Private Eyes: Zero-Leakage Iris Searchable Encryption

Julie Ha* Chloe Cachet† Luke Demarest‡ Sohaib Ahmad§ Benjamin Fuller¶

March 13, 2024

Abstract

This work introduces *Private Eyes*, the first zero-leakage biometric database. The leakage of the system is unavoidable: 1) the log of the dataset size and 2) the fact that a query occurred. Private Eyes is built from symmetric searchable encryption. Approximate proximity queries are used: given a noisy reading of a biometric, the goal is to retrieve all stored records that are close enough according to a distance metric.

Private Eyes combines locality sensitive-hashing or LSHs (Indyk and Motwani, STOC 1998) and oblivious maps. One computes many LSHs of each record in the database, and uses these hashes as keys in an encrypted map with the matching biometric readings concatenated as the value. At search time with a noisy reading, one computes the LSHs, and retrieves the disjunction of the resulting values from the map. The underlying encrypted map needs to efficiently answer disjunction queries.

We focus on the iris biometric. Iris biometric data requires a large number of LSHs, approximately 1000. Boldyreva and Tang's (PoPETS 2021) design yields a suitable map for a small number of LSHs (their application was in zero-leakage k-nearest-neighbor search).

Our cryptographic design is a zero-leakage disjunctive map designed for the setting when most clauses do not match any records. For the iris, on average at most 6% of LSHs match any stored value. Our scheme is implemented and open-sourced.

We evaluate using the ND-0405 dataset; this dataset has 356 irises suitable for testing. To scale our evaluation, we use a generative adversarial network to produce synthetic irises. Accurate statistics on sizes beyond available datasets is crucial to optimizing the cryptographic primitives. This tool may be of independent interest. For the largest tested parameters of a 5000 iris database, search requires 26 rounds of communication and 26 minutes of single-threaded computation.

Searchable Encryption, Biometrics, Proximity Search

1 Introduction

Biometrics are collected into large databases for search [BBOH96, Dau14, Fou]. Learning stored biometric values enables an attacker to break authentication and privacy for a user's lifetime [GRGB+12, MCYJ18, AF20, VS11, HWKL18, SDDN19]. To reduce this risk, this article develops new searchable encryption techniques for biometric databases [SWP00, CGKO11]. See previous reviews of searchable encryption [BHJP14, FVY+17, KKM+22].

A client outsources a database $\mathcal{DB}$ to an honest but curious server that may learn information called *leakage*. Prior work exploits such leakage to reveal sensitive information about the database or queries [IKK12, CGPR15, KKNO16, WLD+17, GSB+17, GLMP18, KPT19a, MT19, KE19, KPT20, FMC+20, FP22, GPP23, KKM+22]. Since biometrics cannot be replaced or revoked, we insist on the importance of a **zero-leakage system. A zero-leakage system leaks only unavoidable information: 1) that a query occurs and 2) $|\mathcal{DB}|$ (which we pad to a power of 2).** Our focus is on an efficient combination of zero-leakage primitives for this task. Our components can be replaced with higher-leakage components for efficiency gains.

*Boston University. Email hajulie@bu.edu
†University of Connecticut. Email chloe.cachet@uconn.edu
‡University of Connecticut. Email luke.h.demarest@gmail.com
§University of Connecticut. Email sohaib.ahmad@uconn.edu
¶University of Connecticut. Email benjamin.fuller@uconn.edu

1.1 Our Contribution

We present `PrivateEyes`, the first zero-leakage iris proximity search system. `PrivateEyes` relies on the following contributions:

1. A parameter analysis that reveals critical inefficiencies of previous designs. Most biometric search systems phrase proximity queries as a large disjunction. For actual biometrics, 1) this disjunction has hundreds or thousands of terms and 2) most clauses will not match any stored record.

2. A two-stage design that uses lighter-weight cryptography to find non-null clauses. That is, clauses that match something in the database. We can then only obliviously search these non-null clauses. Our first phase can be built with private set intersection (PSI). We benchmark with VolePSI [RS21]. Our second phase is built using oblivious tree traversal [WNL+14].

3. A prototype implementation with evaluation on random and iris data up to 5000 records [CHF24]. Search time is at most 26 minutes of single threaded execution and 26 round trips. Our approach is embarrassingly parallel. We focus on the iris (we briefly discuss the face in Section 3.1). A critical building block of our system is oblivious RAM (ORAM) where we use a PathORAM [SDS+18a] implementation that averages 1s per access on a moderate size dataset. Maas et al. [MLS+13] hardware implementation is 2000 times faster. We expect 26 round trips to be the dominant cost in most settings.

To scale beyond the dataset sizes available for irises we also introduce a synthetic iris generation tool that may be of independent interest.

Organization The rest of this paper is organized as follows: Section 2 reviews our design including relevant prior work, Section 3 introduces preliminaries, Section 4 presents the details of our system, Section 5 presents the datasets, Section 6 describes our implementation, Section 7 our evaluation methodology, and Section 8 concludes. Appendix A describes the architecture that generates synthetic irises.

2 Design Overview

Like prior work [KIK12, FWG+16, WYLH14, LPW+20], our design combines locality-sensitive hashes (LSHs) [IM98] with a variant of encrypted maps.[1] A database is a list of biometrics $\mathcal{DB} = w_1, ..., w_\ell$ where each $w_i \in \{0,1\}^n$. The goal of a biometric database is given some w^* to find all values $w_i \in \mathcal{DB}$ that are similar enough to w^*. For the Hamming metric $\mathcal{D}$ and distance threshold t, the goal is to find all w_i such that $\mathcal{D}(w_i, w^*) \leq t$.[2] An LSH maps *near* items to the same value more frequently than it maps *far* items to the same value. Let $\mathcal{H}$ be a family of LSHs then

$$\Pr_{\mathsf{LSH} \leftarrow \mathcal{H}}[\mathsf{LSH}(w_i) = \mathsf{LSH}(w^*) | w_i, w^* \text{ are near}] \geq 1 - p_1,$$
$$\Pr_{\mathsf{LSH} \leftarrow \mathcal{H}}[\mathsf{LSH}(w_j) = \mathsf{LSH}(w^*) | w_j, w^* \text{ are far}] \leq 1 - p_2.$$

where $p_1 < p_2$. Maps associate keys to a value and are used to build inverted indices. For a database of size ℓ, parameter $\beta \in \mathbb{Z}^+$, maps $\mathsf{M}_1, ..., \mathsf{M}_\beta$, and LSH family $\mathcal{H}$, one can achieve proximity search as follows:

1. Sample β LSHs, $\mathsf{LSH}_1, ..., \mathsf{LSH}_\beta \leftarrow \mathcal{H}$.
2. For $j = 1, ..., \beta$, set $\mathsf{M}_j[v] = \{w_i | \mathsf{LSH}_j(w_i) = v\}$.[3]
3. To search for value w^*:
   (a) Compute $\mathsf{LSH}_1(w^*), \cdots, \mathsf{LSH}_\beta(w^*)$.

[1] We don't discuss works that use encrypted maps but require work proportional to the total number of close points, making them impractical for biometrics [LWW+10, WMT+13, BC14].

[2] This functionality differs from k-nearest neighbors where the goal is to retrieve the k closest records [BT21]. There have been leakage abuse attacks against k-nearest neighbor systems that reveal access pattern [KPT19a, KPT19b, LMWY20] and resulting systems [CCD+20]. These attacks do not apply to our leakage profile.

