A preliminary version of this paper appears in *17th International Conference on Applied Cryptography and Network Security (ACNS) 2019.*

Masking Fuzzy-Searchable Public Databases

Alexandra Boldyreva * Tianxin Tang † Bogdan Warinschi ‡

April 27, 2019

Abstract

We introduce and study the notion of *keyless fuzzy search* (KlFS) which allows to mask a publicly available database in such a way that any third party can retrieve content *if and only if* it possesses some data that is "close to" the encrypted data – no cryptographic keys are involved. We devise a formal security model that asks a scheme not to leak any information about the data and the queries except for some well-defined *leakage* function if attackers cannot guess the right query to make. In particular, our definition implies that recovering high entropy data protected with a KlFS scheme is costly. We propose two KlFS schemes: both use locality-sensitive hashes (LSH), cryptographic hashes and symmetric encryption as building blocks. The first scheme is generic and works for abstract plaintext domains. The second scheme is specifically suited for databases of images. To demonstrate the feasibility of our KlFS for images, we implemented and evaluated a prototype system that supports image search by object similarity on a masked database.

1 Introduction

Motivation. Consider an app for finding lookalikes. (This is mostly to gain intuition, we discuss more interesting applications further in the paper.) Using this app, people can post their photos and emails and are willing to be contacted by users who look very similar. Nowadays, image similarity search can be fully automated using modern image recognition/retrieval techniques. The downside is the obvious privacy concerns associated to posting personal information online. As the public is increasingly privacy-cautious, it is strongly desirable to reveal pictures and contact information *only* to lookalikes and not everybody else.

It is not clear how, if at all, existing cryptographic techniques can strengthen privacy of data in the application above. Multiparty computation techniques are not appropriate for this setting. The users are not likely to be all available at the same time to run the protocol. The (public) repository where data is held should neither have access to the raw data. Other cryptographic techniques which rely on secret keys are also not suitable since people who should be able to access the data are not known a-priory and we want to avoid a completely trusted third party in our de-centralized setting.

We propose and rigorously study solutions to this problem. In short, we show how to mask publicly-accessible databases to allow users who know (some information about) what they are looking for to

*Georgia Institute of Technology, USA. sasha@gatech.edu
†Georgia Institute of Technology, USA. ttang@gatech.edu
‡University of Bristol, UK. csxbw@bristol.ac.uk

get that information, yet ensure that mass-harvesting or data mining is prohibitive. From here on, we use "masking" and "encrypting" interchangeably: our methods do not use keys but use (unstructured) data to protect privacy of some content, and the desired hiding properties are somewhat reminiscent of those of encryption.

Narayanan and Shmatikov [36] have proposed obfuscated databases to tackle the same general problem. They treat the case of exact match queries and is not suitable for applications where the match does not have to be exact, such as in the applications we consider. In this paper, we treat the general case of fuzzy queries.

Our results. We propose the concept of *keyless fuzzy search* (KlFS), where a user can query a masked database, retrieve its parts and unmask the content *if only if* it possesses some data "close to" the masked data. We introduce syntax and security models for this primitive and present constructions. We give constructions for the general case (where the structure of the masked data is arbitrary) and for the specific case of image data and show that even without secret keys useful levels of security are possible.

Syntax and Security. The masking algorithm of a KlFS scheme takes inputs *access data* I (e.g. an image of a face) and an auxiliary message M to return a ciphertext C. To query a database of such ciphertexts, a user executes the query algorithm that takes input some access data I' (e.g. an image) and outputs the query. Given the database and the query, the server can efficiently find and return the ciphertexts of all data that have been created with access data "similar" to I' (e.g. all images containing the face in the query). The user can then decrypt and recover the auxiliary message M and optionally the original data I. The formal definition is in Section 3.

For security, we want to capture the idea the attacker obtains information only if it makes the "right" query to retrieve the information. In our formal definition (found in Section 3) we measure if an adversary can compute some useful information about the queries and the underlying with significantly better probability than a simulator who only has access to a *leakage function* of the data. Such definitions (with leakage functions) are common for primitives like searchable encryption and property-preserving encryption [22, 31, 33, 21, 23].

Discussion. Since in our setting there are no keys or trusted parties, security depends on how hard it is for the adversary to come up with data that is close to the one in the masked database. Unlike prior definitions from which we draw inspiration, e.g. those for public-key deterministic encryption [5] and message-lock encryption [8], we do not require this task be computationally infeasible. For the type of applications we envision, message unpredictability is a strong assumptions which is often not true (though later in the paper we discuss a method to improve unpredictability for image encryption). We therefore take a more flexible approach and define security for an arbitrary data set with some (not necessarily negligible) min entropy which we leave as an unspecified parameter, and the difficulty of coming up with an "interesting" query will be reflected by the advantage of the adversary.

This hardness is tightly related to the application domain (how much entropy is there in the stored data) and the closeness threshold which allows unmasking protected data. A KlFS would allow an easy check to see if a specific license plate occurs in a surveillance video recording of an airport parking lot yet, determining all license plates that occur in the video would require exhaustive search. While feasible, it complicates the adversary's goal. Similarly, if KlFS is used to mask a database containing fingerprint readings, then harvesting it would require brute-forcing all possible fingerprints, which could be prohibitive.

The above suggests that in some scenarios KlFS should provide reasonable levels of security, yet it also indicates that the precise level of security may be difficult to assess. Empirical studies for particular datasets could be useful.

We note that it is extremely important that if our schemes get deployed, the users understand that their data is not getting a very strong level of security. The goal is not to hide the data, but to mask it to prevent easy harvesting of information. In other words, we do not show that adversaries

do not exist; we show that adversaries can be *tethered*.

As such, we envision KlFS as an additional layer of protection to be used in conjunction with other mechanisms. For example, the cloud server may be trusted to protect its data storage against malicious compromises with traditional crypto and security tools, but it may not be trusted enough to not mine the data. In this case, the use of a KlFS scheme will protect the data from mass harvesting by the server.

Basic construction. All our constructions use, as a building block, a family of *locality-sensitive hash (LSH)* functions. A randomly chosen LSH function has the property that it collides with high probability when applied to "close" messages, and "far" messages are likely to yield distinct hash values. There are various constructions of LSH families known for different closeness metrics such as Hamming and Euclidean distances, etc. [29, 4, 25]. Most schemes employ the so-called And-Or construction, where a hash value is actually a vector of independent hashes applied to the same input.

The idea behind our Basic KlFS scheme is simple. To mask with access data I a message M we first apply an LSH to I to compute a hash vector $\mathbf{G}$. From $\mathbf{G}$ we compute a vector of tags $\mathbf{T}$ by applying a cryptographic hash function to each entry in $\mathbf{G}$. In addition, we encrypt M (and optionally I) using a standard symmetric encryption scheme under the keys $H(\mathbf{G}[i])$, where $1 \leq i \leq |\mathbf{G}|$ and H is a cryptographic hash function. In practice we recommend to use a slow hash as those used to slow down offline dictionary attacks on passwords, such as a repeated hash.To query I' a user computes the tag vector $\mathbf{T}'$ the same way using the LSH. The server (who indexed the database by the tags) can then efficiently find the required records by the common tags. The user can unmask as the common tag will yield one of the keys used for masking.

KlFS for image search. Since KlFS may be particularly suited to support search on images, we further focus on such data. Existing algorithms for such (unencrypted) search use of *feature* vectors. An image is characterized by a set of such vectors and two images are close if sufficiently many features of the two images are close. What "sufficient" and "close" means is defined by the search algorithm.

Our generic KlFS scheme is not immediately suitable since it cannot take advantage of structured information about the images such as feature vectors. Our high-level idea for an extension is as follows. First, we encode the LSH tags of all extracted feature vectors, so that only their equality is leaked. This way the search algorithm can still identify close feature vectors of database pictures and queried images. Next we encrypt database pictures using the standard symmetric encryption scheme under a key K that can be computed only if one knows a threshold number of feature vectors close to those in the picture or, in other words, possesses an image close to that included in the picture. Technically, we achieve this by secret-sharing the key K (one share per LSH tag) and then encrypting each share with a key deterministically derived from each LSH tag. To unmask, one would need to have an image that shares sufficiently many tags with the image used to mask. We define the scheme in detail and analyze its security (again, in the random oracle model) in Section 5.

Refinements. The security of the scheme described above depends tightly on the closeness unpredictabity of masking data. While some images are reasonably unpredictable, our empirical experiments on some common image datasets showed that feature vectors are likely to be quite predictable. I.e., after trying several images, an attacker will likely have a feature overlap with that of the masked image, and hence will be able to unmask. Another implication is that our search would yield many false positives (approx. 4%), in the sense that each query would receive a fraction of ciphertexts of images that are "technically" close via a common feature, but visually not close. This is a general observation regarding the primitive which we propose: devising KlFS schemes requires careful analysis to ensure that "closeness" as implemented by the schemes corresponds, to the largest extent possible, to "closeness" as desired by applications.

We show how to alleviate this problem for image search. We adapt an "entropy-filling" technique used by Dong et al. [27] to eliminate false positives in image search to work with masked data. We show that it is possible to extract features in a way that best characterizes the images. The technique

filters out the most common features therefore making overlaps between distinct images unlikely. In addition, to improve the true-positive rate while keeping the false-positive rate significantly low, we modify the search algorithm to rely on the PageRank-Nibble algorithm derived from [3] and also used in [27] to improve precision in (unencrypted) image search with very few false positives. We provide more details in Section 6 [12].

Implementation. We realized our findings about KlFS for images as a working prototype system that adds privacy to image search. We describe our implementation and its evaluation in Section 7 [12].

More potential applications. Our KlFS for images can be used directly for searching on masked photo repositories.

Vision-based navigation strategies can greatly increase the scope of application of autonomous mobile vehicles. For example, small Unmanned Aerial Vehicles (UAV) or drones often need to rely on their camera to navigate. Vision-based navigation is necessary in GPS devoid environments or for tracking a visible object; even when GPS is available, vision based localization may offer better precision of self-localization than GPS. Roughly, the process involves acquiring image through the camera, detecting landmarks in current views (edges, corners, objects), matching observed landmarks with those contained in a stored map and calculating the updated position based on the matching results (angle of view, distances between the corresponding points, etc.) and the map's information [13, 39].

In a somewhat similar application, a self-driving car continuously collects a large number of data from its cameras and other sensors. The recorded data may need to be matched against the database of similar recordings made by other vehicles, e.g., to learn about recent traffic conditions and accidents in the area.

In both examples recordings may likely contain information that pose privacy risks, such as recordings of people on sidewalks or in the nearby vehicles. Our primitive would allow to keep the recordings and the associated data in the encoded form and be decoded only by the parties who had recorded data for the same area. Furthermore, depending on the exact use, the people's recordings may never be revealed.

In recent years, the demand for 3D printing is rocketing due to the declining manufacturing costs. 3D printing can be used for printing replacing parts when the parts production is discontinued. Usually customers do not have the expertise either recognizing or describing the small individual part of a complex machinery. Through built-in phone cameras, they can build 3D models based on the damaged parts and then search and retrieve the similar correct 3D models for printing. The KlFS primitive can be useful here since the manufacturers want to limit access to their 3D models.

As the number of devices connected to the Internet grows, network connectivity becomes increasingly critical. Network fault diagnostic and localization is an important research problem. In several proposals [41, 32, 2] users share their local measurements to distinguish important network faults from false positive indications, and to diagnose the root cause of the fault. KlFS will allow to share such information with extra privacy since only the users who experience similar problems in the similar network area will be able to get the information about this area.

We remind that we do not assume that data in the above applications is unpredictable, as it is likely not true. For example, an attacker can possibly obtain aerial images. But this involves some cost. More generally, some entropy in the data will slow down extracting the information from the database, and the data which was not queried about will stay secret.

More related work. Our work is related to the vast literature on efficient searchable encryption, e.g., [24, 31, 11, 21, 23] and especially to fuzzy searchable encryption [33, 10], and to the related areas of property-preserving [38] and structured encryption [22], but all these works are for the symmetric key setting, where a user possesses a secret key. Our focus is on the keyless setting.

Our work is also related to fuzzy vaults [30] and fuzzy extractors [26, 14, 19], even though their main

applications are authorization and key generation based on biometrics. These primitives, however, do not permit efficient search on encrypted data in a remote storage setting, as this would require the users to share a state (helper data).

The works on privacy-preserving data mining, e.g. [35, 34], provide solutions for mining data while preserving hiding privacy of the users. Our goal is different, we want to restrict access to the data for those who do not know what to look for.

Security in the keyless setting has also been considered by the work on message-locked encryption (MLE) [8], which is a generalization of convergent encryption [28]. The main use of these primitives is for secure file de-duplication. KlFS can be viewed as *fuzzy* MLE (MLE ciphertexts leak equality of the underlying plaintexts and KlFS ciphertexts leak their closeness).

From this perspective, KlFS is related to the idea of obfuscation for (fuzzy) point function (PFO) [17], i.e. the task of obfuscating the function which returns the result of the comparison $x \stackrel{?}{=} a$ for some fixed a. A fuzzy PFO (which only reveals closeness of x and a) can be used in the obvious way to implement linear (therefore inefficient) search over encrypted data. The added ability to decrypt can be obtained using (a fuzzy variant of the) multibit output point function obfuscation with auxiliary information (MB-APFO) [18] and refined in later works [16]. While to explore these connections may be theoretically interesting, it is unclear if this would yield efficient enough constructions of KlFS.

2 Preliminaries

Notation and conventions. The algorithms in the following discussions are randomized unless otherwise specified. For some $n \in \mathbb{N}$ we let $[n]$ denote the discrete range $[1, n]$. $x[i]$ denotes the i-th element for some vector, or ordered set x. For a set T we write $|T|$ for the size of T. For some algorithm A that takes inputs $x_1, \ldots, x_n$, its outputs are denoted by $[A(x_1, \ldots, x_n)]$, where each element occurs with positive probability. By true and false, we denote the Boolean values. A random variable X has min-entropy k, denoted by $H_\infty(X) = k$ if $\max_x \Pr[X = x] = 2^{-k}$. For random variables Y, Z we follow Dodis et al [26] and define the conditional min-entropy of Y given Z by $\widetilde{H}_\infty(Y \mid Z) = -\log(\mathbb{E}_{z \leftarrow Z}[2^{-H_\infty(Y|Z=z)}])$. This definition ensures that for any adversary A, $\Pr_{(y,z)\leftarrow(Y,Z)}[A(z) = y] \leq 2^{-\widetilde{H}_\infty(Y|Z)}$.

