HAL
open science

Group Testing for Identification with Privacy

Ahmet Iscen, Teddy Furon

To cite this version:

Ahmet Iscen, Teddy Furon. Group Testing for Identification with Privacy. ACM Workshop on Information Hiding and Multimedia Security, Jun 2016, Vigo, Spain. hal-01309032v2

HAL Id: hal-01309032

https://inria.hal.science/hal-01309032v2

Submitted on 30 May 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Group Testing for Identification with Privacy

Ahmet Iscen
Inria
Campus de Beaulieu
Rennes, France
ahmet.iscen@inria.fr

Teddy Furon
Inria
Campus de Beaulieu
Rennes, France
teddy.furon@inria.fr

ABSTRACT

This paper describes an approach where group testing helps in enforcing security and privacy in identification. We detail a particular scheme based on embedding and group testing. We add a second layer of defense, group vectors, where each group vector represents a set of dataset vectors. Whereas the selected embedding poorly protects the data when used alone, the group testing approach makes it much harder to reconstruct the data when combined with the embedding. Even when curious server and user collude to disclose the secret parameters, they cannot accurately recover the data. Another byproduct of our approach is that it reduces the complexity of the search and the required storage space. We show the interest of our work in a benchmark biometrics dataset, where we verify our theoretical analysis with real data.

1. INTRODUCTION

This paper considers a typical scenario involving the following three entities. The owner $\mathcal{O}$ has a collection of N objects. The user $\mathcal{U}$ would like to know whether there is an object in this collection similar enough to query object, and in that case, which object it is. The owner is not willing to operate the identification and outsources this task to a server $\mathcal{S}$. The targeted applications are typically related to multimedia retrieval, medical diagnosis, biometrics. In the later case, the owner $\mathcal{O}$ is a personification of the enrollment phase. A feature vector is extracted from the objects (iris, face captures) and used as a proxy: similar objects share similar features. Overall, this boils down to managing a database of vectors probed with query vectors.

We add the following privacy / security requirements:

- $\mathcal{U}$ doesn't want to reveal his query,
- $\mathcal{O}$ is reluctant in disclosing his database.

In other words, $\mathcal{S}$, the outsourced server is not trusted. This actor is deemed as semi-honest (or honest but curious): it operates the search task, however it tries to grab information about the query or the database. This would allow $\mathcal{S}$ to profile $\mathcal{U}$, *i.e.* to disclose what $\mathcal{U}$ is interested in, or to perform the search with unauthorized users, *i.e.* without the agreement of $\mathcal{O}$. In biometrics application, disclosing database vectors could lead to spoofing [5]. Some make the distinction between the privacy, which protects user $\mathcal{U}$, and the security, which protect the data of the owner $\mathcal{O}$.

Section 2 describes a classical solution based on embedding and secure multiparty computation. We selected the LSH embedding which maps real vectors to binary hashes. Evaluating the cosine similarity between vectors amounts to computing the Hamming distance between their hashes. We use a very efficient cryptographic protocol called SHADE for securing Hamming distances computations[1]: while $\mathcal{S}$ learns nothing from the query, $\mathcal{U}$ learns nothing from the database vector except the index of the most similar vector. This system enables privacy under the semi-honest model, but the security of owner's data is weak if $\mathcal{U}$ and $\mathcal{S}$ collude.

This weakness is due to $\mathcal{S}$ having binary hashes in the clear. The main counter-measure in literature is fully homomorphic encryption. It allows computing distances between two ciphers s.t. $\mathcal{S}$ now manipulates data previously encrypted by $\mathcal{O}$. However, computation in the encrypted domain is very low preventing the scalability of the system.

This paper trades privacy of the user and security of the data for scale of the database, speed of similarity search and quality of the identification. This is achieved by resorting to advanced signal processing rather than more cryptography. Section 3 describes a second version of the system adding on top of LSH and SHADE a group testing scheme for dealing with efficiency of the search while leaking only very little useful information.

1.1 Setup

The owner $\mathcal{O}$ has a database of vectors $\mathcal{X} := \{\mathbf{x}_i\}_1^N$ s.t. $\mathbf{x}_i \in \mathbb{R}^d$ with Euclidean norm $\|\mathbf{x}_i\| = 1, \forall i \in [N]$, where $[N] := \{1, \cdots, N\}$. We denote the query of the user by $\mathbf{q} \in \mathbb{R}^d$ with $\|\mathbf{q}\| = 1$. The similarity between the query and a database vector $\mathbf{x}$ is defined by $s(\mathbf{q}, \mathbf{x}) := \mathbf{q}^\top \mathbf{x}$.

In this paper, we analyse the proposed system having in mind biometrics *identification*. Database vectors are biometrics recorded at the enrollment phase. $\mathcal{U}$ is interested in knowing the index of the most similar vector in the database if similar enough:

$$\hat{\imath} := \begin{cases} \arg\max_{i \in [N]} s(\mathbf{q}, \mathbf{x}_i) & \text{if } s(\mathbf{q}, \mathbf{x}_{\hat{\imath}}) \geq \rho \\ \emptyset & \text{otherwise} \end{cases} \quad (1)$$

[1]Another option is partial homomorphic encryption

IH&MMSec 2016, June 20 - 23, 2016, Vigo, Spain.