[3] If multiple records share the same LSH value our implementation concatenates the matching values. This allows us to handle a constant number of values associated to each key. This condition is satisfied for the accuracy regimes discussed in this work.

(b) Retrieve $\cup_{j=1}^{\beta} \mathsf{M}_j[\mathsf{LSH}_j(w^*)]$.

Note that the query is a disjunction. Boldyreva and Tang [BT21] constructed a zero-leakage encrypted map scheme called an *oblivious map with encryption* or OMapE. Each clause is submitted to the relevant OMapE, the results are concatenated as in Step 3b above.

Due to their large noise, **biometrics require sampling hundreds or thousands of LSHs** to achieve reasonable accuracy (see analysis in Section 3.1 and Section 7). At the same time, very **few of these LSHs will match anything** in the corresponding map. Constructions frequently perform heavy oblivious operations to hide the null value.

Our design separates the tasks of identifying which values are null and finding the associated values. That is, we first find a small number, called δ, of LSHs values that exist in some map, and then query exactly δ maps in a way that hides which δ maps are being queried. For the above design to be successful, one needs to demonstrate:

1. **Obliviousness** One can hide the queried δ maps,
2. **Accuracy** High accuracy with $\delta < \beta$, and
3. **Speed** The approach is faster.

We now provide a more formal description of the approach.

Oblivious Membership Check An object to check which of the LSHs have matches. For an encrypted stored set X the *oblivious membership check* or OMC takes in a set W and returns $W' \subseteq W \cap X$ where $|W'| \leq \delta$. The set X is the set of all LSH values $X = \{(j, \mathsf{LSH}_j(w_i))\}_{i,j}$. For a searched value w^* the set $W = \{(j, v) | \mathsf{LSH}_j(w^*) = v\}_j$. Our analysis finds a parameter δ that has a small impact on accuracy. We build OMC using private set intersection and pseudorandom permutations. We benchmark this design using VolePSI [RS21]; the resulting implementation is orders of magnitude faster than our oblivious map implementation.

Disjunctive Oblivious Map An object that directly searches for the disjunction of exactly δ items. [4]These δ values are the set of LSHs that exist in the map. Using an oblivious data structure with a constant number of queries, δ, yields a zero-leakage solution. This object has the same functionality as an oblivious map that takes multiple clauses but the fact that all clauses are presented together is crucial for security. We call this object a DOMapE for *disjunctive oblivious map with encryption*. Our focus is on designing a DOMapE.

The input size to both OMC.Search and DOMapE.Search are of constant size which suffices for the two objects and their composition to be zero-leakage. We show the composition in Figure 1.

To summarize, in the Encrypt stage of the protocol, the client stores the set of all LSH values using the OMC (with their index j). They also create the map associating LSH values to the corresponding biometric value. In the Search stage of the protocol, the client computes the LSH values for their input w^*, and uses OMC.Search to find which values exist in the map M. The map is then used to retrieve the corresponding non-null values.

Boldyreva and Tang [BT21] build β separate maps and search each one for an LSH clause. This approach does not work if one only queries δ clauses because it reveals which LSHs matched. We build a DOMapE based on oblivious tree traversal, building on the design principles of Wang et al. [WNL+14]. In our approach, one always performs δ tree traversals. We use oblivious RAMs to store tree nodes. The oblivious RAMs are organized to minimize nodes that are stored together while ensuring that δ tree traversals result in no leakage. This work implements and analyzes this construction on datasets of up to 5000 records. Because iris datasets are not this large, as part of this analysis, we create larger synthetic iris datasets using generative adversarial networks or GANs [CWD+18, GPAM+20].

Comparison with prior work More work is needed to scale zero-leakage biometric database to national biometric identity databases. Cachet et al. [ACD+22] proposed two non-interactive iris proximity search schemes based on inner-product encryption. Both of their constructions have more leakage than our system. The first one leaks the distance between all returned points and the query. The other leaks whether returned records are the same distance from the query. For the solution with more leakage, their search took 4 minutes on a dataset of size 356. For the solution with less leakage, search took 4 hours (reduced to an 1 hour in a journal version [CAD+23]).

[4]If $|W'| < \delta$ we search for dummy values. If $|W'| > \delta$ we select δ random values.

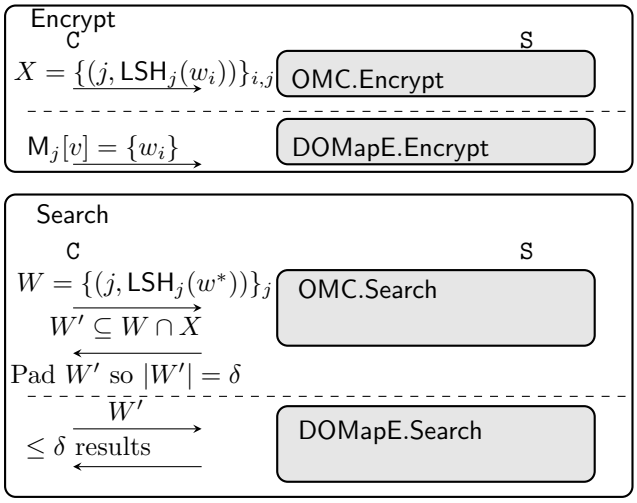


Figure 1: System Overview Composing OMC and DOMapE. In Search one checks which LSH values exist in the database using the OMC, finding at most δ candidates. One then pads with null values to search for exactly δ values in the DOMapE. We design a DOMapE that has no leakage if the size of the disjunction is constant (δ). See Construction 2 for a formal description.

Barni et al. [BBC+10] and Blanton et al. [BG11] evaluated their fingerprints identification systems on small datasets of 320 readings. They respectively achieved search times of 16 and 0.45 seconds. However, it is important to note that fingerprints representations are usually shorter than irises.

Blanton et al. [BG11] also proposed an iris identification system that allows to compare two 2048 bits iris readings in $0.15s$. For a database of size 356 (our smaller setting) this would amount to $53s$.

For face recognition, SciFi [OPJM10] online search runtime is linear in the size of the database: for 100 faces representations search takes $31s$. Erkin et al. [EFG+09]'s system takes roughly $40s$ to search over a dataset of size 320.

Looking ahead to Table 3, our search time for 356 **records is** 11**s.**

Recent work [UCK+21, GRS22, CFR23] has shown how to extend PSI to the setting where one considers items a match if their distance is small. A naive use of fuzzy PSI would tell the client if the biometric exists in the database, but not which item it matches. This issue can be solved using labelled PSI [CHLR18] which associates a label with each set value $x \in X$. This is the approach used by Uzun et al. [UCK+21]. Uzun et al. [UCK+21] evaluate on the face biometric. At a technical level, their fuzzy PSI is similar to our approach, they store LSH outputs in the set X and encrypt each item before sending it to the server. However, they perform a threshold version of labeled PSI, only returning a record if there are enough matches. One match is required in our approach. Their approach is bandwidth efficient but requires fully homomorphic encryption [Gen09] to perform the more complex matching. Uzun et al's scheme does not allow the client to learn about non-matching records which is not a goal of our work. Their scheme only requires $\beta = 64$ due to preprocessing biometric data to reduce noise, such techniques can be applied in our setting but are not standard when evaluating schemes in the biometric literature.

3 Preliminaries

Let λ be the security parameter throughout the paper. We use $\mathtt{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ to denote unspecified functions that are polynomial and negligible in λ, respectively. All definitions are indexed by λ but this indexing is omitted for notational clarity. For some $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \cdots, n\}$. Let $x \xleftarrow{\$} S$ denote sampling x uniformly at random from the finite set S. Let $\perp_1, \perp_2, ...$ be a sequence of distinguished unique symbols. These symbols are allowed inputs to algorithms but omitted for clarity.