Symmetric encryption. We assume the reader is familiar with the standard $\mathsf{IND\text{-}CPA}$ security notion for symmetric encryption schemes. For a symmetric scheme $\mathcal{SE}$ we write $\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(A)$ for the advantage of some adversary A in breaking IND-CPA security of $\mathcal{SE}$.

We use symmetric encryption schemes with the additional property that they are key private, i.e., hide all information about the keys used in encryption: it is not possible for an adversary to tell if two ciphertexts of messages are created with the same key, or with different keys. We write $\mathbf{Adv}^{\mathsf{kh}}_{\mathcal{SE}}(A)$ for the advantage of adversary A in distinguishing between these two scenarios.

Secret sharing scheme. One of our construction will use the standard t-out-of-n secret sharing scheme, whose syntax, correctness and security we now recall.

A *t-out-of-n secret sharing scheme* is defined by algorithms $(\mathsf{KS}, \mathsf{KR})$ for sharing and reconstructing a secret (key). For simplicity we assume that the domain of secrets is $\{0,1\}^\kappa$ (where κ is some parameter). The sharing algorithm KS takes a secret s and outputs a set $\{s_1, s_2, \ldots, s_n\}$ of shares. The reconstruction algorithm KR takes as input a set of shares $s_1, s_2, \ldots, s_m$ and returns a string $s \in \{0,1\}^\kappa$ if $m \geq t$, or $\perp$ if $m < t$.

For correctness we demand that for any $s \in \{0,1\}^\kappa$ and any sets $\texttt{Allshares} \in [\mathsf{KS}(s)]$ and $\texttt{Shares} \subseteq \texttt{Allshares}$, where $|\texttt{Shares}| \geq t$, it holds that $\mathsf{KR}(\texttt{Shares}) = s$ with probability 1.

For privacy we demand that for any $s \in \{0,1\}^\kappa$ and set $\texttt{Allshares} \in [\mathsf{KS}(s)]$ it holds that any subset $\texttt{Shares} \subseteq \texttt{Allshares}$ of size $l < t$ does not give any information about s, i.e., its probability

distribution is independent of s. Most popular Shamir's secret sharing scheme unconditionally satisfies both requirements.

Closeness Domains. We adopt the definitions from [10]. We say that $\Lambda = (\mathcal{D}, \mathrm{Cl})$ is a *closeness domain* if

1. $\mathcal{D}$ is a finite or an infinite set;
2. Cl is the (partial) *closeness function* that takes any $x, y \in \mathcal{D}$ and outputs a member of $\{\texttt{close}, \texttt{far}\}$, so that Cl is symmetric (i.e., $\mathrm{Cl}(x, y) = \mathrm{Cl}(y, x)$).

For example, for a metric space $(\mathcal{D}, d)$ and closeness parameters $\delta^{\mathtt{C}}$ and $\delta^{\mathtt{F}}$ we define the closeness domain $(\mathcal{D}, \mathrm{Cl})$ as follows. For $V, V' \in \mathcal{D}$,

$$\mathrm{Cl}(V, V') = \begin{cases} \texttt{close}, & \text{if } d(V, V') \leq \delta^{\mathtt{C}} \\ \texttt{far}, & \text{if } d(V, V') > \delta^{\mathtt{F}} \end{cases}$$

There are no requirements on the output of `close` for pairs that are "near" (i.e. points that neither close nor far).

Locality Sensitive Hashing (LSH). All of our constructions utilize locality-sensitive hashing (LSH), so we start with recalling the LSH primitive introduced in [29]. Below, we give definitions for an arbitrary metric space $(\mathcal{D}, d)$.

Definition 2.1 (Locality-sensitive Hashing) *A family $\mathcal{H}$ is called $(\delta^{\mathtt{C}}, \delta^{\mathtt{F}}, p_1, p_2)$-sensitive if for any two points $x, y \in \mathcal{D}$ [40].*

- *if $d(x, y) \leq \delta^{\mathtt{C}}$ then $\Pr_{\mathcal{H}}[h(x) = h(y)] \geq p_1$,*
- *if $d(x, y) > \delta^{\mathtt{F}}$ then $\Pr_{\mathcal{H}}[h(x) = h(y)] \leq p_2$.*

In this paper we use an extension of LSH which amplifies the accuracy of the parameters via the following construction. The construction, known in the literature as the And-Or construction, is the following.

Definition 2.2 (Extended LSH (eLSH)) *Let $\mathcal{H}$ be an $(\delta^{\mathtt{C}}, \delta^{\mathtt{F}}, p_1, p_2)$-sensitive hash family. For positive integers k, L, choose random $h_{i,j} \in \mathcal{H}$ for all $i \in [L]$, all $j \in [k]$ and define the hash functions $g_i(\cdot)$ by*

$$g_i(x) = (h_{i,1}(x), h_{i,2}(x), \ldots, h_{i,k}(x)) \qquad \text{for all } i \in [L].$$

*We refer to the set of functions g as the (L, k)-*eLSH *extension of $\mathcal{H}$.*

One can think of (L, k)-eLSH extension of $\mathcal{H}$ as an LSH function with improved parameters. The parameters $(\delta^{\mathtt{C}}, \delta^{\mathtt{F}}, P_1, P_2)$ are established by Lemma 2.3 [29] as follows.

Lemma 2.3 *Let $\mathcal{H}$ be a $(\delta^{\mathtt{C}}, \delta^{\mathtt{F}}, p_1, p_2)$-sensitive hash family. Then the (L, k)-*eLSH *extension of $\mathcal{H}$ satisfies the following. Then for $V, V' \in \mathcal{D}$,*
– if $d(V, V') \leq \delta^{\mathtt{C}}$ then

$$\Pr_{h_{i,j} \xleftarrow{\$} \mathcal{H}} [\, \exists i \in [L] : g_i(V) = g_i(V') \,] \geq 1 - (1 - p_1^k)^L = P_1 \ ,$$

– if $d(V, V') > \delta^{\mathtt{F}}$ then

$$\Pr_{h_{i,j} \xleftarrow{\$} \mathcal{H}} [\, \exists i \in [L] : g_i(V) = g_i(V') \,] \leq 1 - (1 - p_2^k)^L = P_2 \ .$$

One construction of an LSH scheme that we use in this paper is for the *Hamming distance* on the set of binary strings of length l, i.e. $\mathcal{D} = \{0,1\}^l$. Starting from a simple LSH function which simply projects on a single bit of its input, i.e. to sample a function from this family simply select a random index $j \in \{1,\ldots,l\}$ and define $h_j(x) = x_j$ (where $x \in \mathcal{D}$ and x_j is the j'th bit of x). It follows that for any two points $p, q \in \mathcal{D}$ collide with probability $1 - \frac{d(p,q)}{l}$, where $d(p,q)$ is the Hamming distance on $\mathcal{D}$. The parameters for the corresponding (L,k)-eLSH are derived using the formulas above.

3 Keyless Fuzzy Search (KlFS)

Syntax for a KlFS scheme. A *Keyless Fuzzy Search (KlFS) scheme* $\texttt{KlFS}$ is defined for a closeness domain $(\mathcal{D}, \mathrm{Cl})$ and message space $\mathcal{MS}$ by six algorithms $\texttt{KlFS} = (\mathsf{Init}, \mathsf{Mask}, \mathsf{Unmask}, \mathsf{Query}, \mathsf{CreateDS}, \mathsf{FuzzyS})$, where,

- Init is a randomized algorithm which outputs a public parameter $P \in \{0,1\}^*$;
- Mask is randomized. It takes P, an element $I \in \mathcal{D}$, and an element $M \in \mathcal{MS}$ and outputs a ciphertext C; We abuse notation and for any subset $\mathbf{D} \subseteq \mathcal{D} \times \mathcal{MS}$ we write $\mathbf{C} \leftarrow \mathsf{Mask}(P, \mathbf{D})$ for the set of ciphertexts obtained by encrypting each $(I, M) \in \mathbf{D}$ using parameters P. We call the elements of $\mathcal{D}$ the *access data* and those of $\mathcal{MS}$ the *auxiliary message*, or simply the message.
- Unmask is deterministic. It takes P, C, an access data $I' \in \mathcal{D}$, and outputs either message M or $\perp$;
- $\mathsf{CreateDS}$, takes a set of ciphertexts $\mathbf{C}$, which we call an (encrypted) database, and outputs a data structure $\texttt{DS}$.
- Query is deterministic. It takes parameters P, *query data* $I \in \mathcal{D}$, and outputs a query T; notice that access data used in encryption and query data live in the same domain.
- FuzzyS is deterministic. On input a database $\mathbf{C}$, data structure $\texttt{DS}$, and query T it outputs a set of ciphertexts.

Notice that we mask messages $M \in \mathcal{MS}$ under access data $I \in \mathcal{D}$ and demand that unmasking returns M (see below). We do not preclude that M contains some, or even all of the information about I.

Correctness and efficiency: We split the correctness requirement of a KlFS scheme in two parts. The first part is concerned with the masking/unmasking algorithms. It demands that unmasking a ciphertext with query data far from the access data used to mask will fail whereas decrypting with data that is close to the original access data will succeed (i.e. return the auxiliary message used to encrypt). Note that the former is needed for filtering out the false positives.

The second part deals with the results returned by a search query. We demand that, searching using some query data I' will not return ciphertexts created with access data that is not close to I; conversely, we demand that the search returns all ciphertexts created with access data that is close to I. All of these requirements need to hold with sufficiently high probability, which is a parameter of the scheme.

ϵ-Correct Decryption: Let $P \xleftarrow{\$} \mathsf{Init}$ be parameters and $(I,\ M) \in \mathcal{D} \times \mathcal{MS}$ and $I' \in \mathcal{D}$ be arbitrary. Let $C \xleftarrow{\$} \mathsf{Mask}(P, I, M)$. Then for all $I, I' \in \mathcal{D}$, all $M \in \mathcal{MS}$

- if $\mathrm{Cl}(I, I') = \texttt{close}$ then
$$\Pr[\mathsf{Unmask}(P, I', C) = M] \geq 1 - \epsilon\ ,$$

- if $\mathrm{Cl}(I, I') = \texttt{far}$

$$\Pr[\mathsf{Unmask}(P, I', C) = \perp] \geq 1 - \epsilon \ .$$

The probabilities are over the choice of P and the coins used by the algorithms involved.

ϵ-Correct Fuzzy Search: Let $P \xleftarrow{\$} \mathsf{Init}$ be parameters, $\mathbf{D} \subseteq \mathcal{D} \times \mathcal{MS}$ be arbitrary and let $\mathbf{C} \xleftarrow{\$} \mathsf{Mask}(P, \mathbf{D})$. Consider the associate data structure $\mathtt{DS} = \mathsf{CreateDS}(\mathbf{C})$, an arbitrary $I' \in \mathcal{D}$ and $T \leftarrow \mathsf{Query}(P, I')$.

We require that:

- For any $(I, M) \in \mathbf{D}$; let C be the resulting ciphertext in $\mathbf{C}$. Then, if $\mathrm{Cl}(I, I') = \texttt{close}$ then

$$\Pr[C \in \mathsf{FuzzyS}(\mathbf{C}, \mathtt{DS}, T)] \geq 1 - \epsilon \ ,$$

- For any $(I, M) \in \mathbf{D}$; let C be the resulting ciphertext in $\mathbf{C}$. Then, if $\mathrm{Cl}(I, I') = \texttt{far}$ then

$$\Pr[C \notin \mathsf{FuzzyS}(\mathbf{C}, \mathtt{DS}, T)] \geq 1 - \epsilon \ .$$

The probabilities are over the choice of P and any coins used by subsequent algorithms. We do not impose a specific bound on ϵ; the correctness analysis for each scheme would need to determine the best value for ϵ, and, of course, one may be able to derive different bounds for each of the four aspects of the correctness definition.

We say that a KlFS scheme is *ϵ-correct* if it satisfies ϵ-correct decryption and ϵ-correct fuzzy search.

We say $\mathtt{KlFS}$ is an *efficiently keyless fuzzy-searchable encryption* (EKlFS) scheme if for any P generated by Init, (sufficiently large) database $\mathbf{C}$, data structure $\mathtt{DS} = \mathsf{CreateDS}(\mathbf{C})$, and query T with $|\mathsf{FuzzyS}(\mathbf{C}, \mathtt{DS}, T)|$ sub-linear in the size of $\mathbf{C}$, the running time of FuzzyS is sub-linear in the size of $\mathbf{C}$. Notice this condition on the running time limits the number of false positives for a fuzzy query.

KlFS Security. We define security of a KlFS scheme using the semantic security approach. As common with such simulation-based definitions for searchable encryption, the definition requires a *leakage function*, which describes whatever the adversary can (unavoidably) glean from the encrypted database, the search data structure and the queries. Since we cannot (and do not want to) fix a one-size-fits-all leakage function, our definition is parametrized by a function leak which takes as input the parameters of the scheme P, the access data $\mathbf{I}$, the auxiliary messages $\mathbf{M}$ and the search queries $\mathbf{Q}$ and outputs some information to be passed to the simulator. Ideally, this information should be as benign as possible, and a scheme designer/user should understand the consequences entailed by leaking this information. Notice that the function depends on the parameters of the scheme which essentially means that the information leaked may vary as a function of the parameters of the scheme.