ACM ISBN 978-1-4503-4290-2/16/06…$15.00.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
DOI: http://dx.doi.org/10.1145/2909827.2930792

There are two cases: The query is a noisy version of an enrolled vector $\mathbf{x}_{i^\star}$; or the query is random and we denote $i^\star = \emptyset$. For a given threshold ρ, three errors are possible:

False negative: $\hat{\imath} = \emptyset$ while $i^\star \neq \emptyset$
False identification: $\hat{\imath} \neq i^\star$ while $i^\star \neq \emptyset$
False positive: $\hat{\imath} \neq \emptyset$ while $i^\star = \emptyset$

We denote by P_{fn}, P_{fid}, and P_{fp} respectively the probabilities of these events. The 'real' search defined in (1) produces non zero error probabilities depending on how strongly the query is correlated with $\mathbf{x}_{i^\star}$ (when $i^\star \neq \emptyset$) and on the size N of the database. Due to privacy, $\mathcal{S}$ can not perform the real search as defined above. In agreement with $\mathcal{O}$, $\mathcal{S}$ runs a secure and approximate search with lower performances.

1.2 LSH Binary Embedding

An embedding casts a vector $\mathbf{x} \in \mathbb{R}^d$ into a vector of D discrete components.

$$\begin{aligned} e : \mathbb{R}^d &\rightarrow \mathcal{A}^D \\ \mathbf{x} &\rightarrow e(\mathbf{x}) := (e(\mathbf{x})_1, \cdots, e(\mathbf{x})_D) \end{aligned} \tag{2}$$

This function is designed to provide i) a compact representation of vectors, and ii) an estimation of the similarity $s(\mathbf{q}, \mathbf{x})$ between two vectors from their embeddings: $s_e(e(\mathbf{q}), e(\mathbf{x}))$. Function $s_e(\cdot,\cdot)$ is faster to compute than $s(\cdot,\cdot)$.

We propose to use one of the most well-known binary embeddings, LSH [4], where $\mathcal{A} = \{0, 1\}$. In a nutshell, once D directions $\{\mathbf{a}_k\}_{k=1}^D$ have been randomly drawn i.i.d. uniformly in $\mathbb{R}^d$, the k-th symbol of the embedding is computed as follows:

$$e(\mathbf{x})_k := \mathsf{sign}(\mathbf{x}^\top \mathbf{a}_k), \forall k \in [D], \tag{3}$$

with $\mathsf{sign}(x) := 1$ if $x \geq 0$, and 0 otherwise. LSH has the following well-known property:

$$\frac{\mathbf{x}^\top \mathbf{q}}{\|\mathbf{x}\|\|\mathbf{q}\|} \approx \cos\left(\frac{\pi}{D} d_H(e(\mathbf{q}), e(\mathbf{x}))\right). \tag{4}$$

The cosine is estimated via the Hamming distance $d_H(\cdot,\cdot)$. In our setup, database and query vectors have unit norm, the cosine is thus the metric we use to quantify similarity.

1.3 SHADE Protocol

Suppose now that $\mathcal{U}$ and $\mathcal{S}$ have respectively the binary words $\mathbf{w}^\mathcal{U}$ and $\mathbf{w}^\mathcal{S}$, and that $d_H(\mathbf{w}^\mathcal{U}, \mathbf{w}^\mathcal{S})$ is needed. SHADE is an efficient protocol [3] allowing $\mathcal{U}$ to learn nothing about $\mathbf{w}^\mathcal{S}$ except $d_H(\mathbf{w}^\mathcal{U}, \mathbf{w}^\mathcal{S})$, while $\mathcal{S}$ gains no information about $\mathbf{w}^\mathcal{U}$. In short, at the k-th round, $\mathcal{S}$ creates two messages: $m_0^{(k)} = \mathbf{w}_k^\mathcal{S} + r_k$ and $m_1^{(k)} = (\mathbf{w}_k^\mathcal{S} \bigoplus 1) + r_k$, where r_k is an alea uniformly distributed over $\mathbb{Z}_{D+1}$. Thanks to an oblivious transfer, $\mathcal{U}$ receives $v_k := m_{\mathbf{w}_k^\mathcal{U}}^{(k)}$. After D rounds, $\mathcal{S}$ sends $R := \sum_{k=1}^D r_k$ to $\mathcal{U}$ who computes $V := \sum_{k=1}^D v_k$ and $V - R = d_H(\mathbf{w}^\mathcal{U}, \mathbf{w}^\mathcal{S})$. SHADE is secure in the static semi-honest model: $\mathcal{U}$ and $\mathcal{S}$ are honest but curious. There is an efficient version of SHADE for computing a batch of Hamming distances between one query and N vectors [2].

1.4 Adding Comparison and Minimum

Letting $\mathcal{U}$ learning the Hamming distance $d_H(\mathbf{w}^\mathcal{U}, \mathbf{w}^\mathcal{S})$ is dangerous in the dynamic model where $\mathcal{U}$ is allowed to perform several distance computations: With D well chosen queries, $\mathcal{U}$ can disclose $\mathbf{w}^\mathcal{S}$. The authors of SHADE recommend the use of the GMW protocol [8] to first securely compare $V - R$ to a threshold τ. $\mathcal{U}$ only gets $\mathsf{sign}(d_H(\mathbf{w}^\mathcal{U}, \mathbf{w}^\mathcal{S}) - \tau)$. For a batch of distances, their minimum is securely computed and $\mathcal{U}$ only learns $\hat{\imath} = \arg\min d_H(\mathbf{w}^\mathcal{U}, \mathbf{w}^\mathcal{S})$.