For interactive protocols Prot between a client $\mathtt{C}$ and a server $\mathtt{S}$ we use notation

$$\begin{pmatrix} o_{\mathtt{C}} \\ o_{\mathtt{S}} \end{pmatrix} \leftarrow \mathsf{Prot} \begin{pmatrix} i_{\mathtt{C}} \\ i_{\mathtt{S}} \end{pmatrix}$$

with $i_{\mathtt{C}}, o_{\mathtt{C}}, i_{\mathtt{S}}, o_{\mathtt{S}}$ denoting the client's and the server's inputs and outputs respectively. Protocols are written from the perspective of the client with underlying interactive protocols indicating the server's role.

Hamming distance is defined as the distance between the bit vectors x and y of length n, $\mathcal{D}(x, y) = |\{i \mid x_i \neq y_i\}|$, and the fractional Hamming distance is $\mathcal{D}(x, y)/n$. For a map M, let $\mathsf{M.Keywords}$ output the set of all stored keywords. *Locality sensitive hashes* (LSH) frequently map similar values to the same hash.

Definition 1 (Locality-sensitive Hashing)**.** *Let $t \in \mathbb{N}$, $c > 1$ and $p_1, p_2 \in [0, 1]$. $\mathcal{H}$ defines a (t, ct, p_1, p_2)-sensitive hash family if for any $x, y \in \{0,1\}^n$, we have:*

1. *If $\mathcal{D}(x, y) \leq t$, $\Pr_{\mathsf{LSH} \in \mathcal{H}}[\mathsf{LSH}(x) = \mathsf{LSH}(y)] \geq 1 - p_1$ and*
2. *If $\mathcal{D}(x, y) \geq ct$, $\Pr_{\mathsf{LSH} \in \mathcal{H}}[\mathsf{LSH}(x) = \mathsf{LSH}(y)] \leq 1 - p_2$.*

For x, y if $\mathcal{D}(x, y) \leq t$ they are said to be near, if $\mathcal{D}(x, y) \geq ct$ they are said to be far.

We use selection of a single random bit as our LSH. For two values x, y the probability this bit will be the same is $1 - \mathcal{D}(x, y)/n$. That is $p_1 \geq t/n$ while $p_2 \leq ct/n$. The error rates p_1, p_2 of an LSH can be increased by randomly sampling several LSHs and checking that they all match, an α-AND. For α-AND composition, a (t, ct, p_1, p_2)-LSH yields a $(t, ct, p_1^\alpha, p_2^\alpha)$-LSH. **For our LSH, this corresponds to randomly selecting α (with replacement) bits of the input.** Similarly, p_1, p_2 can be decreased by randomly sampling several LSHs and checking that at least one of them matches, a β-OR. For β-OR composition, a (t, ct, p_1, p_2)-LSH yields a $(t, ct, 1 - (1 - p_1)^\beta, 1 - (1 - p_2)^\beta)$-LSH.

3.1 The need for many LSHs in biometric proximity search

We focus on the iris biometric using the ND-0405 dataset [PSO+09, BF16] which has an average distance $t/n \approx .21$ using a state-of-the-art feature extractor [AF19]. There are 712 different irises in the ND0405 dataset which is a superset of the NIST Iris Evaluation Challenge [PBF+08]. Half of these records are used for training the used feature extractor [AF18, Ahm20, AF19, ACD+22] which produces features of length 1024. The remaining 356 right irises are suitable for experimentation. Section 5 describes other datasets used in this work. Our discussion applies to other biometrics with substantive noise such as the face. Deng et al. [DGXZ19, Figure 6] present analogous statistics for the face.

If one only uses the AND of α-LSHs and considers a match if one matches a single LSH out of β, this LSH can be seen as the β-OR of the α-AND of LSHs where

$$p_1' = 1 - (1 - p_1^\alpha)^\beta, \qquad \text{and} \qquad p_2' = 1 - (1 - p_2^\alpha)^\beta.$$

Let w_i' be a noisy reading of w_i. When using w_i' as input to search, a true accept is when w_i is returned and the true accept rate (TAR) is the fraction of queries where this happens. The fraction of false accepts (FFA) is the fraction of $\mathcal{DB} \setminus \{w_i\}$ that is returned on average.

If one assumes that $p_1 = t/n = .21$ and $p_2 = .5$ and all items have these average distances, then TAR is $1 - p_1'$ and FFA is $1 - p_2'$. Under this assumption, achieving a TAR of .95 ($p_1' = .05$) and an FFA of .01 ($p_2' = .99$) requires a number of LSHs $65 \leq \beta \leq 80$ for the minimum $\alpha = 13$. For a dataset of size 10^6 if one seeks at most 100 false accepts (that is, FFA of 10^{-4}) this requires $680 \leq \beta \leq 835$ at the minimum $\alpha = 23$.

However, even though the mean distance between readings of the same biometric is $t/n = .21$ there is substantial variance in this distance (see Figure 13(a)), requiring β to be larger as we show in Table 2. Boldyreva and Tang [BT21] tested on datasets with $\beta \leq 10$.

Furthermore, most of the LSHs will return $\perp$. We call an LSH match *good* if it ensures the query results in a true accept and *bad* otherwise. For the ND-0405 dataset,[5] a histogram of the number of good and bad LSH matches is in Figure 2. The average number of total LSH matches is 23.4.

[5]This uses the following experiment:

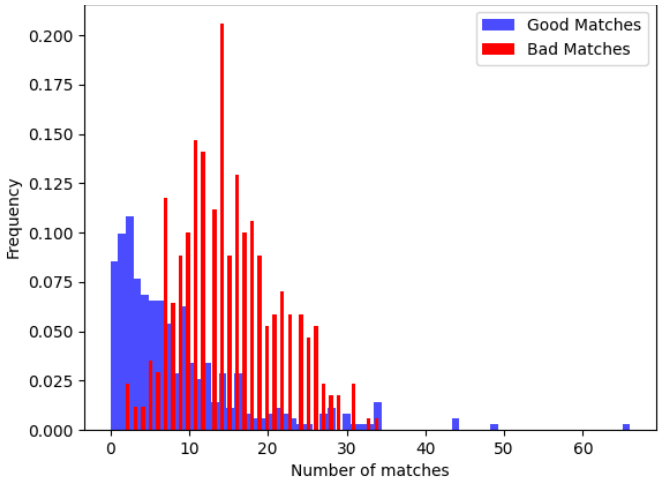


Figure 2: The number of maps returning a value when searching similar values with $\alpha = 15$, $\beta = 225$ trees and 356 records from the ND-0405 dataset.

3.2 Cryptographic Definitions

This work relies on *oblivious RAM* (ORAM) [GO96, Gol87] to achieve zero-leakage. In our constructions, we consider static datasets, ignoring write queries. Our ORAM definition reflects this choice. As we discuss in Section 6, this definition is satisfied by private information retrieval schemes (with appropriate encryption). We discuss other considerations for dynamic data at the end of Section 6.

Definition 2 (Oblivious RAM)**.** *An Oblivious RAM (ORAM) scheme is two protocols,* Setup *and* Access*:*

- $\begin{pmatrix} \sigma, \\ \mathsf{EM} \end{pmatrix} \leftarrow \mathsf{Setup} \begin{pmatrix} 1^\lambda, \mathsf{Mem} \\ 1^\lambda \end{pmatrix}$,
- $\begin{pmatrix} v, \sigma' \\ \mathsf{EM}' \end{pmatrix} \leftarrow \mathsf{Access} \begin{pmatrix} \sigma, i \\ \mathsf{EM} \end{pmatrix}$.