Our definition can be seen as a non-trivial extension of the semantic-security-based definition for deterministic asymmetric encryption by Bellare et al. [7]. We compare two executions, a real one and an idealized one. In the real execution a database $\mathbf{D}$ and search queries $\mathbf{Q}$ are sampled according to some source $\mathcal{M}$. In addition, we let the source sample some target information $\mathtt{target}$ that models any possible information about the data and the queries the attacker can guess. The adversary is provided with the parameters of the scheme, an encryption of the database the search queries and attempts to guess the target information. We compare this execution with that of an adversary (simulator) who needs to guess the same information but only having as input the information which is allowed to be leaked.

If an ideal adversary exists, then the real adversary cannot learn more from the system beyond the information passed to the simulator. Unlike the traditional security definitions, we do not ask that the ideal adversary perform negligibly close to the real one, as this may not be achievable for some classes of sources. Instead, we let the advantage of the attacker (the difference between its and the ideal adversary's performances) be an arbitrary function of the given resources and the data source. We leave it to applications to estimate whether the given bounds are acceptable.

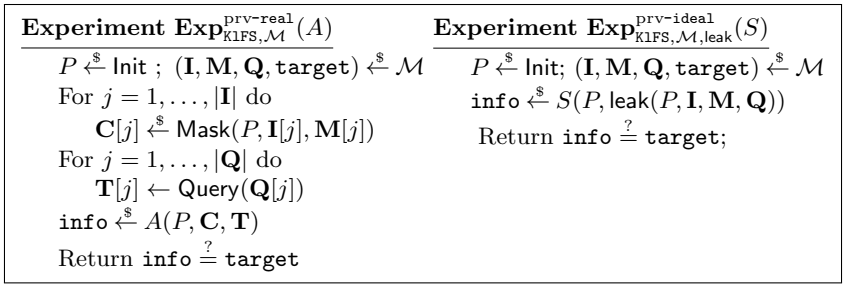

Figure 1: The PRV real (left) and ideal (right) experiments.

Definition 3.1 *For a* KlFS *scheme, closeness domain* $(\mathcal{D}, \mathrm{Cl})$*, source* $\mathcal{M}$*, leakage function* leak*, an adversary* A *with given resources, simulator* S *we define the* prv*-advantage as* $\mathbf{Adv}^{\text{prv}}_{\texttt{KlFS},\mathcal{M},\mathsf{leak}}(A, S)$ *as the difference*

$$\Pr\left[\mathbf{Exp}^{\text{prv-real}}_{\texttt{KlFS},\mathcal{M}}(A) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{prv-ideal}}_{\texttt{KlFS},\mathcal{M},\mathsf{leak}}(S) = 1\right],$$

where the experiments are defined on Figure 1.

Remarks. Note that the above definition is achievable only if the adversary cannot come up with data that is close to the data stored in the database (otherwise, it will be entitled to get the relevant data). This is similar to the requirements of data unpredictability for deterministic and message-lock encryption. We could formally define closeness unpredictability and consider only the sources with such a property. However, our constructions will rely on stronger assumptions so we do not define the minimal assumption for the source and instead define the assumptions required for each scheme.

It is likely that the data set of an application contains data of variable degree of unpredictability. For example, a database of names would have very common names like Adam Smith, somewhat common names like Brent Waters, and rare names like Muthukrishnan Venkitasubramaniam. In this case it makes sense to use the bound on the advantage separately to estimate security for each group, by considering several sources. This would require that there is no correlation between groups (correlations within each group are fine).

We note that our security definition is for a particular source but it is possible to extend the definition to consider a class of sources.

Two remarks are in order regarding the public parameters in the security game. First, our definition only captures security of messages that do not depend on public parameters. This is almost always a reasonable assumption in practice, and is an assumption which is also required in other settings like deterministic and hedged encryption [6] and MLE. Secondly, in our definition the simulator needs to work with honestly generated parameters. One could consider a more permissive definition with a simulator that generates the public parameters of the scheme.

Note that our security notion does capture (though only implicitly) the intuitive goal that it should be harder to extract the entire database than to extract a single entry. The definition demands that the attacker who does not know the right query, gets no information. This means that getting information reduces to coming up with the right queries. Each of these may take time, depending on the underlying message. This brings us back to the essential goal of hardness of retrieving records and tethering the attacker.

4 Basic KlFS

The idea behind the scheme is as follows. The parameters of the scheme consist of an (L, k)-eLSH family. To mask with data I some message M we calculate $g_i(I)$ for each $i \in [L]$ and use these values

in two different ways. First, for each i we derive a key for a standard symmetric encryption scheme by using a hash function H and encrypt M under each of these keys. If the information to be encrypted is large, one may use a "hybrid" scheme where M is encrypted once under a random key K and K is encrypted under each $H(g_i(I))$. As mentioned in the Introduction, in practice we recommend to use a slow hash function to slow down the exhaustive search. In addition, we calculate L tags by applying a (different) hash function G to each $g_i(I)$. The ciphertext of M under I consists of the list of ciphertexts together with the set of tags.

The scheme can support search in a masked database as follows. Given some query data I' one can compute the tags associated to I' (i.e. $G(g_i(I'))$ for each i) to form a query. The server (who can index the database by the tags) can then efficiently locate and return all of the ciphertexts with at least one overlapping tag. The user who is given some data I' close enough to I can then recover at least one of the keys used to mask M by calculating $g_i(I')$ (for all i) and decrypt the ciphertext.

Basic KlFS Scheme. We now define a basic KlFS for any closeness domain $(\mathcal{D}, \mathrm{Cl})$ for which there exists an extended (L,k)-eLSH scheme $\mathcal{H}$ with parameters $(\delta^{\mathtt{C}}, \delta^{\mathtt{F}}, P_1, P_2)$ that are "compatible" with the closeness function, that is for any $I, I' \in \mathcal{D}$ if $\mathrm{Cl}(I, I') = \mathtt{close}$ then $d(I, I') \leq \delta^{\mathtt{C}}$ and if $\mathrm{Cl}(I, I') = \mathtt{far}$ then $d(I, I') \geq \delta^{\mathtt{F}}$. Given a standard symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ we define the Basic KlFS scheme as shown in Figure 2. The initialization algorithm picks two additional hash functions H, G (which we model as random oracles).

Note that although we present search as a linear operation on the database in practice this search is sublinear due to the use of data structures such as K-D trees [15]. In addition, we remark that although the decryption algorithm computes the tags associated to the access data I' used for decryption, in practice this computation is not needed: these tags were computed as part of creating the search query for I' and could be saved to be used in decryption. Moreover, only the "matching" tags could be sent by the server as part of each returned ciphertext.

Correctness and Security. The following theorem establishes the correctness of the basic scheme. Its proof is in Section A [12].

Theorem 4.1 *If $\mathcal{H}$ is an (L,k)-eLSH with parameters $(\delta^{\mathtt{C}}, \delta^{\mathtt{F}}, P_1, P_2)$ then the basic scheme defined above is ϵ-correct, with $\epsilon = \max((1-P_1) + \frac{L}{2^h}, P_2 + \frac{L}{2^h})$, where h is the output length of the random oracle G.*

Next, we analyze the security of the the basic scheme. Each ciphertext consists of a symmetric encryption and a set of tags, each tag is of the form $G(g_i(I))$ (for $1 \leq i \leq L$) and each search query is a collection of tags. We show that the only information that is leaked is the overlap between tags and nothing more, provided a minimal requirement on the interplay between this leakage and the keys used for symmetric encryption.

In our analysis, first we formalize the unavoidable leakage of the scheme, and then spell out and discuss the assumption that the source needs to satisfy.

Given some parameters $P = k||L||g_L(\cdot)$, random oracles G and H and $\mathbf{I}, \mathbf{M}, \mathbf{Q}$, $\mathtt{target} \xleftarrow{\$} \mathcal{M}$ let T be the set of all tags (both associated to ciphertexts and to search queries) that are computed in the experiment. Clearly, the size of T is at most $L \cdot (|\mathbf{I}| + |\mathbf{Q}|)$: each entry in $\mathbf{I}$ and $\mathbf{Q}$ (has at most L associated tags). We can then formalize the information leaked $\mathsf{leak}(P, (\mathbf{I}, \mathbf{Q}))$ as a map $\mathbf{L} : [|T|] \to \mathcal{P}([|\mathbf{I}| + |\mathbf{Q}|]) \times [L]$ which for each tag indicates (the indexes of) the ciphertexts and the queries in which that tag occurs, and the position in the list of tags where it does.

Assume that T is ordered (i.e. lexicographically) and let $T[t]$ be the t'th tag in this order. By abusing notation we write $T[t] \in \mathbf{I}[i]$ to indicate that tag $T[t]$ occurs in the ciphertext associated to $\mathbf{I}[i]$ and we write $T[t] \in \mathbf{Q}[j]$ to indicate that tag occurs in the querry associated to $\mathbf{Q}[j]$. We can then define the information leaked as

$$\mathbf{L}(t) = \{(i,u) \mid \langle u, T[t]\rangle \in \mathbf{I}[i]\} \cup \{\langle j + |\mathbf{I}|, u\rangle \mid \langle u, T[t]\rangle \in \mathbf{Q}[j]\} .$$

Algorithm $\mathsf{Init}_{\mathcal{H}}$
For $i = 1, \ldots, L$ do
For $j = 1, \ldots, k$ do
$h_{i,j}(.) \xleftarrow{\$} \mathcal{H}$; $g_i[j] \leftarrow h_{i,j}(.)$
$\mathbf{g} = (g_1(\cdot), g_2(\cdot), \ldots, g_L(\cdot))$
$P \leftarrow k \| L \| \mathbf{g}$; set random oracles G, H
Return P

Algorithm $\mathsf{Mask}(P, I, M)$
Parse P as $k \| L \| \mathbf{g}$
$\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do
$\mathbf{C}[i] \xleftarrow{\$} \mathcal{E}(H(g_i(I)), M)$
$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(I))\rangle\}$
Return $\mathbf{C} \| \mathbf{Tags}$

Algorithm $\mathsf{Unmask}(P, I', \mathbf{C} \| \mathbf{Tags})$
Parse P as $k \| L \| \mathbf{g}$
For $i = 1, \ldots, L$ do
$T \xleftarrow{\$} G(g_i(I'))$
If $\langle i, T\rangle \in \mathbf{Tags}$ then
$M \leftarrow \mathcal{D}(H(g_i(I')), \mathbf{C}[i])$
Return M
Return $\perp$

Algorithm $\mathsf{Query}(P, I)$
Parse P as $k \| L \| \mathbf{g}$
$\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do
$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(I))\rangle\}$
Return $\mathbf{Tags}$

Algorithm $\mathsf{FuzzyS}(\mathbf{DB}, \mathtt{DS}, \mathbf{Tags}^*)$
$\mathbf{C}_{\mathtt{close}} \leftarrow \emptyset$
For every $i = 1, \ldots, |\mathbf{DB}|$ do
Parse $\mathbf{DB}[i]$ as $\mathbf{C} \| \mathbf{Tags}$
If $\mathbf{Tags} \cap \mathbf{Tags}^* \neq \emptyset$ then
then add $\mathbf{DB}[i]$ to $\mathbf{C}_{\mathtt{close}}$
Return $\mathbf{C}_{\mathtt{close}}$

Figure 2: Algorithms defining Basic KlFS.

Notice that we expressly do not pass $\mathbf{M}$ as input to the leakage function leak since its output $\mathbf{L}$ is independent of $\mathbf{M}$ – this indicates that the scheme leaks no information on the underlying plaintexts.

Next, we identify and explain the assumption on the interplay between the source $\mathcal{M}$ and the parameters of the scheme. Recall that, sensitive data is encrypted under keys of the form $H(g_i(I))$, where H is a random oracle and $g_i(\cdot)$ are hash functions form the extended LSH function $\mathcal{H}$, part of the parameters of the scheme. For security, we need that these keys are unpredictable, even given the information unavoidably leaked by the scheme. That is, for any $g_i(\cdot)$ (sampled from $\mathcal{H}$) and for any index $j \in [|\mathbf{I}|]$ and any index $k \in [|\mathbf{Q}|]$ we have that $\widetilde{H}_\infty(g_i(\mathbf{I}[j] \mid \mathsf{leak}(P, \mathbf{I}, \mathbf{Q}))) \geq l$ for some sufficiently large l. To simplify notation, and avoid the multiple quantifiers we write $\widetilde{H}_\infty(\mathcal{H}(\mathcal{M}) \mid \mathbf{L}(\mathcal{M})) \geq l$ for this requirement. Notice that this requirement is strictly stronger than closeness-unpredictability of $\mathcal{M}$.

The next theorem (which we prove in Section B [12]) establishes the security of the basic scheme, namely that it leaks no information beyond the tag overlap, unless the attacker can predict the tags. This holds under the assumption that the symmetric encryption scheme used in the implementation hides the plaintext and is *key-private*. This latter assumption is needed since otherwise, ciphertexts will leak information about equality of keys which translates to more specific information about equality of tags than leaked by $\mathbf{L}$: an adversary could tell not only that there are tag overlaps, but can tell to which keys these tags correspond.

Theorem 4.2 *Let* $(\mathcal{D}, \mathrm{Cl})$ *be a closeness domain. Let* $\mathcal{M}$ *be an arbitrary source and let* $\mathcal{H}$ *be a*

compatible (L,k)-eLSH *scheme with parameters* $(\delta^{\mathsf{C}},\delta^{\mathsf{F}},P_1,P_2)$ *such that* $\widetilde{H}_\infty(\mathcal{H}(\mathcal{M}) \mid \mathbf{L}(\mathcal{M})) \geq l$. *Let* $\mathcal{SE}=(\mathcal{E},\mathcal{D})$ *be a symmetric encryption scheme. We assume that keys for the scheme are bitstrings length* l *selected uniformly at random. Let* Π *be the Basic KlFS and let* leak *be the leakage function defined above. Then for any adversary* A*, we construct a simulator* S *such that there exist adversaries* B *and* C *so that, in the random oracle model,* $\mathbf{Adv}^{\mathrm{prv}}_{\Pi,\mathcal{M},\mathsf{leak}}(A,S)$ *is upperbounded by*

$$\mathbf{Adv}^{kh}_{\mathcal{SE}}(B)+\mathbf{Adv}^{ind\text{-}cpa}_{\mathcal{SE}}(C)+\frac{L\cdot(q_G+q_H)\cdot(|\mathbf{I}|+|\mathbf{Q}|))}{2^l}\ ,$$

where q_G *is the number of queries that* A *makes to oracle* G*. Furthermore, the running times of* S*,* B *and* C *are essentially that of* A*; the number of encryption queries that* B *and* C *make is* $|\mathbf{I}|$.