2. FIRST VERSION OF THE SYSTEM

A system is a list of procedures followed by the three actors $\mathcal{O}$, $\mathcal{S}$ and $\mathcal{U}$. The owner $\mathcal{O}$ draws D random directions stacked in matrix $\mathbf{A} := [\mathbf{a}_1, \cdots, \mathbf{a}_D]$, computes the embeddings of its vectors, and sends $\mathcal{S}$ the database $\mathcal{E} := \{e(\mathbf{x}_i)\}$. $\mathcal{O}$ grants $\mathcal{U}$ the access of the identification by sending $\mathbf{A}$ so that $\mathcal{U}$ can compute the embedding of its query $\mathbf{q}$.

At query time, entities $\mathcal{U}$ and $\mathcal{S}$ perform the SHADE protocol, where $\mathbf{w}^\mathcal{U} = e(\mathbf{q})$ and $\mathbf{w}^\mathcal{S} = e(\mathbf{x}_i)$ with the batch option. $\mathcal{U}$ maps the Hamming distances $d_H(e(\mathbf{x}_i), e(\mathbf{q}))$ to similarity estimates $\hat{s}(\mathbf{q}, \mathbf{x}_i)$ thanks to (4). $\mathcal{U}$ finds the index with the biggest similarity and takes it as the output if it is above the threshold ρ. The search is approximate because it is based on similarity estimates.

An option is to add GMV protocol to let $\mathcal{U}$ learn either indices of vectors whose approximated similarity with the query is above a threshold (secure comparison), or the index of the vector whose approximated similarity is maximum (secure maximum). Note that this is done on the Hamming distances because (4) is a monotonic mapping. A combination of the two implement the search as defined in (1).

2.1 Analysis

Complexity: The quality of the approximate search depends on D, the embedding length. For any two vectors $\mathbf{x}$ and $\mathbf{q}$ s.t. $\mathbf{q}^\top \mathbf{x} = \cos(\theta)$, and $\mathbf{A}$ randomly generated as early described, $\frac{\pi}{D} d_H(e(\mathbf{q}), e(\mathbf{x}))$ in (4) is indeed an estimation of θ with no bias and variance $\theta(\pi - \theta)/D$. However, the complexity of the SHADE protocol deeply depends on D. The secure computation of a batch of size N has a complexity [2]:

$$\mathcal{C} \propto [(2ND \log_2(D))/o + 3D]\mathcal{C}_{sym}, \tag{5}$$

where o is a constant setting the security of the protocol (at least 128), and $\mathcal{C}_{sym}$ is the complexity of one symmetric cryptography operation (*i.e.* AES encryption / decryption).

Even though SHADE has been a breakthrough considerably lowering the complexity of the secure computation of Hamming distances, it is the bottleneck of our system. It takes 0.2s to securely compute and run the GMV protocol over $N = 200$ embeddings of length $D = 900$ (see [2]).

Security: SHADE enables the privacy of $\mathcal{U}$, but the owner $\mathcal{O}$ has some concerns. The parameter of the embedding (here matrix $\mathbf{A}$) is generated by the owner $\mathcal{O}$ and only shared with $\mathcal{U}$. This parameter is the only secret that prevents a curious server $\mathcal{S}$ from performing illegal identification or estimating the database vectors by inverting the embedding. By colluding with one user, $\mathcal{S}$ learns this secret.

Since (2) is a surjection, the embedding is not reversible and perfect reconstruction of $\mathcal{X}$ is impossible. However, one can see the embedding as a quantizer producing quantification noise. Sect. 2.2 measures the accuracy of a reconstruction of $\mathbf{x}$ from $e(\mathbf{x})$ provided matrix $\mathbf{A}$ is known.

2.2 Inverting LSH

A simple reconstruction of a unit norm vector $\mathbf{x}$ from its embedding is:

$$\hat{\mathbf{x}} = \kappa(\mathbf{A}(2e(\mathbf{x}) - \mathbf{1}_D)), \tag{6}$$

where κ is s.t. $\mathbb{E}_\mathbf{A}[\|\hat{\mathbf{x}}\|^2] = 1$. To quantify its accuracy, we introduce $a(\mathbf{x}) := \mathbf{x}^\top \hat{\mathbf{x}}$. The closer to 1, the better the

reconstruction. We have:

$$a(\mathbf{x}) = \kappa \sum_{k=1}^{D} |\mathbf{a}_k^\top \mathbf{x}|. \quad (7)$$

LSH was originally proposed with $\{\mathbf{a}_k\}$ being independent random directions in $\mathbb{R}^d$. In this case, the appendix shows:

$$\mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] \approx \sqrt{\frac{g}{1+g}} \text{ with } g := \frac{2D}{\pi d}. \quad (8)$$

However, if $D \leq d$, it is known that $\mathbf{A}$ being a random basis of rank D yields a better search [11]. In this case, $\|\mathbf{A}(2e(\mathbf{x}) - \mathbf{1}_D)\|^2 = D$ for any $\mathbf{x}$. The appendix shows:

$$\mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] \approx \sqrt{g} \text{ with } g \leq 1. \quad (9)$$

If $D > d$, a good choice is $\mathbf{A}$ as a random tight frame [10] s.t. $\|\mathbf{A}(2e(\mathbf{x}) - \mathbf{1}_D)\|^2 = d/D\|2e(\mathbf{x}) - \mathbf{1}_D)\|^2 = d$. Yet, the r.v. $\mathbf{a}_k^\top \mathbf{x}$ are no longer independent and it is hard to say something more than:

$$\sqrt{\frac{2}{\pi}} \approx 0.80 \leq \mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] \leq 1. \quad (10)$$

Approximate search based on LSH embedding usually sets $D > d$. On expectation, $\hat{\mathbf{x}}$ is then a good approximation of $\mathbf{x}$ since they correlate more than 0.8.

Fig. 4 shows the reconstruction accuracy w.r.t. D, when $\mathbf{A}$ is a random tight frame or random Gaussian matrix.