Correctness *Consider the following correctness experiment:*

1. *An adversary $\mathcal{A}$ chooses memory* Mem.
2. *Consider* $\begin{pmatrix} \sigma_0 \\ \mathsf{EM}_0 \end{pmatrix} \leftarrow \mathsf{Setup} \begin{pmatrix} 1^\lambda, \mathsf{Mem} \\ \perp \end{pmatrix}$.
3. *Let* ts_0 *be the server's view of the computation.*
4. *For* $1 \leq i \leq q$*:*
   (a) *Run* $y_i \leftarrow \mathcal{A}(\mathsf{ts}_{i-1})$.
   (b) *Run* $\begin{pmatrix} v_i, \sigma_i, \\ \mathsf{EM}_i \end{pmatrix} \leftarrow \mathsf{Access} \begin{pmatrix} \sigma_{i-1}, y_i \\ \mathsf{EM}_{i-1} \end{pmatrix}$.
   (c) *Let* ts_i *be the server's view of the computation.*

1. Storage of a single feature extracted reading for the right eye for each of the 356 persons in the ND-0405 dataset. Sample $\beta = 225$ LSHs of size $\alpha = 15$.
2. Use the second stored template in the ND-0405 dataset to create a search corpus $w'_1, ..., w'_{356}$.
3. Search for each record w'_i. Record the number of good and bad LSH matches.

$\underline{\mathsf{Real}_{\mathcal{A},q}(1^\lambda)}$:

1. $\mathsf{Mem} \leftarrow \mathcal{A}(1^\lambda)$.
2. Run $\begin{pmatrix} \sigma_0, \\ \mathsf{EM}_0 \end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix} 1^\lambda, \mathsf{Mem} \\ \perp \end{pmatrix}$.
   Let ts_0 denote the server's view of the above computation.
3. For $1 \le i \le q$:
   (a) $y_i \leftarrow \mathcal{A}(\mathsf{ts}_{i-1})$.
   (b) $\begin{pmatrix} v_i, \sigma_i \\ \mathsf{EM}_i \end{pmatrix} \leftarrow \mathsf{Access}\begin{pmatrix} \sigma_{i-1}, y_i \\ \mathsf{EM}_{i-1} \end{pmatrix}$.
   Let ts_i be the server's view of the above computation.
4. Output $(\mathsf{ts}_0, ..., \mathsf{ts}_q)$.

$\underline{\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q}(1^\lambda)}$:

1. Output $(\mathsf{ts}_0, ..., \mathsf{ts}_q) \leftarrow \mathsf{Sim}(q, |\mathsf{Mem}|, 1^\lambda)$.

Figure 3: Definition of ORAM security.

The adversary wins if for some $i, v_i \neq \mathsf{Mem}[y_i]$. The ORAM scheme is correct if the probability of $\mathcal{A}$ winning the game is $\mathit{negl}(\lambda)$.

Security *An ORAM scheme is secure in the semi-honest model if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator* Sim *such that for any PPT distinguisher D we have*

$$\left|\Pr[D(\mathsf{Real}_{\mathcal{A},q}(1^\lambda)) = 1] - \Pr[D(\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q}(1^\lambda)) = 1]\right| \le \mathit{negl}(\lambda)$$

with $\mathsf{Real}_{\mathcal{A},q}$ and $\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q}$ as described in Figure 3.

The above is an adaptive simulation definition of ORAM [GMP16], all of our proofs work naturally for the standard non-adaptive definition.

We define generic oblivious searchable encryption (OSE) and in the rest of the paper, will use specific variants of it.

Definition 3 (Oblivious searchable encryption)**.** *Let $\mathcal{M}$ denote the records space, $\mathcal{Q}$ denote the query space and $\mathcal{R}$ denote the result space. Let $\mathcal{DB} \subseteq \mathcal{M}$ be a database and $y \in \mathcal{Q}$ be a query. For string* param*, let the triple of protocols* $\mathsf{OSE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Search})$ *have the following format:*

- $\begin{pmatrix} \mathsf{sk} \\ \mathsf{pp} \end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix} 1^\lambda, \mathit{param} \\ 1^\lambda, \mathit{param} \end{pmatrix}$,
- $\begin{pmatrix} I_\mathsf{c}, \\ I_\mathsf{s} \end{pmatrix} \leftarrow \mathsf{Encrypt}\begin{pmatrix} \mathsf{sk}, \mathcal{DB} \\ \mathsf{pp} \end{pmatrix}$,
- $\begin{pmatrix} J, I'_\mathsf{c} \\ I'_\mathsf{s} \end{pmatrix} \leftarrow \mathsf{Search}\begin{pmatrix} \mathsf{sk}, y, I_\mathsf{c}, \\ \mathsf{pp}, I_\mathsf{s} \end{pmatrix}$.

OSE *is an oblivious searchable encryption if the following hold:*

Correctness: *The set J is the "same" as the result of the query. The formal definition varies per* OSE *variant we consider and is defined later.*

$\underline{\mathsf{Real}_{\mathcal{A},q,\mathsf{param}}(1^\lambda)}$:

1. Compute $\begin{pmatrix}\mathsf{sk}\\ \mathsf{pp}\end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix}1^\lambda, \mathsf{param}\\ 1^\lambda, \mathsf{param}\end{pmatrix}$.
   Let ts_0 be the server's view.
2. $\mathcal{DB} \leftarrow \mathcal{A}(\mathsf{ts}_0)$.
3. Compute $\begin{pmatrix}I_{\mathsf{c},1},\\ I_{\mathsf{s},1}\end{pmatrix} \leftarrow \mathsf{Encrypt}\begin{pmatrix}\mathsf{sk}, \mathcal{DB}\\ \mathsf{pp}\end{pmatrix}$.
   Let ts_1 be the server's view.
4. For $1 \leq j \leq q$:
   (a) $y^j \leftarrow \mathcal{A}(\mathsf{ts}_j)$.
   (b) Run $\begin{pmatrix}J_j, I_{\mathsf{c},j+1}\\ I_{\mathsf{s},j+1}\end{pmatrix} \leftarrow \mathsf{Search}\begin{pmatrix}\mathsf{sk}, y_j, I_{\mathsf{c},j}\\ \mathsf{pp}, I_{\mathsf{s},j}\end{pmatrix}$.
   Let ts_{j+1} be the server's transcript..
5. Output $(\mathsf{ts}_0, ..., \mathsf{ts}_{q+1})$.

$\underline{\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q,\mathsf{param}}(1^\lambda)}$:

1. Compute $\mathsf{ts}_0 \leftarrow \mathsf{Sim}(1^\lambda, q, \mathsf{param})$.
2. $\mathcal{DB} \leftarrow \mathcal{A}(\mathsf{ts}_0)$.
3. $(\mathsf{ts}_1, ..., \mathsf{ts}_{q+1}) \leftarrow \mathsf{Sim}(\mathcal{L}_{\mathsf{Encrypt}}(\mathcal{DB}))$.
4. Output $(\mathsf{ts}_0, ...\mathsf{ts}_{q+1})$.

Figure 4: Definition of $\mathsf{Exp}^{\mathsf{OSE}}_{\mathsf{ADA\text{-}SIM}}$.