Discussion. As the theorem above states, evaluating security requires estimating unpredictability of LSH tags, and we understand this is a difficult task. Evaluation of this property has to be done for the specific LSH instantiation. For example, for the aforementioned random-bit-projection LSH construction for Hamming distance, it is known [42, 19] that the rate of source unpredictability is preserved by random random projections (or *samples*, using the terminology of [19]). It is shown in [19] that for some specific sources it is possible to preserve more entropy. In addition, one still has to estimate the unpredictability of the data (empirically or otherwise).

Similarly, it may not be easy to evaluate the implications of the leak function, a key challenge in studying property-preserving encryption in general. Hopefully future works will bring novel methods that facilitate such analysis. For our case, further work is needed to understand how leakage about tags translates into leakage about the data, but this requires a case by case analysis, depending on the use of a particular LSH and closeness domain.

For the case of random-bit-projection LSH, leak implies leaking the "overlap pattern" of LSH tags. In particular, each LSH tag (for eLSH construction) is a list of k bits. The attacker will learn to which data each tag corresponds to, but it does not learn what each tag is or what random bit positions each tag corresponds to (the latter is due to keeping tags as sets as opposed to lists, and by employing key-private encryption). An interesting challenge would be to see empirical inference attacks in the style of [20, 37] which may rely on domain specific knowledge.

Also recall that the definition of the source implies that we only ensure security for messages that do not depend on public parameters. This is a rather reasonable assumption in practice and moreover, it is possible future research will remove this assumption, similarly to the case of MLE [1].

As we explained in Section 3, the bound can be used to estimate security of data with different entropy, if we consider several independent sources and assume that the data produced by different sources is not correlated across different sources.

5 KlFS for Fuzzy Image Search

Feature vectors. Most algorithms for image search deal with image *feature vectors*. Feature vectors are small pieces of data containing information about the image or parts of the image, such as color, shape, object boundaries, etc. Some applications may need to work with several types of features. For simplicity, in this work we focus on feature vectors of the same type. We remark that our definitions could easily be extended to handle multiple types of feature vectors. E.g., one could assign a specific index to indicate which type the feature vector belongs to.

We consider a domain of feature vectors $\mathcal{V}$ and assume that there exists an efficient deterministic algorithm $\mathsf{extractV}$ that takes an image $I\in\mathcal{D}$ and outputs a set of feature vectors $\mathbf{V}\subset\mathcal{V}$ (such algorithms are well-documented in the computer vision and graphics literature).

Intuition for the Scheme. We aim to add security to the existing image search applications. In one such application, a user can search a database of pictures by an image of a face. The user is able

to retrieve the pictures containing the person in question. More generally, a user holding an image I should be able to find database pictures that contain (as part of the picture) an image close to I in some metrics. As we discussed in the Introduction, the existing algorithms for unencrypted search work roughly by determining how many features are close between those for query and database data and this is done by comparing equality of LSH tags.

Our goal is to let such algorithms work on masked data. I.e., we want to hide information about the database and the queries besides the information necessary for efficient search, such as similarity of underlying feature vectors. And again, since we are working in the keyless setting, security depends on how hard it is to predict the images. Of course, we want to state and prove the exact security guarantees for our construction, even though we do not expect the security guarantees to be very strong (as we also have functionality and efficiency considerations on the other side of the scale).

Since our general KLFS scheme from Section 4 is not immediately suitable (mainly because it does not consider feature vectors), we propose a scheme tailored for the task. First, we encode the LSH tags of all extracted feature vectors, so that only their equality is leaked. This way the search algorithm can still identify close feature vectors of database pictures and queried images. Next we encrypt database pictures using the standard symmetric encryption scheme under a key K that can be computed only if one knows a threshold number of feature vectors close to those in the picture or, in other words, possesses an image close to that included in the picture. We achieve this by secret-sharing the key K and for each feature encrypting the corresponding share with a key deterministically derived from each LSH tag. We now provide the details.

The Construction. We consider closeness domain $\Lambda = (\mathcal{D}, \mathrm{Cl})$, where $\mathcal{D}$ is a domain of images and Cl determines when two images are close for a match. The latter can depend on the application. (See Section 7 for a concrete example.) We assume the existence of deterministic algorithm $\mathsf{extractV}$ that takes an image in $\mathcal{D}$ and outputs a set of feature vectors $\mathbf{V}$. We also assume that Cl defines the parameter $\mathtt{thr}$ which is the number of close features needed to determine a match (closeness) between two images. The construction will use a $(\delta^{\mathsf{C}}, \delta^{\mathsf{F}}, p_1, p_2)$-sensitive hash family $\mathcal{H}_{L,k}$ with parameters L and k, matching the closeness domain as defined for the general schemes, cryptographic hashes H, G (will be treated as random oracles in the security analysis), a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and a secret sharing scheme $(\mathsf{KS}, \mathsf{KR})$. We remark that in the secret sharing scheme, the parameter n (from t-out-of-n) will vary and will be determined in the construction.

The parameter generation algorithm $\mathcal{P}$ is as of the Basic KlFS scheme. The rest of the algorithms are defined in Figure 3. Similarly to the Basic KlFS description, we do not specify in Figure 3 how FuzzyS makes use of the data structure $\mathtt{DS}$ or that the server could only return the matched tags. And in practice unmasking can be sped up if the user stores the tags (and their corresponding indices) so they are not re-computed during decryption.

Correctness and Security. The correctness of fuzzy search is as of the Basic KlFS. Correctness of decryption is similar to that of the Basic KlFS, but it also relies on correctness and security of the key sharing scheme. Specifically, correctness of the latter ensures that the threshold number of shares are sufficient to reconstruct the key, which in turn will ensure that decryption using an image close to the one used to encrypt will be correct. Decryption with a "far" image fails due to the use of the key sharing scheme: in this case the decryptor will not have enough shares. We observe that security of key sharing is actually stronger than what we need here (failure of key reconstruction with insufficient number of shares), as correctness is not an adversarial notion.

Before we specify the security of the scheme, we formalize the information that we expect that the scheme leaks. Given some parameters $P = k||L||g_L(\cdot)$, random oracles G and H and $(\mathbf{I}, \mathbf{M}, \mathbf{Q}, \mathsf{aux}) \xleftarrow{\$} \mathcal{M}$ we define the leakage function $\mathsf{leak}(P, (\mathbf{I}, \mathbf{Q}))$ as follows. (As for the previous scheme we do not pass $\mathbf{M}$ as input to the leak function to indicate that the information revealed by the scheme does not depend on $\mathbf{M}$.) We let F be the (lexicographically ordered) set of features associated to the images in $\mathbf{I}, \mathbf{Q}$; we write $F[i]$ for the i'th feature in F and let $f = |F|$. Let T be the (lexicographically ordered)

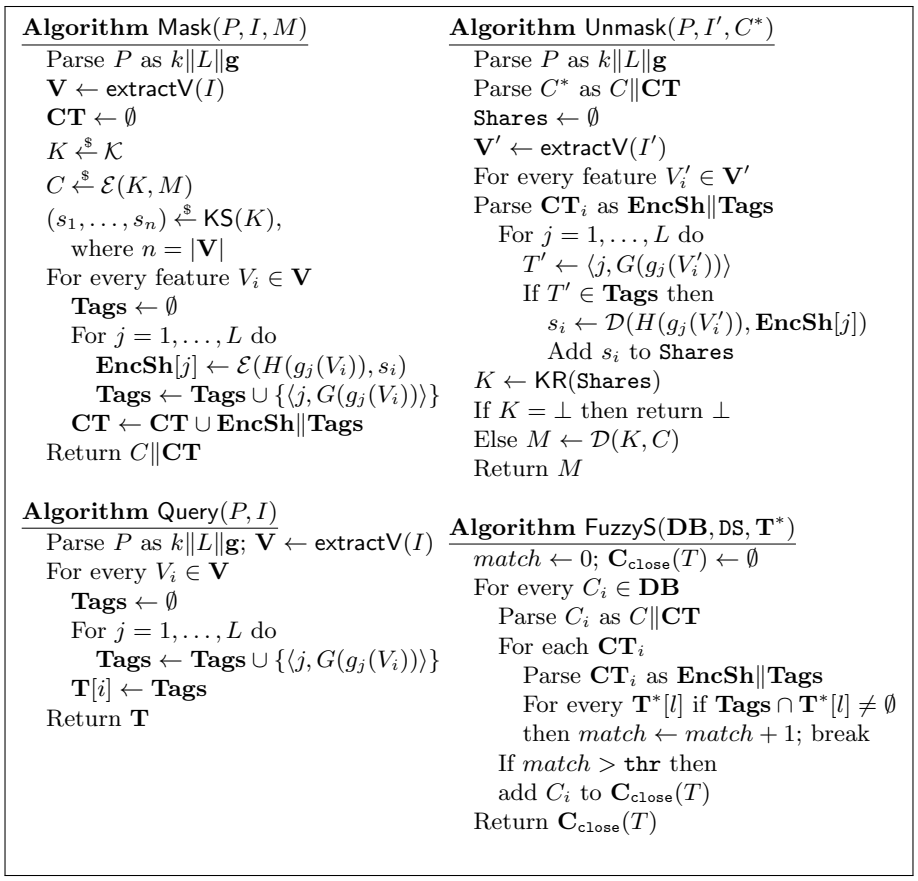

Figure 3: Algorithms defining the KlFS for Images.

set of tags associated to the features above; we let $T[i]$ be the i'th tag and let $n = |T|$. Define the matrix M of $|\mathbf{I}| + |\mathbf{Q}|$ rows, and f columns where the entry on row i and column j of matrix M is the list of tags associated to feature $F[j]$ if $F[j]$ is a feature of $\mathbf{I}[i]$. In other words l is part of the list $M(i, j)$ if $F[j]$ is a feature of $\mathbf{I}[i]$ and $T[l]$ is a tag derived from $F[j]$. The leakage function $\mathbf{L} : [n] \to \mathcal{P}(([|\mathbf{I}| + |\mathbf{Q}|]) \times [f] \times [L]))$ is defined by $\mathbf{L}(t) = \{(i, j, u) \mid \langle u, T[t]\rangle \in M(i, j)\}$. Informally, the function reveals for each tag (identified by an index $t \leq n$) all access data or query entries (identified by $i \in [|\mathbf{I}| + |\mathbf{Q}|]$) and all features (identified by some $j \leq f$) for which the tag was derived from feature j belong to access data (or query) i.

The security theorem below establishes that unless the attacker guesses successfully some tag, provided the unavoidable leakage of the scheme, no information is leaked about the data that is masked. Specifically, we assume that we know l such that for any fixed g_i of $\mathcal{H}$, and $j \in [|\mathbf{I}|]$ if we let $V_k(\mathbf{I}[i])$ be the k'th feature extracted by $\mathsf{extractV}$ from $\mathbf{I}[i]$, then $\widetilde{H}_\infty(g_i(V_k(\mathbf{I}[j]) \mid \mathsf{leak}(P, \mathbf{I}, \mathbf{Q}))) \geq l$: that is there is sufficient entropy left in the (LSH projection) of each feature vector, even given the inherent leakage of the scheme (i.e. the different feature vector overlaps). We make the analogous requirement for $\mathbf{Q}$ by abuse of notation we write $\widetilde{H}_\infty(\mathcal{H}(\mathsf{extractV}(\mathcal{M})) \mid \mathbf{L}(\mathcal{M})) \geq l$ for the resulting condition.

Theorem 5.1 *Let $(\mathcal{D}, \mathrm{Cl})$ be a closeness domain. Let $\mathcal{M}$ be an arbitrary source and let $\mathcal{H}$ be a (L, k)-eLSH scheme with parameters $(\delta^{\mathsf{C}}, \delta^{\mathsf{F}}, P_1, P_2)$ such that $\widetilde{H}_\infty(\mathcal{H}(\mathsf{extractV}(\mathcal{M})) \mid \mathbf{L}(\mathcal{M})) \geq l$. Let*

$\mathcal{SE}$ be a symmetric encryption scheme, let Π-IM be the KlFS described above Then for any adversary A, we construct a simulator S and adversaries B and C so that $\mathbf{Adv}^{\mathrm{prv}}_{\Pi\text{-}IM,\mathcal{M},\mathsf{leak}}(A,S)$ is upperbounded by

$$\mathbf{Adv}^{kh}_{\mathcal{SE}}(B) + 2\mathbf{Adv}^{ind\text{-}cpa}_{\mathcal{SE}}(C) + \frac{L \cdot f \cdot (q_G + q_H) \cdot (|\mathbf{I}| + |\mathbf{Q}|)}{2^l} ,$$

where leak *is as defined above, and q_H and q_G are the number of random oracle calls to H and G. Furthermore, the running time of S, B and C are essentially that of A; the number of queries that B and C make to the encryption oracle is $|\mathbf{I}| \cdot f \cdot L$ (where f is the maximum number of features per image).*

Discussion. As we discussed in Section 4, it is important to provide general means to further understand the extent and implications of leakage. Meanwhile, to gain more intuition about the leakage function, assume that the database contains 2 similar images I_1, I_2, so that I_1 and I_2 have close features V_1, V_2, V_3. Further assume that V_1, V_2 share LSH tags t_1, t_2; V_1, V_3 share LSH tags t_1, t_3; and V_1, V_3 share t_4. Then, according to leak function we defined, the adversary learns exactly that, namely the Venn diagram of the set of tags overlaps.

One could strengthen our theorem by relaxing the tag unpredictability requirement. Instead, one could require that only the threshold (from the key sharing scheme) tags be unpredictable as opposed to each individual one. In this case, the security of the scheme will also rely on security of the key sharing scheme.

