen* also can provide a 9x-15x speedup. For communication cost, compared with [12], *Entbergen* is observed to reduce communication costs by 176–243 times when the shared media is identified as suspicious.

The results show that *Entbergen* could drastically improve the performance of automatically inspecting harmful media in E2EE communications regarding execution time and total bandwidth. Due to local checking efficiency, the harmless media, with its amount far more numerous than that of the harmful media, can be identified in real-time without interaction with the server in *Entbergen*. This is this work's prominent advantage compared with existing methods. Hence, we believe that *Entbergen* is suitable for deployment in real-world social networks for harmful media identification in E2EE communications.

VII. LIMITATIONS AND FUTURE WORK

In light of the threats of harmful content, many governments and international institutions are calling on companies to

TABLE III
COMPARISON OF AVERAGE TIME AND BANDWIDTH BETWEEN *Entbergen* AND PRIOR METHODS. $\mathcal{HLM}$ AND $\mathcal{SSM}$ DENOTE HARMLESS MEDIA AND SUSPICIOUS MEDIA IDENTIFIED BY PBO-FQ, RESPECTIVELY

$\lvert\mathcal{B}\rvert$			2^{20}	2^{21}	2^{22}	2^{23}
Computation Time (s)	USENIX Security21 [12]	$\mathcal{SSM}$	27.5	27.5	27.7	28.3
		$\mathcal{HLM}$	27.5	27.5	27.7	28.3
	Result-Revealed Scheme	$\mathcal{SSM}$	1.80	1.80	2.25	2.72
		$\mathcal{HLM}$	0.76	0.76	0.79	0.81
	Result-Unrevealed Scheme	$\mathcal{SSM}$	1.96	1.96	2.49	3.04
		$\mathcal{HLM}$	0.76	0.76	0.79	0.81
Communication Cost (KB)	USENIX Security21 [12]	$\mathcal{SSM}$	508.07	586.06	742.03	1053.98
		$\mathcal{HLM}$	508.07	586.06	742.03	1053.98
	Result-Revealed Scheme	$\mathcal{SSM}$	2.84	2.84	3.41	4.26
		$\mathcal{HLM}$	0	0	0	0
	Result-Unrevealed Scheme	$\mathcal{SSM}$	2.88	2.88	3.49	4.33
		$\mathcal{HLM}$	0	0	0	0

reserve a 'backdoor' for government and law enforcement departments to access encrypted content for public safety. However, well-known cybersecurity researchers argue that people would then have little chance to resist the expansion of the regulation system or prevent its abuse [49]. **We challenge the assertion of either this or that.** We believe that approaches reconciling public safety and privacy exist, although it is unlikely to have a perfect solution soon. As an attempt in this line, our solution may suffer from the following limitations that require further investigation.

First, PBO-FQ lacks theoretical boundary guarantees on FPR and FNR. Since multiple factors (e.g., sampling parameters and adding perturbation data) will impact FPR and FNR, it is hard to deduce the relations theoretically. In other words, the current version of *Entbergen* lacks a theoretical guarantee on media inspection. We only manage to give some empirical suggestions on setting the filter parameters. Although not robust enough, we believe that leaving the setting to the service provider somehow facilitates tunable performance in their own checking contexts. *As a general guideline evaluated in Section VI-C, one can use small t to make the filter sensitive to harmful ingredients (reducing FNR) while can use light noise injection to reduce FPR effectively*.

Second, updating the harmful hash set would incur repeated filter downloading. The harmful hash set will dynamically add new hashes for different harmful media, making the local filter inconsistent with the global set. Pushing each update to every client would impose high communication costs. In practice, the update and downloading shall be performed periodically for minor maintenance and communication costs. In addition, we also plan to investigate the incremental update of the PBO-FQ filter to provide benign dynamic properties.