2.3 Lessons Learnt from the First System

The parameter of utmost importance is the length of the embedding D. It sets a trade-off between the quality of the identification, the complexity of the protocol, and the security when $\mathbf{A}$ has been compromised. Yet, this trade-off is poor as the reconstruction provides a good accuracy when $D > d$.

3. THE PROPOSED SYSTEM

Our aim is to provide a second line of defense. Group testing was recently introduced in approximate search. The idea is to pack database vectors into groups and to compute a representation per group, so-called memory vector, which allows to perform a group test. This test reveals whether the query is similar to at least one of the vectors in the group. For large dimension d, more vectors can be packed into one group because the correlation between two independent vectors with unit norm concentrates around 0.

Each database vector belongs to several groups. The decoding aims at identifying the matching vector(s) from the results of the group tests. A matching vector is defined as having a similarity with the query high enough: $s(\mathbf{q}, \mathbf{x}) \geq \alpha$. Naively, matching vectors are lying at the intersection of the groups yielding a positive group test. Things are not that simple because these tests suffer from false positives and false negatives. Yet, if the number of groups is large enough, the decoding succeeds in identifying matching vectors.

This approach brings two advantages to our system:

- The number of groups M is smaller than the number of vectors N. This lowers both the storage space and the complexity of the protocol by a gain $\gamma := M/N < 1$.
- $\mathcal{O}$ will not give $\mathcal{S}$ embeddings of database vectors, but of memory vectors. This makes harder the reconstruction of database vectors.

Table 1: Definition of the weights

	$b_{i,j} = 1$	$b_{i,j} = 0$
$\mathcal{H}_i$	$\eta = 1$	$\eta = 0$
$\bar{\mathcal{H}}_i$	$\eta = (n-1)/N$	$\eta = n/N$

3.1 Encoding

The owner $\mathcal{O}$ randomly packs the N database vectors into M groups $\{\mathcal{G}_j\}_{j=1}^M$ s.t. i) any group comprises n vectors, ii) any vector belongs to m groups. This rules enforces that $M = mN/n$. A possible construction is explained in [7]. We call the map the $M \times N$ binary matrix $\mathbf{B}$ indicating which vector belongs to which group: $b_{i,j} = 1$ if $\mathbf{x}_i$ belongs to group $\mathcal{G}_j$, 0 otherwise.

Then, $\mathcal{O}$ computes the memory vectors, *i.e.* the representatives of each group. We adopt here the most simple constructions investigated in [7]:

$$\mathbf{m}_j = \sum_{i:b_{i,j}=1} \mathbf{x}_i. \quad (11)$$

Finally, $\mathcal{O}$ computes the embeddings of the memory vectors and sends $\mathcal{S}$ the following compact description $\mathcal{E}' = \{e(\mathbf{m}_j)\}_{j=1}^M$. $\mathcal{O}$ sends $\mathcal{U}$ parameters $(\mathbf{A}, \mathbf{B}, \{\|\mathbf{m}_j\|\}_{j=1}^M)$: $\mathbf{A}$ is needed to compute query embedding, $\mathbf{B}$ and $\{\|\mathbf{m}_j\|\}_{j=1}^M$ to decode the tests. Note that the database $\mathcal{E}'$ is smaller than $\mathcal{E}$ since it has $M < N$ entries.

3.2 Querying

$\mathcal{U}$ computes $e(\mathbf{q})$ thanks to $\mathbf{A}$ and runs the SHADE protocol with $\mathcal{S}$, who learns nothing about the query. $\mathcal{U}$ obtains estimations of the cosine between $\mathbf{q}$ and $\mathbf{m}_j$. Multiplying by the norm $\|\mathbf{m}_j\|$, this gives estimated similarities $\hat{s}_j \approx \mathbf{q}^\top \mathbf{m}_j$.

The big benefit is the decrease of SHADE complexity, which was the bottleneck of the previous system. Instead of securely computing N Hamming distances $d_H(e(\mathbf{q}), e(\mathbf{x}_i))$, we compute $M < N$ distances $d_H(e(\mathbf{q}), e(\mathbf{m}_j))$.

Soft decoding: From $\{\hat{s}_j\}_{j=1}^M$ and map $\mathbf{B}$, $\mathcal{U}$ runs the decoding to identify $i^\star$, index of the matching vector.

In summary, the decoding computes N scores. c_i is the likelihood ratio w.r.t. two hypothesis: $\mathbf{x}_i$ is the matching vector ($\mathcal{H}_i$) or not ($\bar{\mathcal{H}}_i$).

$$c_i = \sum_{j=1}^{M} \log \frac{f_{\mathcal{H}_i}(\hat{s}_j, b_{i,j})}{f_{\bar{\mathcal{H}}_i}(\hat{s}_j, b_{i,j})}, \quad (12)$$

where the pdfs are modeled as mixtures of two Gaussian distributions, $\eta\mathcal{N}(\alpha; (n-1)/d) + (1-\eta)\mathcal{N}(0; n/d)$, with parameter η given in Table 1. We refer the reader to [7] for justifications of this statistical model. Then, $\mathcal{U}$ computes the maximum of these scores and compares with the threshold.