6 New KlFS for Images

Empirical study of the Basic KlFS for images. We implemented the KlFS for images from the previous section. We implemented the random bit projection LSH. We used the feature extraction algorithm from OpenCV 2.4.13. We limited the number of extracted features to 200 for each image and considered images close if they have at least two close ORB features (have an overlapping tag).

We experimentally evaluated the security of our scheme and found that it does not provide reasonable security, without any contradiction with the theoretical results. The problem is not with the scheme or its analysis. The problem is that the assumption on which security relies on is not true for the data sets we have experimented with. I.e., the feature vectors and hence the tags are predictable, in that images that are not visually close end up being "technically" close since their feature vectors overlap. An attacker can try several images until one of the features will match a feature from the masked image, and then the attacker will succeed.

New KlFS for images. To address the problem, we modify the basic KlFS scheme for images to include "entropy-filtering" and "query expansion" techniques introduced by Dong et al. [27].

Entropy-filtering is a technique which aims to eliminate false positives by filtering the features associated to images. More specifically, after extracting SIFT features from each image, a customized algorithm is used to remove the common features, leaving the more unique ones that better describe the image. This technique helps to decrease the rate of false positives wrt "visually" far images and, as we discuss later also improves security. The downside is that, the technique also decreases the precision (the true-positive rate wrt "visually" close images).

To improve precision, we adopt a "query-expansion" strategy and the corresponding PR-Nibble algorithm used in [27] for the same goal but for unencrypted images. We modify the CreateDS algorithm as follows. The server first creates a search graph DS on top of the masked database **DB**, which utilizes the tag-overlapping pattern of the data. Each masked data is indexed as a vertex, and any pair of masked data sharing at least one tag is connected by an (undirected) edge. Essentially, the server is capable of building such a graph by running the basic KLFS search algorithm on each masked data in the database. The details are in Fig. 5.

The FuzzyS algorithm also creates a temporary data structure $\mathtt{DS}'$, which is initialized as a temporary copy of the data structure DS. It then updates $\mathtt{DS}'$ by adding a new vertex representing Q, and creates new edges between Q and every masked data in $\mathtt{DS}'$ sharing at least one overlapping tag with Q. After that, the server runs an approximate PageRank algorithm as part of the PR-Nibble algorithm on query Q, database $\mathbf{DB}$, and the search graph $\mathtt{DS}'$. The final output of algorithm PR-Nibble contains a set of masked data that is reachable from the query data Q through edges in $\mathtt{DS}'$, while minimizing the "conductance" property, i.e., the ratio between the number of edges connecting the search results (cluster which lies the query image Q) and "visually far" clusters, and the number of the intra-connections of the search results. Since edges establish the closeness relation between any pair of images based on whether they share close features, we can interpret the objective for minimizing the "conductance" as follows: smaller nominator means fewer "visually far" images are captured, and larger denominator implies the search results capture more "visually close" ones with high probability. We outline the algorithms in Fig. 4 and refer readers to [27] for more details. Since minimizing "conductance" is the standard objective for Approximate PageRank algorithms with approximation error par-ϵ and damping factor par-$\alpha \in (0, 1]$ as input. We refer readers to PageRank related literature for parameter selection [3]. Typically, par-α is set to 0.85, and both parameters can be chosen empirically as in [27].

For *correctness* of our scheme, we recall the difference between two correctness notions: a "technical" notion that relies on the distance of the feature vectors and the number of matching features; and a more practical notion dealing with images that are "visually close". The "technical" correctness is as that for the basic KlFS for images. The "visual" correctness can only be justified empirically: we provide the results in the next section.

For *security* we note that the bound is the same as for the basic KlFS scheme since the modified algorithms output strictly less data (and therefore information). What changes, is the assumptions on the unpredictability of masking data – we argue below that with the modifications spelled out above, tags closeness unpredictability is now a realistic assumption. Indeed, the extractV algorithm removes the common features from the images, which leads to improving the closeness-unpredictability of the features, thus enhances the closeness-unpredictability of the LSH hashes. Under the random oracle model, with practical instantiation using a collision-resistant hash function such as SHA256 or some slow hashes, the new scheme makes it harder for the adversary to come up with LSH hashes matching the tags in the database, to recover the keys and break the security. We validate this observation empirically through experiments.

7 Experimental Results

We implemented the revised KLFS scheme for Images with cryptographic library Crypto++6.5.4 in C++ on Ubuntu 16.04 with a 6-core processor (Intel® Core™ i7-8750H CPU @ 2.20GHz ×12), 16GB RAM and demonstrated that the masking scheme was practical. In particular, we used SHA3-256 hashing (with different prepended bits to realize two independent hashes), AES-CBC encryption with random IV and Shamir's secret sharing. We did not use a slow hash, but plan to experiment with it in the future. We used the same image dataset as in [27]. The test database contains 10839 images of famous paintings and CD covers in total, and are divided into 81 groups. In each group, images were manually checked to ensure that the were visually close.

We implemented the extractV algorithm using the techniques in [27] to extract the most representative SIFT features and transform them into Hamming space. We chose to count any two features with more than 3-bit difference out of 128 bits as "far", and setting a single "close" feature match as the threshold for an image match in the experiment to produce significantly low rate of false-positives regarding the "visual closeness". To choose the projection LSH parameters, we knew that the parameter k, the number of bits of each LSH hash is highly related to security since the adversary can

Algorithm $\mathsf{Mask}(P, I, M)$

Parse P as $k\|L\|\mathbf{g}$
$\mathbf{V} \leftarrow \mathsf{extractV}(I)$
$\mathbf{CT} \leftarrow \emptyset$
$K \xleftarrow{\$} \mathcal{K}$
$C \xleftarrow{\$} \mathcal{E}(K, M)$
$(s_1, \ldots, s_n) \xleftarrow{\$} \mathsf{KS}(K)$,
  where $n = |\mathbf{V}|$
For every feature $V_i \in \mathbf{V}$
  $\mathbf{Tags} \leftarrow \emptyset$
  For $j = 1, \ldots, L$ do
    $\mathbf{EncSh}[j] \leftarrow \mathcal{E}(H(g_j(V_i)), s_i)$
    $\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{G(g_j(V_i))\}$
  $\mathbf{CT} \leftarrow \mathbf{CT} \cup \mathbf{EncSh}\|\mathbf{Tags}$
Return $C\|\mathbf{CT}$

Algorithm $\mathsf{Unmask}(P, I', C^*)$

Parse P as $k\|L\|\mathbf{g}$
Parse C^* as $C\|\mathbf{CT}$
$\mathtt{Shares} \leftarrow \emptyset$
$\mathbf{V}' \leftarrow \mathsf{extractV}(I')$
For every feature $V_i' \in \mathbf{V}'$
Parse $\mathbf{CT}_i$ as $\mathbf{EncSh}\|\mathbf{Tags}$
  For $j = 1, \ldots, L$ do
    $T' \leftarrow G(g_j(V_i'))$
    If $T' \in \mathbf{Tags}$ then
      $s_i \leftarrow \mathcal{D}(H(g_j(V_i')), \mathbf{EncSh}[j])$
      Add s_i to $\mathtt{Shares}$
$K \leftarrow \mathsf{KR}(\mathtt{Shares})$
If $K = \perp$ then return $\perp$
Else $M \leftarrow \mathcal{D}(K, C)$
Return M

Algorithm $\mathsf{Query}(P, I)$

Parse P as $k\|L\|\mathbf{g}$; $\mathbf{V} \leftarrow \mathsf{extractV}(I)$
For every $V_i \in \mathbf{V}$
  $\mathbf{Tags} \leftarrow \emptyset$
  For $j = 1, \ldots, L$ do
    $\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{G(g_j(V_i))\}$
  $\mathbf{T}[i] \leftarrow \mathbf{Tags}$
Return $\mathbf{T}$

Algorithm $\mathsf{FuzzyS}(\mathbf{DB}, \mathtt{DS}, \mathbf{T}^*)$

$\mathtt{DS}' \leftarrow \mathsf{CreateTempDS}(\mathbf{DB}, \mathtt{DS}, \mathbf{T}^*)$
$\mathbf{C}_{\mathtt{close}}(T) \leftarrow \mathsf{PR\text{-}Nibble}_{\text{par-}\alpha,\text{par-}\epsilon}(\mathbf{DB}, \mathtt{DS}', \mathbf{T}^*)$
Return $\mathbf{C}_{\mathtt{close}}(T)$

Figure 4: Algorithms defining the revised KlFS for Images.

compute the hashes by brute force. Consequently, we need to choose $k > 60$. Given k, based on the equation for theoretic bounds of eLSH extension, we can compute the parameter L accordingly with reasonably high P_1 and low P_2. Since we are only dealing with the "closeness-unpredictable" distribution instead of arbitrary distribution the eLSH theoretic bounds apply, we can reduce L significantly. We found out that since the closeness threshold of filtered features was small, then choosing $k = 80$ and $L = 8$ was sufficient in our experiment and the LSH application did not increase the rate of false positives.

Based on the "visual correctness" notion, we select 5 most representative images per group, and test in the same database, with closeness threshold 3 out of 128 bits on each feature; closeness threshold 1 on each image, i.e., one close feature match will result in one close image match. Average TP and FP is computed by averaging all TPs, FPs of each group respectively. Then without LSH tags, average TP = 0.65 and FP = 0; with LSH tags, average TP = 0.69 and FP = 0. The average of TP is not high, but still practical in some applications that we mentioned earlier. That no false-positive results were returned is partially due to the small image database we were working on. Still, the "entropy-filtering" and "query expansion" techniques do scale, shown by [27] testing against one million random Flicker images as background images, i.e., "visually far" images to each test image and the average TP = 0.79 and FP = 2.5×10^{-6}. The spatial overhead of our revised KLFS scheme is roughly 400%, same as the basic scheme, but with an additional graph structure which takes negligible amount of space (e.g., 5.8 MB). The extractV algorithm and tags generation take less than a second per image; average time for encryption per image including the encryption of tags is 0.3 second. Average time for decryption per image is 0.008 second. CreateDS takes about an hour to complete on the whole data set. The average processing time of PR-Nibble algorithm running on the graph structure per query is 0.006 second to retrieve the file ids. The overall significantly low false-positive rate of the result against

Algorithm $\mathsf{TagsUnion}(C^*)$
$\mathbf{Tags}^* \leftarrow \emptyset$
Parse C^* as $C \| \mathbf{CT}$
For each $\mathbf{CT}_i$
  Parse $\mathbf{CT}_i$ as $\mathbf{EncSh} \| \mathbf{Tags}$
  $\mathbf{Tags}^* \leftarrow \mathbf{Tags}^* \cup \mathbf{Tags}$
Return $\mathbf{Tags}^*$

Algorithm $\mathsf{CreateTempDS}(\mathbf{DB}, \mathtt{DS}, \mathbf{T}^*)$
Parse $\mathtt{DS}$ as $(\mathbf{V}, \mathbf{Ad})$
$\mathtt{id}_\mathrm{Q} \leftarrow |\mathbf{V}| + 1$
$\mathbf{V} \leftarrow \mathbf{V} \cup \{\mathtt{id}_\mathrm{Q}\}$
$\mathbf{Ad}_{\mathtt{id}_\mathrm{Q}} \leftarrow \emptyset$
For each $C_i \in \mathbf{DB}$
  $\mathbf{Tags} \leftarrow \mathsf{TagsUnion}(C_i)$
  If $\mathbf{T}^* \cap \mathbf{Tags} \neq \emptyset$ then
    $\mathbf{Ad}_{\mathtt{id}_\mathrm{Q}} \leftarrow \mathbf{Ad}_{\mathtt{id}_\mathrm{Q}} \cup \{i\}$
    $\mathbf{Ad}_i \leftarrow \mathbf{Ad}_i \cup \{\mathtt{id}_\mathrm{Q}\}$
$\mathtt{DS}' \leftarrow (\mathbf{V}, \mathbf{Ad})$
Return $\mathtt{DS}'$

Algorithm $\mathsf{CreateDS}(\mathbf{DB})$
$\mathbf{V} \leftarrow \emptyset$
For each $C_i \in \mathbf{DB}$
  $\mathbf{V} \leftarrow \mathbf{V} \cup \{i\}$
For $i = 1, \ldots, |\mathbf{V}|$ do
  $\mathbf{Ad}_i \leftarrow \emptyset$
For each $C_i \in \mathbf{DB}$
  $\mathbf{Tags}^{(i)} \leftarrow \mathsf{TagsUnion}(C_i)$
For each $(\mathbf{Tags}^{(i)}, \mathbf{Tags}^{(j)})$ and $i \neq j$ and $\mathbf{Tags}^{(i)} \cap \mathbf{Tags}^{(j)} \neq \emptyset$
  $\mathbf{Ad}_i \leftarrow \mathbf{Ad}_i \cup \{j\}$
  $\mathbf{Ad}_j \leftarrow \mathbf{Ad}_j \cup \{i\}$
$\mathtt{DS} \leftarrow (\mathbf{V}, \mathbf{Ad})$
Return $\mathtt{DS}$

Figure 5: Algorithms defining the revised KlFS for Images.

public database shows that our scheme prevents the computational bounded adversaries harvesting data from a masked database at a scale.

We observe that the true-positive rate is not very high, but this is as expected, given the necessity for almost no false positives and inability to execute advanced search techniques on unencrypted images use because the data is masked (such as, for example, feeding images into the trained deep neural networks). Still, for data domains where similar images are closely clustered together and clusters are reasonably far apart low true-positive rate may be sufficient. For example, for applications which needs to check for the presence of a specific and highly distinct image in a database where the multiples variations of that picture are present (and therefore does not need to recover all occurrences of that image). This is the case, for example, when determining if a particular human face, license plate number, animal, or logo appears in a collection of frames of a given video. Similarly, in machine learning applications, if an image (with objects or scenes) needs to be classified using a database of labeled images, then it is enough to match the image with the most likely class, and it is not necessary to match all close images.