Third, although the memory footprint of PBO-FQ is affordable (40 MB-900 MB in our evaluation) for current mobile devices, qualitative user acceptance for such overhead remains to be studied. Unlike other probabilistic data structures, PBO-FQ is not characterized by the advantage of space efficiency, as PBO-FQ includes a linked list to record indices for achieving fuzzy queries and reducing false positives. Future work will explore user sensitivity to memory cost and compact encoding techniques for reducing memory consumption for this issue.

Fourth, the proposed constructions might increase the attack surface for E2EE services. *Entbergen* reveals a Hamming distance to the server or client. Although it is difficult to speculate on w or $\mathcal{B}$-related information under Hamming distance alone, the Hamming distance still provides additional information, which may be somehow used by advanced attackers for a possible client or server privacy breach. A possible method is integrating oblivious transfer (OT) [50] to the system to hide the Hamming distance for better robustness. However, this will cause a huge latency as the OT protocol is computationally expensive. It would be an interesting topic to explore an effective way to hide the Hamming distance.

VIII. CONCLUSION

Aiming to combat the issue of harmful media moderation in E2EE communications, we propose a novel harmful media identification framework, named *Entbergen*. It explores the prospect of a novel approach – inspecting the media differentially. To achieve this goal, we first design a novel private filter named PBO-FQ. The filter supports checking whether an element is similar to a certain item in a harmful database. Meanwhile, it preserves the harmful materials' privacy by hash-mapping the binary string in the harmful database and adding Laplace noise to the output of the filter. *Entbergen* then utilizes the Paillier cryptosystem and randomization means to privately compute the Hamming distance between the media identified as suspicious by PBO-FQ and its similar hashes to eliminate false positives' impact. The comprehensive evaluation and privacy analysis with varying size sets and lightweight mobile devices demonstrate that *Entbergen* trades acceptable memory consumption for superior performance in computation and communication.

APPENDIX
DETAILED DESCRIPTION OF TECHNIQUES IN SECTION III-C

A. Encoding

The key idea of the encoding operation is to translate the closeness of two hashes into the t-out-of-T set-based matching without sacrificing accuracy [15]. Given a hash string x, the operation encodes it into a set $S = \{s_i | s_i = x \wedge mask_i\}$, where $i \in [1, T]$, $mask$ is a randomly generated projection hash string and has the same length with x, and $\wedge$ denotes AND

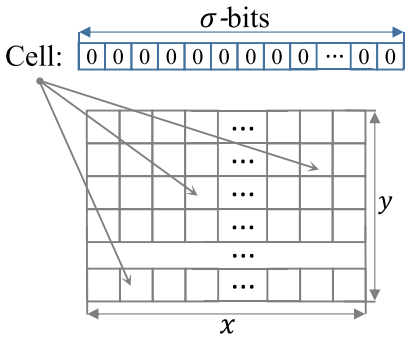


Fig. 11. Structure of the 2DBF.

operation. For another hash string y, it can be encoded into a set S' using identical masks, i.e., $S' = \{s_i | s_i = y \wedge mask_i\}$. Finally, if x and y are Hamming-close, at least t items in S' and S will be the same. For notation conciseness, we use $\mathsf{Encode}(\cdot)$ to denote the sampling algorithm.

B. 2-Dimensional Bloom Filter

As an extension of the conventional BF, the 2DBF possesses a faster query time, lower memory size, and higher accuracy. In particular, 2DBF only requires one hash function and maps each item to one bit [16], [36]. Fig. 11 shows a structure of a 2DBF $\mathbb{B}_{x,y}$, where x and y are dimensions of it. $\mathbb{B}_{x,y}$ contains $x \times y$ cells, and each cell's length is σ-bits, wherein each bit is initialized zero. To insert an item into the filter, we first compute a value h by inputting the item to a hash function. Then, we need to perform three times modulus operations: $i \leftarrow h \bmod x$, $j \leftarrow h \bmod y$, and $p \leftarrow h \bmod \sigma$. i and j jointly locate a particular cell, and p place the item in a specific bit in that cell. Assuming $\mathbb{B}_{i,j,p}$ denotes that bit, the item can be inserted into the filter by setting $\mathbb{B}_{i,j,p} = 1$. To query if an item is in $\mathbb{B}_{x,y}$, the same operations are required to perform. If the particular bit's value equals 1, the input item is a member of the inserted set.