Hard decoding: To prevent oracle attacks from $\mathcal{U}$, we use the GMW protocol to apply a secure comparison: $\mathcal{U}$ learns nothing more than $d_j = \mathsf{sign}(\hat{s}_j - \tau)$, where threshold τ has been carefully selected by $\mathcal{O}$. Then, the decoding computes the following scores:

$$c_i = \sum_{j=1}^{M} d_j \log \frac{p_{\mathcal{H}_i}(b_{i,j})}{p_{\bar{\mathcal{H}}_i}(b_{i,j})} + (1-d_j) \log \frac{1 - p_{\mathcal{H}_i}(b_{i,j})}{1 - p_{\bar{\mathcal{H}}_i}(b_{i,j})}, \quad (13)$$

with $p_{\mathcal{H}_i}(b_{i,j}) = \mathbb{E}_{\hat{s}_j \sim f_{\mathcal{H}_i}(\hat{s}_j, b_{i,j})}[\hat{s}_j > \tau]$:

$$p_{\mathcal{H}_i}(b_{i,j}) = \eta\Phi\left((\tau-\alpha)\sqrt{\frac{d}{n-1}}\right) + (1-\eta)\Phi\left(\tau\sqrt{\frac{d}{n}}\right)$$

and parameter η given in Table 1.

4. SECURITY

Unauthorized identification is possible whenever an untrusted actor, be it $\mathcal{U}$ and/or $\mathcal{S}$, has in his hands both $(\mathbf{A}, \mathbf{B}, \{\|\mathbf{m}_j\|\}_{j=1}^M)$ and $\mathcal{E}'$. This happens only when $\mathcal{S}$ and $\mathcal{U}$ colluded, or if $\mathcal{U}$ succeeds to steal $\mathcal{E}'$.

The group testing approach brings a second line of defense by mixing the database vectors into memory vectors. We focus here on the reconstruction of the database vectors. To do so, the untrusted actor must i) reconstruct the memory vectors by 'inverting' their embeddings, and ii) estimate database vectors from the reconstructed memory vectors of groups they belong to. Reconstruction i) needs $\mathcal{E}'$, $\mathbf{A}$ and $\{\|\mathbf{m}_j\|\}$. The quality of the reconstruction is quite high as shown in Sec. 2.2. Estimation ii) needs $\mathbf{B}$. The quality of the reconstruction is investigated in the next section.

4.1 Inverting Memory Units

Equation (11) can be rephrased as $\mathbf{M} = \mathbf{X}\mathbf{B}^\top$ where $\mathbf{M} := [\mathbf{m}_1, \cdots, \mathbf{m}_M]$ is a $d \times M$ matrix storing the memory units while $\mathbf{X} := [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ stores the database vectors and $\mathbf{B}$ is the map (See Sect. 3.1). Estimating back $\mathbf{X}$ from $\mathbf{M}$ is possible using a ridge regression or the pseudo-inverse of $\mathbf{B}^\top$: $\hat{\mathbf{X}} \propto \mathbf{M}(\mathbf{B}^\top)^\dagger$. We can show that reconstructing $\mathbf{x}$ from the exact memory units, produces an estimation $\hat{\mathbf{x}}$ s.t.

$$\mathbb{E}[a(\mathbf{x})] = \min(1, \sqrt{\gamma}). \tag{14}$$

Yet, this requires the inverse of large $M \times M$ matrix $\mathbf{B}\mathbf{B}^\top$. The average of some memory vectors albeit not optimal is faster: $\hat{\mathbf{X}} \propto \mathbf{M}\mathbf{B}$ achieving ($\nu := n/N$):

$$\mathbb{E}[a(\mathbf{x})] \approx \sqrt{\frac{\gamma}{1+\nu^2(M-1)+\gamma(1-\nu)^2}}. \tag{15}$$

4.2 Full Reconstruction

The attacker first reconstructs the group vectors $\{\hat{\mathbf{m}}_k\}$ from their embeddings $\{e(\mathbf{m}_k)\}$, and then reconstructs the database vectors $\{\hat{\mathbf{x}}_i\}$. It is easy to show that if the first step produces a reconstruction accuracy measured by $\mathbb{E}[a(\mathbf{m})] = a_1$ while the second step achieves $\mathbb{E}[a(\mathbf{x})] = a_2$ starting from true memory vectors, then the total reconstruction yields $\mathbb{E}[a(\mathbf{x})] = a_1 a_2$ thanks to the linearity of the second step. This is evidenced in the experimental work.

5. EXPERIMENTS

5.1 Experimental Setup

We test our system using both synthetic and real data. For both cases, we keep the ratio $\gamma = M/N = m/n = 0.1$. This means that the retrieval system needs only 0.1 of memory storage and vector comparisons compared to exhaustive search. The embedding is parametrized s.t. $D = 2d$.

Synthetic Data. We create a synthetic dataset of N vectors distributed randomly on the unit hypersphere of $\mathbb{R}^d$. We then create N_q random query vectors, such that each query has exactly one match in the dataset, $\mathbf{x}_{i^\star}$, and the similarity between the query and the matching vector is $\mathbf{q}^\top \mathbf{x}_{i^\star} = \alpha$. We set $d = 1,920$, $N = 10,000$, $N_q = 100$, and $\alpha = 0.5$.

Real Data. We also use Labeled Faces in the Wild (LFW) dataset [6], which has 13,233 images of 5,749 people. For the query set, we choose a random image from people who have i) at least two images , ii) have at least one match with a similarity greater than or equal to α. We use the full ($d = 67,584$) Fisher face descriptors [9] to calculate the similarities to generate the query set. Then, we use the same query set for all experiments, regardless of different descriptors or dimensionality. Setting $\alpha = 0.5$ gives us $N_q = 104$ queries, each belonging to a different person. We also choose $1,000$ random queries from people who have no other matching vectors. All queries and random queries are removed from the dataset.

For our experiments, we reduce the dimension of Fisher face descriptors from $67,584$ to $1,920$ for fair comparison with synthetic data. We also use CNN-based descriptors [1], whose dimension is reduced from $4,096$ to $1,920$.