Security: *Let $q = \mathtt{poly}(\lambda)$, $\mathcal{L}_{\mathsf{OSE}} = \{\mathcal{L}_{\mathsf{Encrypt}}, \mathcal{L}_{\mathsf{Search}} = \perp\}$ be the leakage profile of* OSE*'s algorithms. For any PPT adversary $\mathcal{A}$, there exists a simulator* Sim *such that for any PPT distinguisher D we have*

$$\left|\Pr[D(\mathsf{Real}_{\mathcal{A},q}(1^\lambda)) = 1] - \Pr[D(\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q}(1^\lambda)) = 1]\right| \leq \mathit{negl}(\lambda)$$

with $\mathsf{Real}_{\mathcal{A},q}$ and $\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q}$ as described in Figure 4.

Our goal is to build an OSE scheme for proximity queries, we define this particular variant of OSE as follows:

Definition 4 (Oblivious Proximity Search)**.** *Consider Definition 3 with the following specificities:*

- *Let $\mathcal{M} = \mathcal{Q} = \mathcal{R} = \{0,1\}^n$ and $\mathsf{param} = t$.*
- *Consider $\mathcal{DB} = w_1, \cdots, w_\ell$ where each $w_i \in \mathcal{M}$.*

(ϵ, t)-Approximate Correctness: *For all $\mathcal{DB}, y \in \mathcal{Q}$ define $J_{\mathcal{DB},\mathtt{near},y} := \{w_i | \mathcal{D}(w_i, y) \leq t\}$. Let $q = \mathtt{poly}(\lambda)$ and $\epsilon > 0$. For all $\mathcal{DB}$ and all $y_1, ..., y_q$ define:*

$$\begin{pmatrix}\mathsf{sk}\\ \mathsf{pp}\end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix}1^\lambda\\ 1^\lambda\end{pmatrix}, \begin{pmatrix}I^1_{\mathsf{c}},\\ I^1_{\mathsf{s}}\end{pmatrix} \leftarrow \mathsf{Encrypt}\begin{pmatrix}\mathsf{sk}, \mathcal{DB}\\ \mathsf{pp}\end{pmatrix}, \begin{pmatrix}J_j, I_{\mathsf{c},j+1}\\ I_{\mathsf{s},j+1}\end{pmatrix} \leftarrow \mathsf{Search}\begin{pmatrix}\mathsf{sk}, y_j, I_{\mathsf{c},j}\\ \mathsf{pp}, I_{\mathsf{s},j}\end{pmatrix}$$

OSE *is ϵ-approximately correct if $\forall 1 \leq j \leq q$ for all $\mathcal{DB}$*

$$\Pr\left[J^j \supseteq J_{\mathcal{DB},\mathit{near},y}\right] \geq 1 - \epsilon.$$

Definition 4 doesn't limit the number of false matches. Furthermore, in Section 4, we never show that our construction satisfies approximate correctness. Proving formal bounds for an LSH based construction requires many assumptions about the data. Instead, we evaluate approximate correctness using data in Section 7.

Our scheme first finds a list of candidate LSH matches, and then uses an appropriate oblivious map to find the relevant records using the candidate LSH matches. The first stage is called *oblivious membership checking* or OMC.

An OMC can be built from private set intersection (PSI), client storage, and full set retrieval (see Section 4.4). We are **not** offering constructions of OMC as a technical contribution. We benchmark separately using PSI, see discussion in Section 7. In our full implementation we use a local Bloom filter to simplify evaluation.

OMC only handles sets, that is, a collection of values *without repeats*. In our search system, these values will be LSH outputs. It is possible for two distinct LSHs to have the same output. To avoid this, we prepend the LSH id to each LSH output value. For LSH j, the corresponding values to use would then be $\{j \mid\mid \mathsf{LSH}_j(x)\}$. We define OMC as a variant of OSE:

Definition 5 (Oblivious Membership Check)**.** *Let* $\mathsf{OMC} = (\mathsf{Encrypt}, \mathsf{Search})$ *be a pair with stored set size* ρ*, query size* γ*, and result size* δ*, abbreviated* ρ*-ssize,* γ*-qsize, and* δ*-rsize. Consider Definition 3 with the following specificities:*

- *Let* $\mathcal{M} = \mathcal{Q} = \mathcal{R}$ *and* $\mathsf{param} = (\rho, \gamma, \beta)$*.*
- *Consider* $X \subseteq \mathcal{M}$*, such that* $|X| = \gamma$*, and* $Y \subseteq \mathcal{Q}$*, such that* $|Y| = \rho$*. Set* $\mathcal{DB} = X$ *and query* $y = Y$*.*

Correctness: *We use* $\perp_1, \cdots, \perp_\beta$ *to denote a sequence of unique symbols that cannot appear in* X *or* Y*. For all* $X, |X| = \gamma$ *and* $Y, |Y| = \beta$*, let*

$$\begin{pmatrix} EC, \\ ES \end{pmatrix} \leftarrow \mathsf{Encrypt}\begin{pmatrix} 1^\lambda, X \\ 1^\lambda \end{pmatrix}, \begin{pmatrix} I, \\ \perp \end{pmatrix} \leftarrow \mathsf{Search}\begin{pmatrix} EC, Y \\ ES \end{pmatrix}.$$

Then $|I| = \delta$ *and for all* $i \in I$ *such that* $\forall j, i \neq \perp_j$ *it holds that* $\Pr[i \in X \cap Y] \geq 1 - \mathsf{negl}(\lambda)$*.*

Finally, we define the second stage of our system:

Definition 6 (Disjunctive Oblivious Map with Encryption)**.** *Let* $\mathsf{param} = (\beta, \delta)$*, such that* $\beta, \delta \in \mathbb{N}$ *and* $\delta \leq \beta$ *and let* $\mu \in \mathbb{N}$*. Let* $\mathsf{DOMapE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Search})$ *be a triple with* β ***maps****,* μ ***map size****, and* δ ***query size****, abbreviated* $\beta - \mathtt{nmaps}$*,* $\mu - \mathtt{msize}$*, and* $\delta - \mathtt{qsize}$*. Then* DOMapE *is a disjunctive oblivious map with encryption if it satisfies Definition 3 with the following correctness guarantee.*

Correctness: *Let* $\mathcal{M} = \{\mathsf{M}_i \mid \mathsf{M}_i : \mathcal{Q} \leftarrow \mathcal{R}\}$*, where* M_i *denotes a map such that for* $1 \leq i \leq \beta, |\mathsf{M.Keywords}| \leq 2^\mu$*. Set* $\mathcal{DB} = \mathsf{M}_1, \cdots, \mathsf{M}_\beta$*. Let* $\epsilon > 0$*,* $q, \beta, \delta = \mathtt{poly}(\lambda)$ *and* $\delta \leq \beta$*. Let* $\mathsf{param} = (\beta, \mu, \delta)$*. Fix some* $(\{\mathsf{M}_i\}_{i=1}^\beta, \{y^j \in (\mathcal{X} \times [1, \ell])^\delta\}_{j=1}^q)$ *and define for* $1 \leq j \leq q$*:*

- $\begin{pmatrix} \mathsf{sk} \\ \mathsf{pp} \end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix} 1^\lambda, \beta, \delta \\ 1^\lambda \end{pmatrix}$,
- $\begin{pmatrix} \sigma_1, \\ \mathsf{EM}_1 \end{pmatrix} \leftarrow \mathsf{Encrypt}\begin{pmatrix} \mathsf{sk}, \mathsf{M}_1, ..., \mathsf{M}_\beta \\ \mathsf{pp} \end{pmatrix}$,
- $\begin{pmatrix} r_j, \sigma_{j+1}, \\ \mathsf{EM}_{j+1} \end{pmatrix} \leftarrow \mathsf{Search}\begin{pmatrix} \mathsf{sk}, \sigma_j, y_j \\ \mathsf{pp}, \mathsf{EM}_j \end{pmatrix}$.