C. Differential Privacy

DP is a theoretical framework for ensuring the privacy of individual-level data when performing statistical analysis of privacy-sensitive datasets. Since DP is a statistical property of the behavior of the randomized function, it is independent of the computational power and auxiliary information available to the adversary. Dwork et al. [51] achieved DP using the Laplace noise mechanism. It can be briefly described as follows. Let $\mathscr{L}(\lambda)$ be the Laplace distribution whose density function is $h(x) = \frac{1}{2}\exp -\frac{|X-\mu|}{\lambda}$ where μ is a mean and $\lambda > 0$ is a scale factor. For a given query function f and a database X, a randomized mechanism M_f that returns $f(X) + Y$ as an answer where Y is drawn from $\mathscr{L}\left(\frac{\triangle f}{\epsilon}\right)$, satisfies ϵ-DP.

D. Paillier Cryptosystem

A cryptographic tool that we use is the Paillier cryptosystem [17]. The security of the Paillier cryptosystem relies on the hardness of the decisional composite residuosity assumption (DCRA), which states that given an integer $n = pq$, where p and q are two large prime numbers, it is computationally infeasible to determine whether a given residue y is a quadratic residue mod n or not.

The basic form of the Paillier scheme involves the following functions:

- **KeyGen:** This function produces the public and private keys. It first selects two large prime numbers p and q such that pq and $(p-1)(q-1)$ are relatively prime. It then computes $n = pq$, and $\lambda = \mathrm{lcm}(\mathrm{p}-1, \mathrm{q}-1)$, where λ is the least common multiple of $p-1$ and $q-1$. After that, it selects a random integer $g \in Z^*_{n^2}$, and the order of g is a multiple of n. Finally, it sets the public key and private key as (n, g) and (p, q, λ).
- **Enc:** For a given plaintext message m, this function returns an encrypted ciphertext c by computing $c = g^m r^n (\mod n^2)$, where r is a random value.
- **Dec:** This function takes an encrypted ciphertext c as input and returns the plaintext message m. The plaintext message $m = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n$, where $L(x) = \frac{x-1}{n}$.

REFERENCES

[1] W. Bai, M. Pearson, P. G. Kelley, and M. L. Mazurek, "Improving non-experts' understanding of end-to-end encryption: An exploratory study," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops*, Sep. 2020, pp. 210–219.

[2] WhatsApp. (2017). *WhatsApp Encryption Overview*. [Online]. Available: https://whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

[3] Apple. (2020). *Apple Platform Security*. [Online]. Available: https://manuals.info.apple.com/MANUALS/1000/MA1902/en_US/apple-platform-security-guide.pdf

[4] Facebook. (2017). *Messenger Secret Conversations*. [Online]. Available: https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf

[5] Telegram. (2018). *Telegram Mobile Protocol*. [Online]. Available: https://core.telegram.org/mtproto

[6] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the signal messaging protocol," *J. Cryptol.*, vol. 33, no. 4, pp. 1914–1983, Oct. 2020.

[7] National Center for Missing & Exploited Children. (2021). *Cybertipline—By the Numbers*. [Online]. Available: https://www.missingkids.org/gethelpnow/cybertipline

[8] Facebook. (2018). *How is Facebook Addressing False Information Through Independent Fact-Checkers?* [Online]. Available: https://www.facebook.com/help/1952307158131536

[9] YouTube. (2022). *How Does Youtube Combat Misinformation?* [Online]. Available: https://www.youtube.com/howyoutubeworks/our-commitments/fighting-misinformation/#policies

[10] U.K. Department for Digital, Culture. (2020). *Online Harms White Paper*. [Online]. Available: https://www.gov.uk/government/consultations/online-harms-white-paper/online-harms-white-paper

[11] EPR Service. (2021). *Liability of Online Platforms*. [Online]. Available: https://www.europarl.europa.eu/RegData/etudes/STUD/2021/656318/EPRS_STU(2021)656318_EN.pdf

[12] A. Kulshrestha and J. Mayer, "Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation," in *Proc. USENIX Secur. Symp.*, 2021, pp. 893–910.