Parameters setting. As stated earlier, we can set the ratio of $M/N = 0.1$ with different m and n combinations. We choose the optimal setting empirically by maximizing the Kullback-Leibler distance between negative and positive distributions. This gives $n = 200$, $m = 20$, and $\tau = 0.8 \times \alpha$.

Evaluation. The three metrics for evaluating the performance are the probabilities of the three types of error, P_{fp}, P_{fn} and P_{fid} as functions of threshold ρ (see Sect. 1.1). Security is gauged by the quality of the reconstruction of the database vectors measured by $\mathbb{E}[a(\mathbf{x})]$.

5.2 Approximate Search

Fig. 1 shows the performance of the identification of the baseline, *i.e.* the exhaustive search on real vectors, without any privacy and security issues, as defined in (1). Fig. 2 shows the same evaluation for the first system described in Sect. 2. Fig. 3 shows the system proposed in Sect. 3 with the hard decoding variant (13). We are smoothly degrading the performance of the approximate search over synthetic data, however it still performs well on the real dataset.

5.3 Security

Fig. 4 shows the reconstruction performance while inverting LSH with the synthetic dataset. It encompasses two matrix $\mathbf{A}$ generation procedures: Gaussian i.i.d. entries and uniform tight frame. The latter option is known to produce better approximate search. This illustrates the security of the first system described in Sect. 2. In our setup where $D = 2d$, we end up with $\mathbb{E}[a(\mathbf{x})] \approx 0.88$.

Fig. 5 shows the reconstruction performance while inverting the memory units with $m \in [2, 20]$ and $n = 200$ on the synthetic dataset. It encompasses two reconstruction methods: the pseudo inverse of $\mathbf{B}$ and the average of memory vectors (See Sect. 4.1). This illustrates the security improvement thanks to the second line of defense provided by group testing. In our setup where $n = 200$ and $m = 20$, we end up with $\mathbb{E}[a(\mathbf{x})] \approx 0.33$.

Overall, the reconstruction of vectors $\{\mathbf{x}_i\}$ from $\{e(\mathbf{m}_k)\}$ outputs estimates correlated on expactation $\mathbb{E}[a(\mathbf{x})] \approx 0.88 \times 0.33 = 0.29$. We run the attack on the real datasets and get $\mathbb{E}[a(\mathbf{x})] \approx 0.88 \times 0.31 = 0.27$.

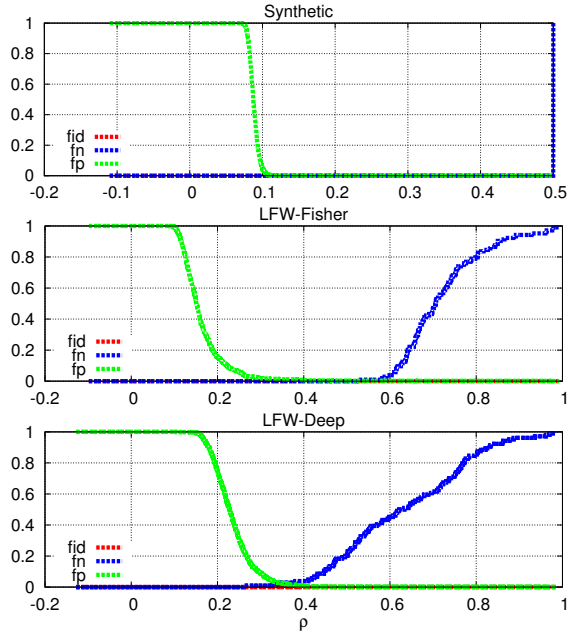

Figure 1: Baseline performance, synthetic and LFW (Fisher / deep learning features) datasets.

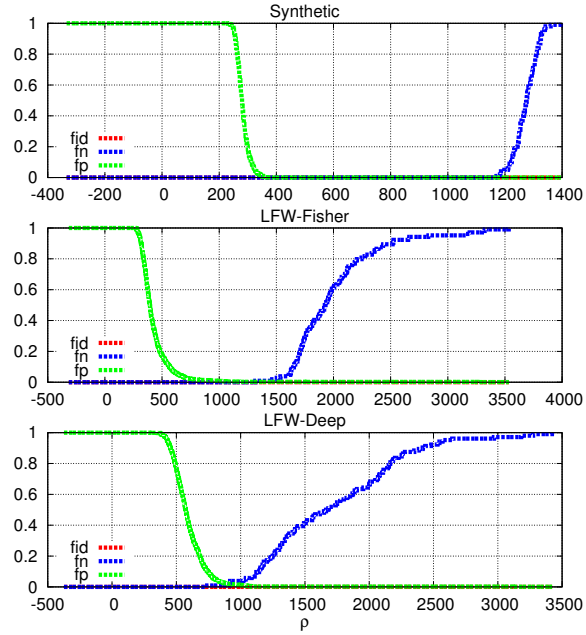

Figure 2: 1st system performance, synthetic and LFW (Fisher / deep learning features) datasets.

6. CONCLUSION

We presented a privacy and security enhancing scheme for approximate search. It is built upon a first version which enables users privacy but not security of the owner's data especially when user and server collude. Contrary to the "Signal Processing in the Encrypted Domain" trend which uses even more advanced cryptographic primitives like full homomorphic encryption, we propose an alternative only based on signal processing. It is much faster and more scalable (even more that the first version). Yet, the attack is not absolutely blocked, but relatively in the sense that vector reconstruction is so bad that it can't be exploited.