8 Acknowledgements

We thank Dima Damen, Walterio Mayol Cuevas, Hugo Krawczyk, Leo Reyzin, Tom Ristenpart and Dan Shepard for useful comments and suggestions. We also thank the anonymous reviewers. Alexandra Boldyreva and Tianxin Tang were supported in part by NSF 1422794 and 1749069 awards.

References

[1] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev. Message-locked encryption for lock-dependent messages. In *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 374–391.

Springer, 2013.

[2] B. Agarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker. Netprints: Diagnosing home network misconfigurations using shared knowledge. In *NSDI*, pages 349–364. USENIX Association, 2009.

[3] R. Andersen, F. R. K. Chung, and K. J. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, pages 475–486. IEEE Computer Society, 2006.

[4] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468. IEEE Computer Society, 2006.

[5] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2007.

[6] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 2009.

[7] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 360–378. Springer, 2008.

[8] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 296–312. Springer, 2013.

[9] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT 2006 Proceedings*, pages 409–426, 2006.

[10] A. Boldyreva and N. Chenette. Efficient fuzzy search on encrypted data. In *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 613–633. Springer, 2014.

[11] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 578–595. Springer, 2011.

[12] A. Boldyreva, T. Tang, and B. Warinschi. Masking fuzzy-searchable public databases. Full version of this paper. Available at ePrint archive `https://eprint.iacr.org/`., 2019.

[13] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008.

[14] X. Boyen. Reusable cryptographic fuzzy extractors. In *ACM Conference on Computer and Communications Security*, pages 82–91. ACM, 2004.

[15] L. Brown and L. Gruenwald. Tree-based indexes for image data. *J. Visual Communication and Image Representation*, 9(4):300–313, 1998.

[16] C. Brzuska and A. Mittelbach. Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In *ASIACRYPT (2)*, volume 8874 of *Lecture Notes in Computer Science*, pages 142–161. Springer, 2014.

[17] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1997.

[18] R. Canetti and R. R. Dakdouk. Obfuscating point functions with multibit output. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 489–508. Springer, 2008.

[19] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. D. Smith. Reusable fuzzy extractors for low-entropy distributions. In *EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 117–146. Springer, 2016.

[20] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *ACM Conference on Computer and Communications Security*, pages 668–679. ACM, 2015.

[21] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *NDSS*. The Internet Society, 2014.

[22] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2010.

[23] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage. In *FSE*, volume 9783 of *Lecture Notes in Computer Science*, pages 474–493. Springer, 2016.

[24] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *ACM Conference on Computer and Communications Security*, pages 79–88. ACM, 2006.

[25] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, pages 253–262. ACM, 2004.

[26] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[27] W. Dong, Z. Wang, M. Charikar, and K. Li. High-confidence near-duplicate image detection. In *ICMR*, page 1. ACM, 2012.

[28] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *ICDCS*, pages 617–624, 2002.

[29] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613. ACM, 1998.

[30] A. Juels and M. Sudan. A fuzzy vault scheme. *Des. Codes Cryptography*, 38(2):237–257, 2006.

[31] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In *ACM Conference on Computer and Communications Security*, pages 965–976. ACM, 2012.

[32] K. Kim, H. Nam, V. K. Singh, D. Song, and H. Schulzrinne. DYSWIS: crowdsourcing a home network diagnosis. In *ICCCN*, pages 1–10. IEEE, 2014.

[33] M. Kuzu, M. S. Islam, and M. Kantarcioglu. Efficient similarity search over encrypted data. In *ICDE*, pages 1156–1167. IEEE Computer Society, 2012.

[34] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–54. Springer, 2000.

[35] S. Matwin. Privacy-preserving data mining techniques: Survey and challenges. In *Discrimination and Privacy in the Information Society*, volume 3 of *Studies in Applied Philosophy, Epistemology and Rational Ethics*, pages 209–221. Springer, 2013.

[36] A. Narayanan and V. Shmatikov. Obfuscated databases and group privacy. In *ACM Conference on Computer and Communications Security*, pages 102–111. ACM, 2005.

[37] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In *ACM Conference on Computer and Communications Security*, pages 644–655. ACM, 2015.

[38] O. Pandey and Y. Rouselakis. Property preserving symmetric encryption. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 375–391. Springer, 2012.

[39] A. Qadir, J. Neubert, and W. Semke. On-board visual tracking with unmanned aircraft system (UAS). *CoRR*, abs/1203.2386, 2012.

[40] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-neighbor methods in learning and vision: theory and practice (neural information processing)*. The MIT press, 2006.

[41] V. K. Singh, H. Schulzrinne, and K. Miao. DYSWIS: an architecture for automated diagnosis of networks. In *NOMS*, pages 851–854. IEEE, 2008.

[42] S. P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1):43–77, 2004.

A Proof of Theorem 4.1

We analyze in turn, each of the four aspects that comprise correctness. We start with correctness of decryption. Let $I, I' \in \mathcal{D}$ arbitrary such that $\mathrm{Cl}(I, I') = \texttt{close}$ and let M be some message associated to I. We lowerbound the probability that

$$\Pr[P \leftarrow \mathsf{Init}; C \leftarrow \mathsf{Mask}(P, I, M) \ : \mathsf{Unmask}(P, I', C) = M] \ .$$

Notice that the decryption succeeds if for some i the values of $g_i(I)$ and $g_i(I')$ coincide, and for any i for which the values of $g_i(I)$ and $g_i(I')$ are different, the values of $G(g_i(I))$ and $G(g_i(I'))$ do not collide. If we write $\mathsf{coll}(i)$ for the event that $g_i(I) = g_i(I')$ and $\mathsf{tagcoll}$ for the event that $G(g_i(I)) = G(g_i(I'))$ then we can lowerbound the above probability by

$$\Pr[(\exists i \in |L|)\ \mathsf{coll}(i) \wedge ((\forall i \in |L|)\ (\overline{\mathsf{coll}(i)} \implies \overline{\mathsf{tagcoll}(i)})] \ ,$$

which ensure that if for some i $G(g_i(I)) = G(g_i(I'))$ then the collision holds because I and I' collide under g_i and not because a collision in G. We can lowerbound the above probability by:

$$\begin{aligned} & 1 - \Big(\Pr[(\forall i \in |L|)\ \overline{\mathsf{coll}(i)} \vee (\exists i \in |L|)\ \overline{\mathsf{coll}(i)} \wedge \mathsf{tagcoll}(i)]]\Big) \\ & \geq \Pr\left[\ (\exists i \in |L|)\ \mathsf{coll}(i)\ \right] - \sum_{i \in [|L|]} \Pr\left[\ \overline{\mathsf{coll}(i)} \wedge \mathsf{tagcoll}(i))\ \right] \geq P_1 - \frac{L}{2^h} \ . \end{aligned}$$

Above, by the assumption that $\texttt{close}(I, I') = \texttt{close}$ we have that $\Pr\left[\ (\exists i \in |L|)\ \mathsf{coll}(i)\ \right] \geq P_1$. Furthermore, for any i we have that $\Pr\left[\ ((\exists i \in |L|)\ \mathsf{coll}(i) \wedge \mathsf{tagcoll}(i))\ \right] = \Pr\left[\ \overline{\mathsf{coll}(i)}\ \right] \cdot \Pr\left[\ \mathsf{tagcoll}(i) \mid \overline{\mathsf{coll}(i)}\ \right] \leq \frac{1}{2^h}$. We conclude that the the probability that the decryption succeeds is lowerbounded by $P_1 - \frac{L}{2^h}$.

Next, we bound the probability that for $\mathrm{Cl}(I, I') = \texttt{far}$ a decryption under I would succeed with I'. Notice that decryption is not successful if for any i it holds that $\mathbf{T}[i] \neq \mathbf{T}'[i]$; we lowerbound the probability of this event, and for each

$$\begin{aligned} & \Pr\left[\ (\forall i \in |L|)\ \overline{\mathsf{tagcoll}(i)}\ \right] \geq 1 - \sum_{i \in [|L|]} \Pr\left[\ \mathsf{tagcoll}(i)\ \right] \geq 1 - \sum_{i \in [|L|]} \Pr\left[\ \mathsf{coll}(i) \vee (\overline{\mathsf{coll}(i)} \wedge \mathsf{tagcoll}(i))\ \right] \\ \geq \ & 1 - \sum_{i \in [|L|]} \Big(\Pr\left[\ \mathsf{coll}(i) \vee (\overline{\mathsf{coll}(i)} \wedge \mathsf{tagcoll}(i))\ \right]\Big) \geq 1 - \sum_{i \in [|L|]} \Big(\Pr\left[\ \mathsf{coll}(i)\ \right] + \Pr\left[\ \mathsf{tagcoll}(i)) \mid \overline{\mathsf{coll}(i)}\ \right]\Big) \\ \geq \ & 1 - (P_2 + \frac{|L|}{2^h}) \ . \end{aligned}$$

For the correctness of the fuzzy search procedure we note that if $(C, Tags) \in \mathbf{C}$ for some $(C, Tags) \leftarrow \mathsf{Mask}(P, I, M)$ and $I' \in \mathcal{D}$ then $C \in \mathsf{FuzzyS}(\mathbf{C}, \texttt{DS}, \mathsf{Query}(I'))$ if and only if $Tags$ and $\mathsf{Query}(I')$ overlap on at least one position. Compatibility between $(\mathcal{D}, \texttt{close})$ and $\mathcal{H}$ ensures that these collisions happen with probability at least P_1 if $\mathrm{Cl}(I, I') = \mathrm{Cl}$ and at most P_2 if $\mathrm{Cl}(I, I') = \texttt{far}$.

Since $p_1 > p_2$, by increasing k and L (L much faster than k) we can make P_1 very close to 1 and P_2 very close to 0. For some fixed parameters $L, k \in \mathbb{N}$, we refer to the above extension of an LSH function family as an (L, k)-eLSH function family.

B Proof of Theorem 4.2

Proof idea. We prove the theorem by exhibiting a simulator which (roughly) works as follows. The simulator (who does not have access to $\mathbf{M}$) creates a "fake" database where each individual ciphertext is an encryption of $\mathbf{0}$ under a fresh randomly chosen key. The tags associated to each ciphertext are

uniformly generated, but subject to the constraints that the overlap between tags is consistent with the leak information that the simulator is given as input. The simulator runs the adversary A on this fake database and outputs whatever that adversary outputs.

The argument that the view of the adversary in the real game is close to the one provided by the simulator follows a standard "game-hopping" technique [9] where we incrementally change how the view of the adversary is computed. In the first hop, we change how the tags are computed by selecting them at random, as explained above. Here we rely on the unpredictability of the entries in $\mathcal{H}(\mathcal{M})$ to argue that the probability that the adversary notices the change is upper bounded by a collision in the random oracle calls. In the next step we first replace each of the keys used with keys selected uniformly at random (without respecting the overlapping patterns of the real keys) and the real messages that are encrypted with some fixed message. The key-hiding and plaintext-privcacy property of the encryption scheme guarantees that the adversary does not notice the difference. Finally, in the next step, instead of encrypting real messages from $\mathbf{M}$ we encrypt some fixed-length message.

The overall idea behind the simulator that we exhibit is as follows. Recall that in the basic scheme, for each entry $(I, M) = (\mathbf{I}[i], \mathbf{M}[i]) \in \mathbf{D}$ a ciphertext is formed from a "parallel" encryption of M under keys of the form $H(g_i(I))$ (with g_i the components of the extended LSH defined by the parameters of the scheme) together with a set of tags, each of the form $\langle i, G(g_i(I))\rangle$:

$$\mathcal{E}(H(g_1(I)), M)|| \ldots ||\mathcal{E}(H(g_L(I)), M)||\{\langle 1, G(g_1(\mathbf{I}[i]))\rangle, \langle l, \ldots, G(g_L(\mathbf{I}[i]))\rangle\} \ .$$

For each query Q the adversary obtains the set of tags $\{G(g_1(Q)), \ldots, G(g_L(Q))\}$.

The leakage function only provides the simulator with information regarding the overlap between the different sets of tags (belonging to ciphertext and/or queries), but nothing else – recall that we have assumed that the lengths of $\mathbf{I}, \mathbf{Q}$ and the size of entries in $\mathbf{M}$ are all public.

The idea for the simulator construction is to run the adversary A on an input where tags and ciphertexts are fake: since (as we will argue) the actual values of the tags are unpredictable the simulator will selects fresh tags in a way consistent with the information about their overlap. Similarly, encryptions will be encryptions of some fixed string, say $\mathbf{0}$, each encryption using a fresh key. A fake ciphertext would then be of the form

$$\mathcal{E}(K_1, \mathbf{0})|| \ldots ||\mathcal{E}(K_L, \mathbf{0})||\{\langle 1, T_1\rangle, \langle 2, T_2\rangle, \ldots, \langle L, T_L\rangle\}.$$

with keys K_i distinct (even across different entries in the encrypted database) and with the tags satisfying the different equalities captured by the function $\mathbf{L}$.

Since, the values of $g_i(I)$ and $g_i(Q)$ (for $I \in \mathbf{I}, Q \in \mathbf{Q}$) are unpredictable, even given $\mathbf{L}(\mathbf{I}, \mathbf{Q})$ (since $\widetilde{H}_\infty(\mathcal{H}(\mathcal{M}) \mid \mathbf{L}(\mathcal{M})) \geq l$), the only way for the adversary to observe that the values corresponding to the random oracles have been replaced by random strings is to query one of these values to the random oracle. By the assumption, for each individual query to the random oracle(s), the likelihood of a collision is $\frac{1}{2^l}$ and a union bound gives an overall distinguishing advantage at most $\frac{q_{G,H}}{2^l}$. Since per the discussion above the keys used for encryption are unpredictable, the adversary will notice a difference only if it breaks the security of the underlying symmetric encryption scheme.