[13] Y. Hua, A. Namavari, K. Cheng, M. Naaman, and T. Ristenpart, "Increasing adversarial uncertainty to scale private similarity testing," in *Proc. USENIX Secur. Symp.*, 2022, pp. 1777–1794.

[14] W. Xue, D. Vatsalan, W. Hu, and A. Seneviratne, "Sequence data matching and beyond: New privacy-preserving primitives based on Bloom filters," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2973–2987, 2020.

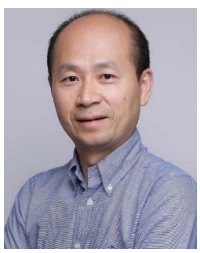
[15] E. Uzun, S. P. Chung, V. Kolesnikov, A. Boldyreva, and W. Lee, "Fuzzy labeled private set intersection with applications to private real-time biometric search," in *Proc. USENIX Secur. Symp.*, 2021, pp. 911–928.

[16] R. Patgiri, S. Nayak, and S. K. Borgohain, "rDBF: A r-dimensional Bloom filter for massive scale membership query," *J. Netw. Comput. Appl.*, vol. 136, pp. 100–113, Jun. 2019.
[17] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1999, pp. 223–238.
[18] E. Bursztein et al., "Rethinking the detection of child sexual abuse imagery on the internet," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 2601–2607.
[19] J. F. Clark. (2019). *Protecting Innocence in a Digital World*. [Online]. Available: https://www.judiciary.senate.gov/imo/media/doc/Clark/Testimony.pdf
[20] Internet Watch Foundation. (2021). *Image Hash List*. [Online]. Available: https://www.iwf.org.uk/our-technology/our-services/image-hash-list/
[21] Canadian Centre for Child Protection. (2017). *Project Arachnid*. [Online]. Available: https://projectarachnid.ca/en/#how-does-it-work
[22] Global Internet Forum to Counter Terrorism. (2017). *Joint Tech Innovation*. [Online]. Available: https://www.gifct.org/joint-tech-innovation/
[23] Facebook. (2019). *The TMK+PDQF Video-Hashing Algorithm and the PDQ Image-Hashing Algorithm*. [Online]. Available: https://github.com/facebook/ThreatExchange/blob/master/hashing/hashing.pdf
[24] Microsoft. (2020). *Photodna*. [Online]. Available: https://www.microsoft.com/en-us/photodna
[25] Facebook. (2019). *Applying to Threatexchange*. [Online]. Available: https://developers.facebook.com/programs/ threatexchange/
[26] B. Coskun and N. Memon, "Confusion/diffusion capabilities of some robust hash functions," in *Proc. 40th Annu. Conf. Inf. Sci. Syst.*, Mar. 2006, pp. 1188–1193.
[27] S. Scheffler and J. Mayer, "SoK: Content moderation for end-to-end encryption," 2023, *arXiv:2303.03979*.
[28] H. Chen, I. Chillotti, Y. Dong, O. Poburinnaya, I. Razenshteyn, and M. S. Riazi, "SANNS: Scaling up secure approximate k-nearest neighbors search," in *Proc. USENIX Secur. Symp.*, 2020, pp. 2111–2128.
[29] Apple Inc. *CSAM Detection—Technical Summary*. [Online]. Available: https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf
[30] L. Liu, D. S. Roche, A. Theriault, and A. Yerukhimovich, "Fighting fake news in encrypted messaging with the fuzzy anonymous complaint tally system (FACTS)," 2021, *arXiv:2109.04559*.
[31] S. Scheffler, A. Kulshrestha, and J. Mayer, "Public verification for private hash matching," in *Proc. IEEE Symp. Secur. Privacy*, Mar. 2023, pp. 2074–2094.
[32] L. L. Gremillion, "Designing a Bloom filter for differential file access," *Commun. ACM*, vol. 25, no. 9, pp. 600–604, Sep. 1982.
[33] O. Rottenstreich, Y. Kanizo, and I. Keslassy, "The variable-increment counting Bloom filter," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1092–1105, Aug. 2014.
[34] A. Kirsch and M. Mitzenmacher, "Less hashing, same performance: Building a better Bloom filter," *Random Struct. Algorithms*, vol. 33, no. 2, pp. 187–218, Sep. 2008.