7. REFERENCES

[1] B. Bhattarai, G. Sharma, and F. Jurie. Cp-mtml: Coupled projection multi-task metric learning for large scale face retrieval. In *CVPR*, 2016.

[2] J. Bringer, H. Chabanne, M. Favre, A. Patey, T. Schneider, and M. Zohner. GSHADE: faster privacy-preserving distance computation and biometric identification. In *Proceedings of the 2Nd ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '14, pages 187–198, New York, NY, USA, 2014. ACM.

[3] J. Bringer, H. Chabanne, and A. Patey. SHADE: secure hamming distance computation from oblivious transfer. In *Financial Cryptography and Data Security*, volume 7862 of *Lecture Notes in Computer Science*, pages 164–176, 2013.

[4] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, May 2002.

[5] A. Hadid, N. Evans, S. Marcel, and J. Fierrez. Biometrics systems under spoofing attack. *IEEE Signal Processing Magazine*, 32(5):20–, September 2015.

[6] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Oct. 2007.

[7] A. Iscen, T. Furon, V. Gripon, M. G. Rabbat, and H. Jégou. Memory vectors for similarity search in high-dimensional spaces. *CoRR*, abs/1412.3328, 2014.

[8] T. Schneider and M. Zohner. GMW vs. Yao? efficient secure two-party computation with low depth circuits. In S. B. Heidelberg, editor, *Financial Cryptography and Data Security*, 2013.

[9] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In *British Machine Vision Conference*, 2013.

[10] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. Technical report, Department of Engineering Science, University of Oxford, 2013.

[11] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, December 2009.

APPENDIX

A. RECONSTRUCTION FROM LSH

Assuming that $\mathbf{A}$ has an isotropic distribution (either independent Gaussian entries or a random frame), w.l.o.g. we set $\mathbf{x} = (1, 0, \ldots, 0)$ and

$$\|\hat{\mathbf{x}}\|^2 = \kappa^2 \left(\sum_{k=1}^{D} \|\mathbf{a}_k\|^2 + \sum_{k \neq \ell} |a_k(1)||a_\ell(1)| \right) \tag{16}$$

If $a_k(1) \sim \mathcal{N}(0,1)$, $\mathbb{E}_{\mathbf{A}}[\|\hat{\mathbf{x}}\|^2] = \kappa^2 D(d + 2(D-1)/\pi)$ and

$$\kappa \approx \sqrt{\frac{1}{Dd(1+g)}} \text{ with } g := \frac{2D}{\pi d}. \tag{17}$$

For this $\mathbf{x}$, $a(\mathbf{x}) = \kappa \sum_{k=1}^{D} |a_k(1)|$ s.t.

$$\mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] = \kappa D \sqrt{2/\pi} = \sqrt{\frac{g}{1+g}}. \tag{18}$$

If $\mathbf{a}_k$ is uniformly distributed on the unit sphere (random

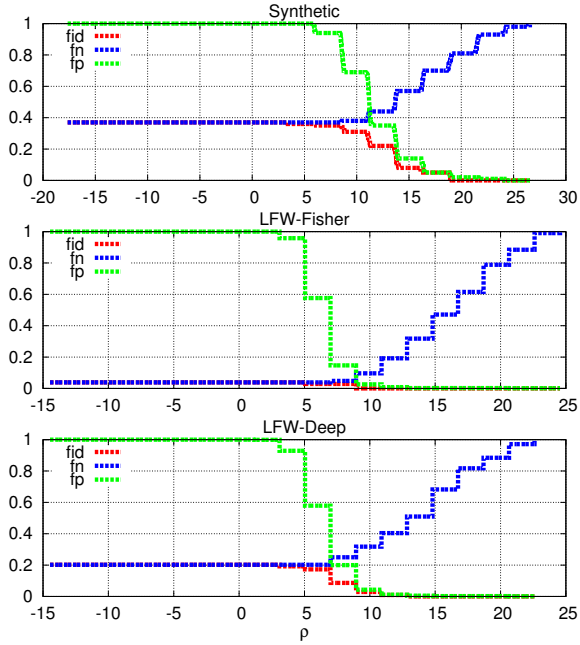

Figure 3: Proposed system performance, synthetic and LFW (Fisher / deep learning features) datasets.

uniform frame or basis), the marginal distribution of $a_k(1)$:

$$f(s) = \frac{(1-s^2)^{\frac{d-3}{2}}}{B(1/2,(d-1)/2)}, \forall s, -1 \leq s \leq 1, \qquad (19)$$

s.t. $\mathbb{E}[|a_k(1)|] = \sqrt{2/d\pi}(d-2)!!/(d-1)!! \approx \sqrt{2/d\pi}$. For $D \leq d$, the random uniform tight frame is indeed a basis: $\mathbb{E}_{\mathbf{A}}[\|\hat{\mathbf{x}}\|^2] = D$ and $\mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] = \sqrt{g}$.

B. RECONSTRUCTION FROM MEMORY

We assume the following model for matrix $\mathbf{B}$. For row r, $1 \leq r \leq M$, we randomly select n indices in $[N]$ and set these coefficients to 1. With a high probability $\mathsf{rank}(\mathbf{B}) = M$ and $\mathsf{Tr}(\mathbf{B}\mathbf{B}^\top) = nM$. As for the vector, $\mathbb{E}_{\mathbf{X}}[\mathbf{X}^\top\mathbf{X}] = \mathbf{I}_N$.