Recall that we have defined the leaking function for the scheme as: $\mathsf{leak}(P, \mathbf{I}, \mathbf{Q})$ as the map $\mathbf{L} : [n] \to \mathcal{P}([|\mathbf{I}| + |\mathbf{Q}|]) \times [L]$ defined by

$$\mathbf{L}(t) = \{(i, u) \mid \langle u, T[t]\rangle \in \mathbf{I}[i]\} \cup \{\langle j + |\mathbf{I}|, u\rangle \mid \langle u, T[t]\rangle \in \mathbf{Q}[j]\} \ .$$

To describe the simulator it is more convenient to work with the "converse" function which associates to each $\mathbf{I}[i]$ and $\mathbf{Q}[j]$ the set of indexes of tags. By abuse of notation we write $\mathbf{L}^{-1}$ for this function. $\mathbf{L}^{-1} : [|\mathbf{I}| + |\mathbf{Q}|] \times [L] \to \mathcal{P}([n])$ defined by $\mathbf{L}^{-1}((i, u)) = t$ iff $\mathbf{L}(t) = (i, u)$.

The simulator is described in Figure 6. Its preamble samples values for the keys and the tags that could occur in the system: there are at most L distinct key used for each entry in $\mathbf{I}$ and there are at

Algorithm $S(P, \mathbf{L})$

For $(i, l) \in [|\mathbf{I}|] \times [L]$ do
  set $K_{i,l} \xleftarrow{\$} \{0,1\}^h$
For $1 \le i \le n$
  select $T[i] \xleftarrow{\$} \{0,1\}^h$
For $i = 1, \ldots, |\mathbf{I}|$ do
  For $l = 1 \ldots L$ do
    $\mathbf{C}[i][l] \xleftarrow{\$} \mathsf{Mask}(K_{i,l}, \mathbf{0})$
    $\mathbf{Tags}[i] \leftarrow \mathbf{Tags}[i] \cup \{\langle l, T[\mathbf{L}^{-1}(i,l)]\rangle\}$
  $\mathbf{E}[i] \leftarrow \mathbf{C}[i] || \mathbf{Tags}[i]$
For $j = 1 \ldots |\mathbf{Q}|$ do
  For $l = 1 \ldots L$ do
  $\mathbf{T}[j] \leftarrow \mathbf{T}[j] \cup \{\langle l, T[\mathbf{L}^{-1}(j + |\mathbf{I}|, l)]\rangle\}$
$\mathtt{info} \xleftarrow{\$} A(P, \mathbf{E}, \mathbf{T})$

Figure 6: Simulator for the security of the Basic KlFS scheme.

$\mathbf{Exp}^0_{\Pi,\mathcal{M}}(A)$	$\mathbf{Exp}^1_{\Pi,\mathcal{M}}(A)$
For $i = 1, \ldots, L$ do	For $i = 1, \ldots, L$ do
For $j = 1, \ldots, k$ do	For $j = 1, \ldots, k$ do
$h_{i,j}(.) \xleftarrow{\$} \mathcal{H}$, $g_i[j] \leftarrow h_{i,j}(.)$	$h_{i,j}(.) \xleftarrow{\$} \mathcal{H}$, $g_i[j] \leftarrow h_{i,j}(.)$
$\mathbf{g} = (g_1(\cdot), g_2(\cdot), \ldots, g_L(\cdot))$	$\mathbf{g} = (g_1(\cdot), g_2(\cdot), \ldots, g_L(\cdot))$
$P \leftarrow k \| L \| \mathbf{g}$	$P \leftarrow k \| L \| \mathbf{g}$
$(\mathbf{I}, \mathbf{M}, \mathbf{Q}, \mathtt{target}) \xleftarrow{\$} \mathcal{M}$	$(\mathbf{I}, \mathbf{M}, \mathbf{Q}, \mathtt{target}) \xleftarrow{\$} \mathcal{M}$
For $j = 1, \ldots, \|\mathbf{I}\|$ do $\mathbf{Tags} \leftarrow \emptyset$	For $j = 1, \ldots, \|\mathbf{I}\|$ do $\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do	For $i = 1, \ldots, L$ do
$\mathbf{C}[i] \xleftarrow{\$} \mathcal{E}(H(g_i(\mathbf{I}[j])), \mathbf{M}[j])$	$\mathbf{C}[i] \xleftarrow{\$} \mathcal{E}(H(g_i(\mathbf{I}[j])), \mathbf{M}[j])$
$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{I}[j]))\rangle\}$	$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{I}[j]))\rangle\}$
$\mathbf{E}[j] \leftarrow \mathbf{C} \| \mathbf{Tags}$	$\mathbf{E}[j] \leftarrow \mathbf{C} \| \mathbf{Tags}$
For $j = 1, \ldots, \|\mathbf{Q}\|$ do $\mathbf{Tags} \leftarrow \emptyset$	For $j = 1, \ldots, \|\mathbf{Q}\|$ do $\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do	For $i = 1, \ldots, L$ do
$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{Q}[j]))\rangle\}$	$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{Q}[j]))\rangle\}$
$\mathbf{T}[j] \leftarrow \mathbf{Tags}$	$\mathbf{T}[j] \leftarrow \mathbf{Tags}$
$\mathtt{info} \xleftarrow{\$} A^{G,H}(P, \mathbf{E}, \mathbf{T})$	$\mathtt{info} \xleftarrow{\$} A^{G,H}(P, \mathbf{E}, \mathbf{T})$
Return $\mathtt{info} \stackrel{?}{=} \mathtt{target}$	If **bad** then Return 0
	Return $\mathtt{info} \stackrel{?}{=} \mathtt{target}$

Figure 7: In $\mathbf{Exp}^1_{\Pi,\mathcal{M}}$ event **bad** is set to true if for some i and j, adversary A queries H on a value $g_i(\mathbf{I}[j])$ or queries G with $g_i(\mathbf{I}[j])$ or $g_i(\mathbf{Q}[j])$ calculated elsewhere in the experiment.

most n tags (see above). Then, for each entry i in $\mathbf{I}$ it creates a fake ciphertext: it encrypts $\mathbf{0}$ under each of the different L keys $K_{i,l}$ and associates a set of tags, one for each tag identity leaked in $\mathbf{L}[i]$.

For our analysis it is convenient to work with stronger notions of security for the underlying $\mathcal{SE}$ scheme, both in terms of plaintext privacy as well as key-hiding property. For plaintext privacy we consider a multi-key, multi-message setting where one considers an adversary who repeatedly issues pairs of messages to a set of encryption oracles, each oracle keyed with an independently generated key. The oracles (consistently) return encryptions of the left or that of the right message and the goal

$\mathbf{Exp}^{2}_{\Pi,\mathcal{M}}(A)$	$\mathbf{Exp}^{3}_{\Pi,\mathcal{M}}(A)$
For $i = 1, \ldots, L$ do	For $i = 1, \ldots, L$ do
For $j = 1, \ldots, k$ do	For $j = 1, \ldots, k$ do
$h_{i,j}(.) \xleftarrow{\$} \mathcal{H}$, $g_i[j] \leftarrow h_{i,j}(.)$	$h_{i,j}(.) \xleftarrow{\$} \mathcal{H}$, $g_i[j] \leftarrow h_{i,j}(.)$
$\mathbf{g} = (g_1(\cdot), g_2(\cdot), \ldots, g_L(\cdot))$	$\mathbf{g} = (g_1(\cdot), g_2(\cdot), \ldots, g_L(\cdot))$
$P \leftarrow k \| L \| \mathbf{g}$	$P \leftarrow k \| L \| \mathbf{g}$
$(\mathbf{I}, \mathbf{M}, \mathbf{Q}, \mathtt{target}) \xleftarrow{\$} \mathcal{M}$	$(\mathbf{I}, \mathbf{M}, \mathbf{Q}, \mathtt{target}) \xleftarrow{\$} \mathcal{M}$
	For $j = 1 \ldots \lvert\mathbf{I}\rvert$ do
	For $i = 1 \ldots L$ do
	If $g_i(\mathbf{I}[j]) = g_{i_0}(\mathbf{I}[j_0])$ for some $(i_0, j_0) \leq (i, j)$
	Then $K_{i,j} \leftarrow K_{i_0,j_0}$
	Else $K_{i,j} \leftarrow \{0,1\}^{\lambda}$
For $j = 1, \ldots, \lvert\mathbf{I}\rvert$ do $\mathbf{Tags} \leftarrow \emptyset$	For $j = 1, \ldots, \lvert\mathbf{I}\rvert$ do $\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do	For $i = 1, \ldots, L$ do
$\mathbf{C}[i] \xleftarrow{\$} \mathcal{E}(H(g_i(\mathbf{I}[j])), \mathbf{M}[j])$	$\mathbf{C}[i] \xleftarrow{\$} \mathcal{E}(K_{i,j}, \mathbf{M}[j])$
$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{I}[j]))\rangle\}$	$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{I}[j]))\rangle\}$
$\mathbf{E}[j] \leftarrow \mathbf{C} \| \mathbf{Tags}$	$\mathbf{E}[j] \leftarrow \mathbf{C} \| \mathbf{Tags}$
For $j = 1, \ldots, \lvert\mathbf{Q}\rvert$ do $\mathbf{Tags} \leftarrow \emptyset$	For $j = 1, \ldots, \lvert\mathbf{Q}\rvert$ do $\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do	For $i = 1, \ldots, L$ do
$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{Q}[j]))\rangle\}$	$\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{Q}[j]))\rangle\}$
$\mathbf{T}[j] \leftarrow \mathbf{Tags}$	$\mathbf{T}[j] \leftarrow \mathbf{Tags}$
For $i = 1$ to n do $T[i] \leftarrow \{0,1\}^{l}$	For $i = 1$ to n do $T[i] \leftarrow \{0,1\}^{l}$
For $j = 1, \ldots, \lvert\mathbf{I}\rvert$ do	For $j = 1, \ldots, \lvert\mathbf{I}\rvert$ do
$\mathbf{C} \| \mathbf{Tags} \leftarrow \mathbf{E}[j]$	$\mathbf{C} \| \mathbf{Tags} \leftarrow \mathbf{E}[j]$
$\mathbf{E}[j] \leftarrow \mathbf{C} \| \{\langle l, T[\mathbf{L}^{-1}(j, l)\rangle] \mid l \in [L]\}$	$\mathbf{E}[j] \leftarrow \mathbf{C} \| \{\langle l, T[\mathbf{L}^{-1}(j, l)\rangle] \mid l \in [L]\}$
For $j = 1, \ldots, \lvert\mathbf{Q}\rvert$ do	For $j = 1, \ldots, \lvert\mathbf{Q}\rvert$ do
$\mathbf{T}[j] \leftarrow \{\langle l, T[\mathbf{L}^{-1}(l, \lvert\mathbf{I}\rvert + j)] \mid l \in [L]\}$	$\mathbf{T}[j] \leftarrow \{\langle l, T[\mathbf{L}^{-1}(l, \lvert\mathbf{I}\rvert + j)] \mid l \in [L]\}$
$\mathtt{info} \xleftarrow{\$} A^{G,H}(P, \mathbf{E}, \mathbf{T})$	$\mathtt{info} \xleftarrow{\$} A^{G,H}(P, \mathbf{E}, \mathbf{T})$
If **bad** then Return 0	If **bad** then Return 0
Return $\mathtt{info} \stackrel{?}{=} \mathtt{target}$	Return $\mathtt{info} \stackrel{?}{=} \mathtt{target}$

Figure 8: In $\mathbf{Exp}^{2}_{\Pi,\mathcal{M}}$ proceeds as $\mathbf{Exp}^{1}_{\Pi,\mathcal{M}}$ – the tags that occur in ciphertexts and queries are replaced with independently selected random strings, but to preserving the overlap between tags, as specified by the leakage function $\mathbf{L}$ which is evaluated given the real tags. $\mathbf{Exp}^{3}_{\Pi,\mathcal{M}}$ is identical with $\mathbf{Exp}^{2}_{\Pi,\mathcal{M}}$ except that symmetric encryption keys are selected uniformly and independently at random, subject to the same equality patterns as when computed within oracle H.

of the adversary is to guess which message the oracles encrypt. This notion is equivalent with one where the adversary only has access to a single encryption oracle via a reduction that incurs a loss proportional to the number of encryption oracles.