[35] D. Derler, K. Gellert, T. Jager, D. Slamanig, and C. Striecks, "Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange," *J. Cryptol.*, vol. 34, no. 2, pp. 1–19, Apr. 2021.
[36] R. Patgiri, S. Nayak, and S. K. Borgohain, "PassDB: A password database with strict privacy protocol using 3D Bloom filter," *Inf. Sci.*, vol. 539, pp. 157–176, Oct. 2020.
[37] Y. Hua, B. Xiao, B. Veeravalli, and D. Feng, "Locality-sensitive Bloom filter for approximate membership query," *IEEE Trans. Comput.*, vol. 61, no. 6, pp. 817–830, Jun. 2012.
[38] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Comput.*, 1998, pp. 604–613.
[39] X. Shu, D. Yao, and E. Bertino, "Privacy-preserving detection of sensitive data exposure," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 5, pp. 1092–1103, May 2015.
[40] U.S. Department of Justice. (2021). *International Statement: End-to-End Encryption and Public Safety*. [Online]. Available: https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf
[41] D. Cutting and J. Pedersen, "Optimization for dynamic inverted index maintenance," in *Proc. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 1989, pp. 405–411.
[42] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proc. 14th Annu. ACM Symp. Theory Comput.*, New York, NY, USA, 1982, pp. 365–377.
[43] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, "SCiFI—A system for secure face identification," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 239–254.
[44] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2014, pp. 740–755.
[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.
[46] R. Rothe, R. Timofte, and L. Van Gool, "DEX: Deep expectation of apparent age from a single image," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Dec. 2015, pp. 252–257.
[47] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "WebVision database: Visual learning and understanding from web data," 2017, *arXiv:1708.02862*.
[48] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Trans. Assoc. Comput. Linguistics*, vol. 2, pp. 67–78, Dec. 2014.
[49] H. Abelson et al., "Bugs in our pockets: The risks of client-side scanning," 2021, *arXiv:2110.07450*.
[50] M. Naor and B. Pinkas, "Computationally secure oblivious transfer," *J. Cryptol.*, vol. 18, no. 1, pp. 1–35, Jan. 2005.
[51] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends® Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

Tengfei Zheng received the Ph.D. degree in computer science and technology from the National University of Defense Technology, Changsha, in 2023. His current research interests include AI security and data privacy.

Tongqing Zhou received the bachelor's, master's, and Ph.D. degrees in computer science and technology from the National University of Defense Technology, Changsha, in 2012, 2014, and 2018, respectively. He is currently an Assistant Researcher with the College of Computer, NUDT. His current research interests include ubiquitous computing, mobile sensing, and data privacy.

Kai Lu received the B.S. and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, in 1995 and 1999, respectively. He is currently a Full Professor with the College of Computer, NUDT. His current research interests include operating systems, parallel computing, and security.



Zhiping Cai received the B.Eng., M.A.Sc., and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), China, in 1996, 2002, and 2005, respectively. He is currently a Full Professor with the College of Computer, NUDT. His current research interests include network security and data privacy. He is a Senior Member of CCF.