Pseudo-inverse reconstruction: $\hat{\mathbf{X}} = \eta\mathbf{M}(\mathbf{B}^\top)^\dagger$, with $(\mathbf{B}^\top)^\dagger = (\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B}$. Constant η is s.t. the expectation (over $\mathbf{X}$ and $\mathbf{B}$) of the average squared norm of $\hat{\mathbf{x}}_i$ equals 1.

$$\begin{aligned}\frac{\mathbb{E}[\sum_{i=1}^N \|\hat{\mathbf{x}}_i\|^2]}{N} &= \mathbb{E}[\mathsf{Tr}(\hat{\mathbf{X}}^\top\hat{\mathbf{X}})]/N \qquad (20)\\ &= \eta^2\mathbb{E}[\mathsf{Tr}(\mathbf{X}^\top\mathbf{X}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B})]/N\\ &= \eta^2\mathbb{E}[\mathsf{Tr}((\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B}\mathbf{B}^\top)]/N = \eta^2 M/N\end{aligned}$$

The expectation of the average correlation is given by:

$$\begin{aligned}\frac{\mathbb{E}[\sum_{i=1}^j \mathbf{x}_i^\top\hat{\mathbf{x}}_i]}{N} &= \mathbb{E}[\mathsf{Tr}(\mathbf{X}^\top\hat{\mathbf{X}})]/N \qquad (21)\\ &= \eta\mathbb{E}[\mathsf{Tr}(\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B})]/N = \sqrt{M/N}.\end{aligned}$$

Sum reconstruction: $\hat{X} = \eta\mathbf{M}\mathbf{B}$. Constant η is s.t. the expectation of the average squared norm of $\|\hat{\mathbf{x}}_i\|^2$ equals 1.

$$\begin{aligned}\frac{\mathbb{E}[\sum_{i=1}^N \|\hat{\mathbf{x}}_i\|^2]}{N} &= \eta^2\mathbb{E}[\mathsf{Tr}(\hat{\mathbf{X}}^\top\hat{\mathbf{X}})]/N\\ &= \eta^2\mathbb{E}[\mathsf{Tr}((\mathbf{B}\mathbf{B}^\top)^2)]/N. \qquad (22)\end{aligned}$$

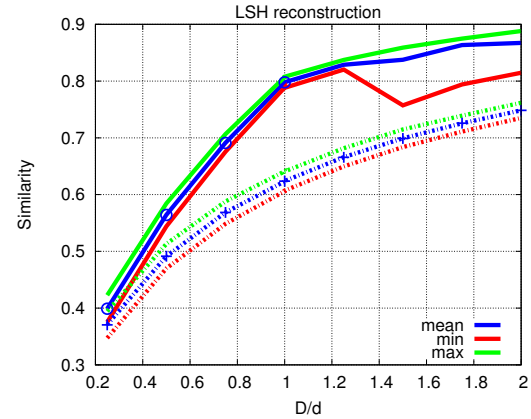

Figure 4: Reconstruction from LSH: $a(\mathbf{x})$ as a function of D/d. Empirical mean, min. and max. over 10,000 reconstructions. Unif. Tight Frame (plain), Gaussian i.i.d. (dashed), Eq. (8) (+) and Eq. (9) (o).

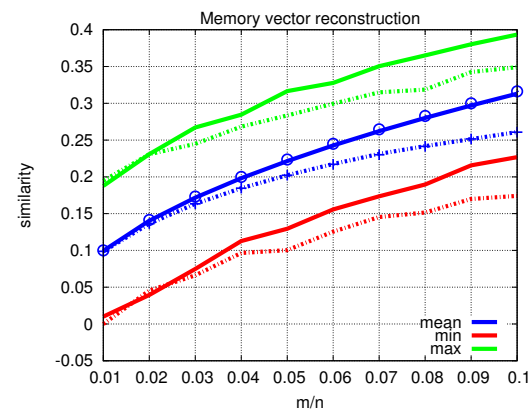

Figure 5: Reconstruction from memory vectors: $a(\mathbf{x})$ as a function of m/n. Empirical mean, min., and max. over 40,000 reconstructions. Pseudo-inverse (plain), average (dashed), Eq. (14) (o) and (15) (+).

We define $\tilde{\mathbf{B}} := \frac{(\mathbf{B}-\nu\mathbf{1}_{M:N})}{\nu(1-\nu)}$ with $\nu := n/N$. Its columns are centered random vectors whose covariance matrix is the identity. The eigenvalues of their empirical covariance matrix $\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top/N$ follows the Marchenko-Pastur distribution. For large N and $\gamma = M/N$, we then have:

$$\mathbb{E}[\mathsf{Tr}((\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top)^2)]/MN^2 = 1+\gamma. \qquad (23)$$

Expressing $(\mathbf{B}\mathbf{B}^\top)^2$ as a function of $\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top$ leads to:

$$\mathbb{E}[\mathsf{Tr}((\mathbf{B}\mathbf{B}^\top)^2)] = \nu^2MN^2(1-\nu^2+\gamma(1-\nu)^2+\nu^2M), \quad (24)$$

which gives the value of η thanks to (22). On the other hand, the expectation of the average correlation is given by:

$$\begin{aligned}\frac{\mathbb{E}[\sum_{i=1}^j \mathbf{x}_i^\top\hat{\mathbf{x}}_i]}{N} &= \eta\mathbb{E}[\mathsf{Tr}(\mathbf{X}^\top\mathbf{X}\mathbf{B}^\top(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}]/N\\ &= \eta\mathbb{E}[\mathsf{Tr}(\mathbf{B}^\top(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}]/N = \eta M/N\\ &= \sqrt{\frac{\gamma}{1-\nu^2+\gamma(1-\nu)^2+\nu^2M}} \qquad (25)\end{aligned}$$