Similarly, as far as key-hiding is concern we use a stronger definition that considers an adversary who specifies the number of ciphertexts it wants to be challenged on, and a "pattern" specifying which of the ciphertexts should use the same key. The adversary then needs to distinguish between a set of ciphertexts where the keys follow the specified pattern from another where each individual encryption key is generated independently at random. A standard argument shows that security in this sense can

$\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}(A)$

For $i = 1, \ldots, L$ do
  For $j = 1, \ldots, k$ do
    $h_{i,j}(.) \xleftarrow{\$} \mathcal{H}$, $g_i[j] \leftarrow h_{i,j}(.)$
$\mathbf{g} = (g_1(\cdot), g_2(\cdot), \ldots, g_L(\cdot))$
$P \leftarrow k \| L \| \mathbf{g}$
$(\mathbf{I}, \mathbf{M}, \mathbf{Q}, \mathtt{target}) \xleftarrow{\$} \mathcal{M}$

For $j = 1 \ldots |\mathbf{I}|$ do
  For $i = 1 \ldots L$ do
    Else $K_{i,j} \leftarrow \{0,1\}^{\lambda}$

For $j = 1, \ldots, |\mathbf{I}|$ do $\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do
  $\mathbf{C}[i] \xleftarrow{\$} \mathcal{E}(K_{i,j}, \mathbf{M}[j])$
  $\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{I}[j]))\rangle\}$
$\mathbf{E}[j] \leftarrow \mathbf{C} \| \mathbf{Tags}$

For $j = 1, \ldots, |\mathbf{Q}|$ do $\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do
  $\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{Q}[j]))\rangle\}$
$\mathbf{T}[j] \leftarrow \mathbf{Tags}$

For $i = 1$ to n do $T[i] \leftarrow \{0,1\}^{l}$

For $j = 1, \ldots, |\mathbf{I}|$ do
$\mathbf{C} \| \mathbf{Tags} \leftarrow \mathbf{E}[j]$
$\mathbf{E}[j] \leftarrow \mathbf{C} \| \{\langle l, T[\mathbf{L}^{-1}(j,l)\rangle] \mid l \in [L]\}$

For $j = 1, \ldots, |\mathbf{Q}|$ do
$\mathbf{T}[j] \leftarrow \{\langle l, T[\mathbf{L}^{-1}(l, |\mathbf{I}| + j)] \mid l \in [L]\}$

$\mathtt{info} \xleftarrow{\$} A^{G,H}(P, \mathbf{E}, \mathbf{T})$
If **bad** then Return 0
Return $\mathtt{info} \stackrel{?}{=} \mathtt{target}$

$\mathbf{Exp}^{5}_{\Pi,\mathcal{M}}(A)$

For $i = 1, \ldots, L$ do
  For $j = 1, \ldots, k$ do
    $h_{i,j}(.) \xleftarrow{\$} \mathcal{H}$, $g_i[j] \leftarrow h_{i,j}(.)$
$\mathbf{g} = (g_1(\cdot), g_2(\cdot), \ldots, g_L(\cdot))$
$P \leftarrow k \| L \| \mathbf{g}$
$(\mathbf{I}, \mathbf{M}, \mathbf{Q}, \mathtt{target}) \xleftarrow{\$} \mathcal{M}$

For $j = 1 \ldots |\mathbf{I}|$ do
  For $i = 1 \ldots L$ do
    Else $K_{i,j} \leftarrow \{0,1\}^{\lambda}$

For $j = 1, \ldots, |\mathbf{I}|$ do $\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do
  $\mathbf{C}[i] \xleftarrow{\$} \mathcal{E}(K_{i,j}, \mathbf{0})$
  $\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{I}[j]))\rangle\}$
$\mathbf{E}[j] \leftarrow \mathbf{C} \| \mathbf{Tags}$

For $j = 1, \ldots, |\mathbf{Q}|$ do $\mathbf{Tags} \leftarrow \emptyset$
For $i = 1, \ldots, L$ do
  $\mathbf{Tags} \leftarrow \mathbf{Tags} \cup \{\langle i, G(g_i(\mathbf{Q}[j]))\rangle\}$
$\mathbf{T}[j] \leftarrow \mathbf{Tags}$

For $i = 1$ to n do $T[i] \leftarrow \{0,1\}^{l}$

For $j = 1, \ldots, |\mathbf{I}|$ do
$\mathbf{C} \| \mathbf{Tags} \leftarrow \mathbf{E}[j]$
$\mathbf{E}[j] \leftarrow \mathbf{C} \| \{\langle l, T[\mathbf{L}^{-1}(j,l)\rangle] \mid l \in [L]\}$

For $j = 1, \ldots, |\mathbf{Q}|$ do
$\mathbf{T}[j] \leftarrow \{\langle l, T[\mathbf{L}^{-1}(l, |\mathbf{I}| + j)] \mid l \in [L]\}$

$\mathtt{info} \xleftarrow{\$} A^{G,H}(P, \mathbf{E}, \mathbf{T})$
If **bad** then Return 0
Return $\mathtt{info} \stackrel{?}{=} \mathtt{target}$

Figure 9: In $\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}$ keys $K_{i,j}$ are selected independently at random and in $\mathbf{Exp}^{5}_{\Pi,\mathcal{M}}$ each $\mathbf{C}[i]$ is an encryption of $\mathbf{0}$ rather than $\mathbf{M}[\mathbf{i}]$

be reduced to security when the adversary is only allowed to see two ciphertexts via a reduction that incurs a loss proportional to the number of ciphertexts the adversary is allowed to request. Below, the advantage functions for IND-CPA and key-hiding security refer to security in the sense defined here.

To analyze the distance between the output distribution of the real experiment with A and of the ideal one with S we proceed as follows. Below we write $\mathbf{Exp}^{0}_{\Pi,\mathcal{M}}(A)$ for $\mathbf{Exp}^{\mathtt{prv\text{-}real}}_{\Pi,\mathcal{M}}(A)$. First, we consider a modified variant of the experiment $\mathbf{Exp}^{0}_{\Pi,\mathcal{M}}(A)$ where the experiment returns 0 if the adversary queries the random oracles G, H on values calculated elsewhere in the experiment; otherwise it behaves as usual. The definitions of $\mathbf{Exp}^{0}_{\Pi,\mathcal{M}}(A)$ and $\mathbf{Exp}^{1}_{\Pi,\mathcal{M}}(A)$ are in Figure 7.

Notice that these are queries of the form $g_i(\mathbf{I}[j])$ sent to oracle H (for some i, j) and queries of the form $g_i(\mathbf{I}[j])$ or $g_i(\mathbf{Q}[j])$ sent to oracle G. Since by assumption these values have min-entropy l (even against an adversary who has the information $\mathbf{L}(\mathbf{I}, \mathbf{Q})$ – which is all the information passed as input

to the simulator), and since there are $L \cdot (|\mathbf{I}| + |\mathbf{Q}|)$ tags (since there are L tags associated to each ciphertext and each query) we have that

$$\Pr\left[\mathbf{Exp}^{0}_{\Pi,\mathcal{M}}(A) = 1\right] - \Pr\left[\mathbf{Exp}^{1}_{\Pi,\mathcal{M}}(A) = 1\right] \leq \frac{L \cdot (|\mathbf{I}| + |\mathbf{Q}|) \cdot (q_H + q_G)}{2^l} .$$

In the next step, we consider experiment $\mathbf{Exp}^{2}_{\Pi,\mathcal{M}}(A)$ (formally described in Figure 8), where we replace each of the tags calculated via calls to G by independently selected strings of appropriate length (subject to the condition that equality patterns between strings are preserved). Subject to event **bad** not being raised the distribution of the tags are identical in $\mathbf{Exp}^{1}_{\Pi,\mathcal{M}}(A)$ and $\mathbf{Exp}^{2}_{\Pi,\mathcal{M}}(A)$ and therefore

$$\Pr\left[\mathbf{Exp}^{1}_{\Pi,\mathcal{M}}(A) = 1\right] = \Pr\left[\mathbf{Exp}^{2}_{\Pi,\mathcal{M}}(A) = 1\right] . \tag{1}$$

In the next step we replace encryption keys calculated as $H(g_i(\mathbf{I}[j]))$ by independently selected keys, subject to preserving equality between keys. The resulting experiment, which we call $\mathbf{Exp}^{3}_{\Pi,\mathcal{M}}(A)$ is in Figure 8. As above, if event **bad** is not raised (so the adversary did not issue any query $g_i(\mathbf{I}[j])$ to H), the distribution of keys is identical in the two experiments, hence:

$$\Pr\left[\mathbf{Exp}^{2}_{\Pi,\mathcal{M}}(A) = 1\right] = \Pr\left[\mathbf{Exp}^{3}_{\Pi,\mathcal{M}}(A) = 1\right] . \tag{2}$$

Next we consider experiment $\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}(A)$ obtained by changing the way the encryption keys are generated. Its definition is in Figure 9. Instead of selecting them in a way that preserves the overlaps, we simply select each of them uniformly at random. The idea behind this transformation is that the adversary should not observe the change of keys, if the encryption scheme $\mathcal{SE}$ is key-hiding. Technically, we construct an adversary B against this property as follows: the adversary simulates all of the execution of $\mathbf{Exp}^{3}_{\Pi,\mathcal{M}}(A)$, except that the encryptions of the messages in $\mathbf{M}$ are created using the abilities that B when attacking key hiding: the adversary submits a key-pattern expressing the desired key equality (essentially those corresponding to equal keys in the real execution).

If the keys used by the oracle follow the requested pattern then the view of A is as in $\mathbf{Exp}^{3}_{\Pi,\mathcal{M}}(A)$; otherwise (i.e. the keys are each selected independently at random) the view of A is as in $\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}(A)$. Therefore B can break the key-hiding security of $\mathcal{SE}$ with the same probability that the outputs of $\mathbf{Exp}^{3}_{\Pi,\mathcal{M}}(A)$ and $\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}(A)$ are distinguished by D.

$$\Pr\left[\mathbf{Exp}^{3}_{\Pi,\mathcal{M}}(A) = 1\right] - \Pr\left[\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}(A) = 1\right] \leq \mathbf{Adv}^{kh}_{\mathcal{SE}}(B) .$$

In the next step we modify $\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}(A)$: for each ciphertext, instead of encrypting $\mathbf{M}[i]$ using $\mathcal{SE}$ the experiment $\mathbf{Exp}^{5}_{\Pi,\mathcal{M}}(A)$ encrypts $\mathbf{0}$. The definition of $\mathbf{Exp}^{5}_{\Pi,\mathcal{M}}(A)$ is in Figure 9. Recall that we have assumed that the length of entries in $\mathbf{M}$ is fixed and known – $\mathbf{0}$ is simply a message of this length. In the variation where the length of entries in $\mathbf{M}$ varies and is part of the output of the leak function we would encrypt some fixed message of the appropriate length. Since in both experiments the encryption keys used by Mask are selected at random and the only difference are the messages that these keys encrypt, if distance between the output distributions of $\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}$ and $\mathbf{Exp}^{5}_{\Pi,\mathcal{M}}$ are different we can use them to break the $\mathsf{IND\text{-}CPA}$ security of the underlying encryption scheme. That is, we construct an adversary C such that:

$$\Pr\left[\mathbf{Exp}^{4}_{\Pi,\mathcal{M}}(A) = 1\right] - \Pr\left[\mathbf{Exp}^{5}_{\Pi,\mathcal{M}}(A) = 1\right] \leq \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{SE}}(C) .$$

Finally we note that the output of $\mathbf{Exp}^{5}_{\Pi,\mathcal{M}}(A)$ is distributed identically with the output of $\mathbf{Exp}^{\mathtt{prv\text{-}ideal}}_{\Pi,\mathcal{M}}(S)$.

C Proof of Theorem 5.1

The proof idea is the same as before: we construct a simulator that runs the adversary A on a fake encrypted database, and argue that the adversary A does not distinguish between a real database and the fake one.

Just like for the basic scheme, we use access data to generate cryptographic keys (as $Hg_k((V_i(\mathbf{I}[j])))$ and tags (as $G(V_i(\mathbf{I}[j])))$as ; the main difference is that we use the keys to encrypt shares of an encryption key, which in turn is used to protect the data. Notice that security of the construction does not rely on the security of the secret sharing scheme, since our assumption is that each individual $g_k(V_i(\mathbf{I}[j]))$ is unpredictable (given the inherent leakage). In particular, this implies that each of the keys used to encrypt the key shares is unpredictable, so an adversary obtains no information even about a single share. The secret sharing scheme is used however to enforce that only decrypting with a close image (i.e. an image with sufficiently many overlapping features) succeeds.

The unpredictability assumption allows the simulator to produce fake ciphertexts based on the information it learns from leakage function: For each entry $\mathbf{I}[i]$ and for each feature vector associated to $\mathbf{I}[i]$, the simulator associates a collection of fake tags. These tags are selected independently at random in such a way that the overlap of tags across features (and images) is consistent with the information leaked by scheme. The symmetric encryption ciphertext uses a fresh random key, and encrypts $\mathbf{0}$. To construct the encrypted shares, the simulator runs the secret sharing algorithm on a key K' unrelated to K, and encrypts each of the resulting shares under a fresh encryption key. The argument that this simulation is indistinguishable from the real encrypted database proceeds through a similar sequence of game-hops.

In more detail, we start with the real execution and change it incrementally. First we change how the tags and keys are calculated: for each of the n possible distinct tags and for each possible distinct key we select a a random string of length h. Due to entropy of the individual values $V_i(\mathbf{I}[j])$ and $V_i(\mathbf{Q}[j])$ used to derive the real tags and keys, the adversary will only see a difference with probability $L \cdot f \cdot (|\mathbf{I}| + |\mathbf{Q}|) \cdot (q_G + q_H)/2^l$: there are f possible features associated to each element of $\mathbf{I}$ and $\mathbf{Q}$ and for each feature vector V_i and for each function g_j, the value $g_j(V_i)$ is queried to G. Oracle H is queried with $f \cdot L$ such values. Per our assumption, all of these values have minentropy l and therefore an adversary could query the oracle on one of these values with probability:

$$\frac{f \cdot L \cdot (|\mathbf{I}| + |\mathbf{Q}|) \cdot q_G}{2^l} + \frac{f \cdot L \cdot |\mathbf{I}| \cdot q_H}{2^l} \leq \frac{f \cdot L \cdot (|\mathbf{I}| + |\mathbf{Q}|) \cdot (q_G + q_H)}{2^l} .$$

In the above experiment, the keys used to encrypt shares still respect the overlap pattern that the leakage function reveals. Next, we replace each of the keys with fresh ones (that do not respect the pattern anymore). Any change in the output distribution of the adversary is then due to the adversary observing this change. Formally, we build an adversary B against the key-hiding property of the $\mathcal{SE}$ scheme, as described in the case of the basic scheme. The adversary simulates all of the execution except when creating ciphertexts under these keys: for these ciphertexts the adversary uses the key-hidding game (to which it specifies the overlapping pattern between keys).

In the next hop, we replace each encrypted key share with some fixed string of the same length. The intuition is that since each of the keys used to encrypt the shares is unknown to the adversary the only way for the adversary to notice a difference is to contradict IND-CPA security of the scheme. Specifically, we can construct an adversary C against IND-CPA security of the $\mathcal{SE}$ scheme. The adversary simulates the execution of the experiment perfectly except that whenever it needs to encrypt some share, the adversary uses a left-right encryption oracle to obtain either an encryption of the share or the encryption of a fixed message. The adversary uses a different oracle for each such encryption.

In the last change we replace the encryptions of the messages in $\mathbf{M}$ with encryptions of $\mathbf{0}$. As before, this hop is justified through IND-CPA security of the encryption scheme $\mathcal{SE}$ via a different adversary E.

Notice that the proof does not rely on the security of the secret sharing scheme (indeed, key that is used to encrypt the shares is unpredictable so an adversary obtains no information even about a single share). The secret sharing scheme is used however to enforce that only decrypting with a close image (i.e. an image with sufficiently many overlapping features) succeeds.