DOMapE *is correct if there exists a set* $\mathcal{I} \subseteq [2^\mu]$ *where* $|\mathcal{I}| \leq \delta$ *such that :*

$$\Pr\left[(\cup_i r_i^j) \setminus \emptyset \subseteq \cup_{i \in \mathcal{I}} \mathsf{M}_{k_i}\left[x_i^j\right]\right] \geq 1 - \mathtt{ngl}(\lambda).$$

4 Oblivious Proximity Search for Biometrics

This section presents our technical designs, focusing on the design of DOMapE. We describe constructions of OMC in Section 4.4. The most relevant related work is by Boldyreva and Tang [BT21], whose construction is for the approximate k-nearest neighbors search problem. While Boldyreva and Tang discuss two ways of implementing OMapE, one using a tree and the other using a skip list [Pug90], we only present a tree based construction. Similar modifications can be made to the skip list construction. In this work, we consider static data. For static data, B-trees and skip lists are equivalent data structures [LN+96]. However, updates and the resulting performance differ.

Parameter	Meaning
ℓ	Database size
β	`nmaps`, Total number of trees, maps, LSHs, query size to OMC
δ	`qsize`, Result size from OMC, Number of trees to traverse/result size in DOMapE
μ	`msize`, $\log_2$ maximum number of keywords in each map
γ	Server set size in OMC, in composed protocol $\gamma = 2^\mu \cdot \beta$

Table 1: Parameters and their usage across the different schemes.

- $\underline{\mathsf{BIndex}(\mathsf{M}, \mu)}$:
  1. Sort map M using the comparator $\leq$.
     Let $\mathsf{Leaves} = (x_i, \mathsf{M}[x_i])$ be the sorted result.
  2. Pad Leaves to length 2^μ with pairs $(\perp, \perp)$.
  3. Build balanced binary search tree Tr over the values of x_i and for each internal node, attach pointers to its left and right child, LC and RC.
  4. Associate $\mathsf{M}[x_i]$ as data for leaf x_i.
- $\underline{\mathsf{ApplyORAM}(\mathsf{Tr}_1, ..., \mathsf{Tr}_\beta)}$:
  1. For $j \in [0, \mu]$:
     (a) $\mathsf{Level}_j = \perp$.
     (b) For all $\mathsf{Tr}_{i \in [\beta]}$, $\mathsf{Level}_j = \mathsf{Level}_j \mid\mid \mathsf{Level}(\mathsf{Tr}_i, j)$.
     (c) $\begin{pmatrix} \sigma_j, \\ \mathsf{EM}_j \end{pmatrix} \leftarrow \mathsf{ORAM.Setup}\begin{pmatrix} 1^\lambda, \mathsf{Level}_j \\ , \perp \end{pmatrix}$
  2. Denote $\Sigma := \{\sigma_j\}_{j=0}^{\mu}$ and $\mathsf{EM} = \{\mathsf{EM}_j\}_{j=0}^{\mu}$.
- $\underline{\mathsf{Level}(\mathsf{Tr}, j)}$: Return all nodes at level j in Tr.

Figure 5: Build tree index and apply ORAM algorithms.

4.1 Overview of DOMapE design

Recall the unprotected solution for proximity search from the Introduction:

1. Sample β LSHs, $\mathsf{LSH}_1, ..., \mathsf{LSH}_\beta \leftarrow \mathcal{H}$.
2. For $j = 1, ..., \beta$, set $\mathsf{M}_j[v] = \{w_i | \mathsf{LSH}_j(w_i) = v\}$.
3. To search w^*, compute $\mathsf{LSH}_1(w^*), ..., \mathsf{LSH}_\beta(w^*)$, and retrieve $\cup_{j=1}^{\beta} \mathsf{M}_j[\mathsf{LSH}_j(w^*)]$.

The maps consists of $y_i, \{w_i\}$ pairs. The values placed into the map are sorted (lexographically) and used as nodes in a binary tree. Internal nodes are given the value of the minimum value in the right subtree and the location of the two children LC, RC. We show this design in Figure 5. Let $\mathsf{Tr}_1, ..., \mathsf{Tr}_\beta$ be the output of BIndex on maps $\mathsf{M}_1, ..., \mathsf{M}_\beta$ respectively. Each tree is placed in a distinct ORAM. The construction fully traverses every tree Tr_i meaning that there is a constant number of accesses to each ORAM with every search. Let μ_i be the number of elements in M_i, define $\mu = \lceil \log \max_i \mu_i \rceil$, by padding each ORAM to length 2^μ each ORAM receives exactly $\mu + 1$ accesses with each query ($(\mu + 1)\beta$ across the β trees). This is shown in Figure 6 . However, since each level of each map receives a single access per query, one can store each level of the tree in a separate ORAM, the design is shown visually in with each shaded region representing a separate ORAM in Figure 7. In this design with each query, each shaded region sees exactly one query.

Our approach Recall that our goal is a two part construction: First one queries the OMC to find out which $\delta \leq \beta$ LSHs have matches. Then one queries the relevant δ maps M_i to find records. In this new design, one does not query every $\mathsf{M}_1, ..., \mathsf{M}_\beta$. As such, the set of queried maps would be leakage. We merge the ORAMs across maps to

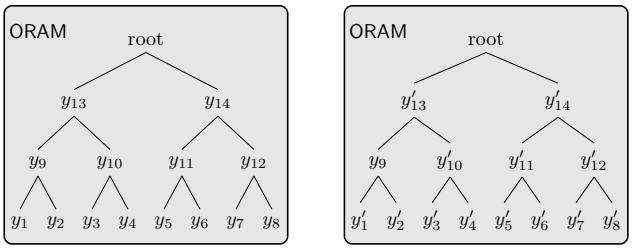


Figure 6: Basic access strategy of Boldyreva and Tang [BT21]. Each shaded region represents data stored together in a single oblivious RAM.

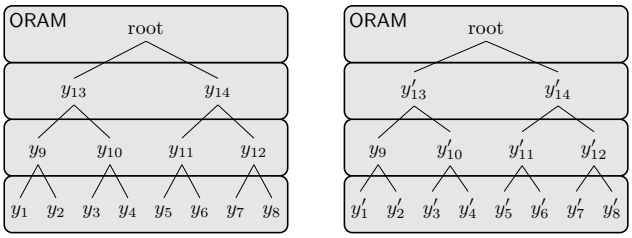


Figure 7: The first optimization to Boldyreva and Tang's construction, where each ORAM is applied per level.

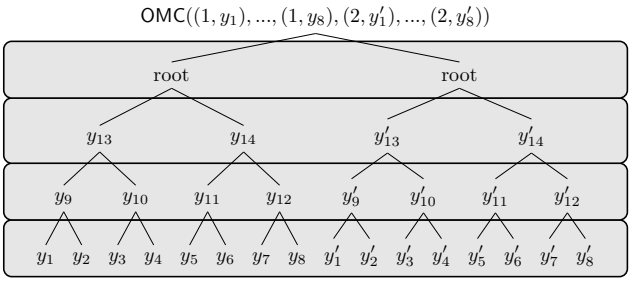


Figure 8: Our design of DOMapE each level across binary trees is stored in a single ORAM.

prevent this. However, we retain a separate ORAM for each level of the maps. This is shown visually in Figure 8 and also described by the ApplyORAM algorithm in Figure 5. This means that each query now makes δ accesses at each ORAM level. There are $\mu + 1$ levels in total resulting in $\delta(\mu + 1)$ ORAM accesses.

4.2 Detailed design of DOMapE

Construction 1. *Let $\mathcal{X}$ and $\mathcal{M}$ be the domain and range of a map, such that elements in $\mathcal{X}$ are comparable with the $\leq$ operator. Define β maps $\mathsf{M}_1, ..., \mathsf{M}_\beta$. Let $\mathsf{ORAM} = (\mathsf{ORAM.Setup}, \mathsf{ORAM.Access})$ be an oblivious RAM as defined in Definition 2, and let ORAM_i denote its instantiation for level $0 \leq i \leq \mu$. Consider the DOMapE construction shown in Figure 9.*

Theorem 1. *For any $\delta, \beta \in \mathbb{N}$ where $\delta \leq \beta$. Construction 1 describes an DOMapE for $\mathcal{L}_{\mathsf{BIndex}}(\mathsf{M}_1, ..., \mathsf{M}_\beta) = \mu = \max_i \lceil \log |\mathsf{M}_i| \rceil$ for $\beta - \mathtt{nmaps}$, $\mu - \mathtt{msize}$, and $\delta - \mathtt{qsize}$.*

Proof. We need to show that for every adversary $\mathcal{A}_{\mathsf{DOMapE}}$ there exists simulator $\mathsf{Sim}_{\mathsf{DOMapE}}$ for DOMapE such that $\mathsf{Real}_{\mathcal{A}_{\mathsf{DOMapE}}}(\lambda) \approx \mathsf{Ideal}_{\mathcal{A},\mathsf{Sim}_{\mathsf{DOMapE}}}(\lambda)$ where $\mathsf{Real}_{\mathcal{A}_{\mathsf{DOMapE}}}$ and $\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim}_{\mathsf{DOMapE}}}$ are defined as in Definition 3.

The $\mathsf{Sim}_{\mathsf{DOMapE}}$ gets as input μ which it uses to initialize that many levels of ORAM simulators. Let $\mathsf{Sim}_{\mathsf{ORAM}_j}$ denote the simulator for the j^{th} level ORAM, $0 \leq j \leq \mu$. We build $\mathsf{Sim}_{\mathsf{DOMapE}}$, the simulator for DOMapE, as follows:

1. Receive inputs $(q, 1^\lambda)$ and $\mathsf{param} = (\beta = \mathtt{nmaps}, \mu = \mathtt{msize}, \delta - \mathtt{qsize})$.

- $\begin{pmatrix}\beta,\nu,\delta\\ \beta,\nu,\delta\end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix}1^\lambda,\beta,\nu,\delta\\ 1^\lambda\end{pmatrix}$
- $\mathsf{Encrypt}\begin{pmatrix}\mathsf{M}_1,\cdots,\mathsf{M}_\beta\\ 1^\lambda\end{pmatrix}$:

  1. Let η_i be the number of elements in M_i and define $\mu = \lceil \log \max_i \eta_i \rceil$.
  2. For $i \in [\beta]$, client sets $\mathsf{Tr}_i \leftarrow \mathsf{BIndex}(\mathsf{M}_i, \mu)$.
  3. Client runs $(\Sigma, \mathsf{EM}) \leftarrow \mathsf{ApplyORAM}(\mathsf{Tr}_1, ..., \mathsf{Tr}_\beta)$.
  4. Server receives EM and client keeps Σ.

- $\mathsf{Search}\begin{pmatrix}\mathsf{sk},\Sigma,y \in (\mathcal{X}\times[1,\ell])^\nu\\ \mathsf{EM}\end{pmatrix}$:

  C does:

  1. Parse $y = (x_1, k_1, ..., x_\nu, k_\nu)$ and $\Sigma = \sigma_1, \cdots, \sigma_\mu$.
  2. Set $\mathsf{Nodes}_1 = ((k_1, 1), ..., (k_\nu, 1))$, $\mathsf{Res} = \perp$.
  3. For $j = [0, \mu - 1]$ and for i in 1 to ν:

$$\begin{pmatrix}\sigma'_j, x', \mathsf{LC}, \mathsf{RC}\\ \mathsf{EM}'_j\end{pmatrix} \leftarrow \mathsf{ORAM.Access}\begin{pmatrix}\sigma_j, \mathsf{Nodes}_j[i],\\ \mathsf{EM}_j\end{pmatrix}.$$

     (a) If $x' \le x_i$, $\mathsf{Nodes}_{j+1} = \mathsf{Nodes}_{j+1} || (k_i, \mathsf{LC})$.
     (b) Else $\mathsf{Nodes}_{j+1} = \mathsf{Nodes}_{j+1} || (k_i, \mathsf{RC})$.

  4. For i in 1 to ν:

$$\begin{pmatrix}\sigma'_\mu, x', \mathsf{M}[x']\\ \mathsf{EM}_i\end{pmatrix} \leftarrow \mathsf{ORAM.Access}\begin{pmatrix}\sigma_\mu, \mathsf{Nodes}_\mu[i],\\ \mathsf{EM}_j\end{pmatrix}.$$

     (a) If $x' = x_i$, $\mathsf{Res} = \mathsf{Res} \cup \mathsf{M}[x']$.

  5. Return Res and $\Sigma' = \sigma'_1, \cdots, \sigma'_\mu$.

Figure 9: DOMAPE Construction. The BIndex algorithm is shown in Figure 5.

2. For $1 \le i \le q \cdot \delta$:
   (a) For $0 \le j \le \mu$, run ORAM_j simulator

$$(\mathsf{ts}^i_{\mathsf{ORAM},j,0}, \cdots, \mathsf{ts}^i_{\mathsf{ORAM},j,\beta}) \leftarrow \mathsf{Sim}_{\mathsf{ORAM}_j}(\beta, |\mathsf{Level}_j|, 1^\lambda).$$

   (b) Set $\mathsf{ts}^{i+1} = (\mathsf{ts}^i_{\mathsf{ORAM},0,1}, \cdots, \mathsf{ts}^i_{\mathsf{ORAM},\mu,\beta})$.
3. Set $\mathsf{ts}^0 = \perp$ and $\mathsf{ts}^1 = (\mathsf{ts}^1_{\mathsf{ORAM},0,0}, \cdots, \mathsf{ts}^1_{\mathsf{ORAM},\mu,0})$.
4. Return $(\mathsf{ts}^0, \cdots, \mathsf{ts}^{q\cdot\delta+1})$.

We then use a hybrid argument where at each step, we replace an ORAM by its corresponding simulator. We obtain the following games

- *Game* 0: $\mathsf{ORAM}_0, \cdots, \mathsf{ORAM}_\mu$,
- *Game* j: $\mathsf{Sim}_{\mathsf{ORAM}_0}, \cdots, \mathsf{Sim}_{\mathsf{ORAM}_j}, \mathsf{ORAM}_{j+1}, \cdots, \mathsf{ORAM}_\mu$,
- *Game* $\mu + 1$: $\mathsf{Sim}_{\mathsf{ORAM}_0}, \cdots, \mathsf{Sim}_{\mathsf{ORAM}_\mu}$,

with $0 \leq j \leq \mu$.

Note that Game 0 contains $\mu + 1$ ORAM instantiations, which corresponds to the real world $\mathsf{Real}_{\mathcal{A}_{\mathsf{DOMapE}}}$. Also note that Game $\mu + 1$, contains $\mu + 1$ ORAM simulators, which is equivalent to $\mathsf{Sim}_{\mathsf{DOMapE}}$ and to the ideal world $\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim}_{\mathsf{DOMapE}}}$. Then for each Game, we show indistinguishability with the previous one by relying on the security of the underlying ORAM_i.

Lemma 1. *For $1 \leq i \leq \mu + 1$, Game i is indistinguishable from Game $i - 1$.*

Proof. By security of ORAM_i, we have $\mathsf{ORAM}_i \approx \mathsf{Sim}_{\mathsf{ORAM}_i}$. Since Game $i - 1$ and Game i only differ at index i, we conclude that these two games are indistinguishable. □

By applying Lemma 1 to each game, we obtain that Games 0 and $\mu + 1$ are indistinguishable, which implies that $\mathsf{Real}_{\mathcal{A}_{\mathsf{DOMapE}}}$ and $\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim}_{\mathsf{DOMapE}}}$ are also indistinguishable and concludes this proof. □

4.3 OSE design

Construction 2. *For a database $\mathcal{DB} = (w_1, ..., w_\ell)$ define $\mu = \lceil \log \ell \rceil$. Fix parameters $\delta, \nu, \beta \in \mathbb{N}$ where $\delta \leq \nu \leq \beta$.*

1. *Let* DOMapE *be a disjunctive oblivious map with encryption with $\beta - \texttt{nmaps}$, $\mu - \texttt{msize}$, $\delta - \texttt{qsize}$, and $\delta - \texttt{rsize}$,*
2. *Let* OMC *be an oblivious membership check with $2^\mu * \beta - \texttt{ssize}$, $\beta - \texttt{qsize}$, and $\delta - \texttt{rsize}$, and*
3. *Let $\mathcal{H}$ be a family of locality sensitive hashes.*

For a database $\mathcal{DB} = (w_1, ..., w_\ell)$, define $\mathsf{OSE} = (\mathsf{OSE.Setup}, \mathsf{OSE.Encrypt}, \mathsf{OSE.Search})$ *as in Figure 10.*

Theorem 2. *Let* DOMapE *and* OMC *be as in Construction 2. Then Construction 2 is an oblivious searchable encryption scheme with leakage $\mathcal{L}_{\mathsf{Encrypt}}(\mathsf{M}_1, ...\mathsf{M}_\beta) = \mu$.*

Proof. Let $\mathsf{Sim}_{\mathsf{OSE}}$ a the simulator for the OSE scheme. This simulator will run as follows:

1. Upon input $(q, \beta, \delta, \mu, 1^\lambda)$, run the simulator for the oblivious membership check with $\mathsf{param} = (\beta * 2^\mu - \texttt{ssize}, \beta - \texttt{qsize}, \delta - \texttt{rsize}$, $(\mathsf{ts}^1_{\mathsf{OMC}}, \cdots, \mathsf{ts}^{q+1}_{\mathsf{OMC}}) \leftarrow \mathsf{Sim}_{\mathsf{OMC}}(q, \gamma = \beta 2^\mu, \beta, \delta)$.
2. Run $(\mathsf{ts}^0_{\mathsf{DOMapE}}, \cdots, \mathsf{ts}^{q+1}_{\mathsf{DOMapE}}) \leftarrow \mathsf{Sim}_{\mathsf{DOMapE}}(q, \mathsf{param}, 1^\lambda)$ with $\mathsf{param} = (\beta - \texttt{nmaps}, \mu - \texttt{msize}, \delta - \texttt{qsize})$.
3. Set $\mathsf{ts}^*_0 = \mathsf{ts}^0_{\mathsf{DOMapE}}$.
4. For $1 \leq i \leq q + 1$, set $\mathsf{ts}^*_i = (\mathsf{ts}^i_{\mathsf{OMC}}, \mathsf{ts}^i_{\mathsf{DOMapE}})$.
5. Output transcripts $\mathsf{ts}^*_0, \cdots, \mathsf{ts}^*_{q+1}$.

We use a hybrid argument to show security of OSE. Consider the following games:

1. $\mathsf{Real}_{\mathsf{OSE}}$,
2. $\mathsf{Sim}^*_{\mathsf{OSE}}$, which runs $\mathsf{Sim}_{\mathsf{OMC}}$ and $\mathsf{Real}_{\mathsf{DOMapE}}$,
3. $\mathsf{Sim}_{\mathsf{OSE}}$, which runs $\mathsf{Sim}_{\mathsf{OMC}}$ and $\mathsf{Sim}_{\mathsf{DOMapE}}$ as described above.

We want to show that game 1 is indistinguishable from game 3.

Lemma 2. *Game 1 and Game 2 are indistinguishable.*

Proof. The difference between games 1 and 2 is $\mathsf{Sim}^*_{\mathsf{OSE}}$'s use of $\mathsf{Sim}_{\mathsf{OMC}}$ instead of $\mathsf{Real}_{\mathsf{OMC}}$. By security of the OMC scheme, game 1 and 2 are indistinguishable. □

Lemma 3. *Game 2 and Game 3 are indistinguishable.*

Proof. The difference between games 2 and 3 is $\mathsf{Sim}_{\mathsf{OSE}}$'s use of $\mathsf{Sim}_{\mathsf{DOMapE}}$ instead of $\mathsf{Real}_{\mathsf{DOMapE}}$. Then by security of the DOMapE scheme, game 2 and 3 are indistinguishable. □

Combining lemmas 2 and 3, we obtain that games 1 and 3 are indistinguishable, which conclude our proof. □

Theorem 2 does not handle correctness. Since there is an overlap between the histograms for real data in Figure 13 one cannot make strong correctness claims. We evaluate correctness empirically in Section 7.

- $\underline{\mathsf{OSE.Setup}\begin{pmatrix}1^\lambda\\1^\lambda\end{pmatrix}:}$

  1. Client samples $\mathsf{LSH}_1, ..., \mathsf{LSH}_\beta \leftarrow \mathcal{H}(1^\lambda)$.
  2. Client runs $\begin{pmatrix}\mathsf{sk}_{\mathsf{OMC}}\\ \perp\end{pmatrix} \leftarrow \mathsf{OMC.Setup}\begin{pmatrix}1^\lambda\\ \perp\end{pmatrix}$ and $\begin{pmatrix}\mathsf{sk}_{\mathsf{DOMapE}}\\ \mathsf{pp}_{\mathsf{DOMapE}}\end{pmatrix} \leftarrow \mathsf{DOMapE.Setup}\begin{pmatrix}1^\lambda\\1^\lambda\end{pmatrix}$.
  3. Denote $\mathsf{pp} = \mathsf{pp}_{\mathsf{DOMapE}}, \mathsf{sk} = (\mathsf{sk}_{\mathsf{OMC}}, \mathsf{sk}_{\mathsf{DOMapE}}, \mathsf{LSH}_1, ..., \mathsf{LSH}_\beta)$.
  4. Client sends pp to server and keeps sk.

- $\underline{\mathsf{OSE.Encrypt}\begin{pmatrix}\mathsf{sk}, \mathcal{DB} = (w_1, ..., w_\ell)\\ \mathsf{pp}_{\mathsf{DOMapE}}\end{pmatrix}:}$

  1. For $1 \le i \le \beta$, client:
     (a) Initializes map M_i.
     (b) For $1 \le j \le \ell$ sets $\mathsf{M}_i[\mathsf{keyword}_{i,j}] = \{w_j | \mathsf{LSH}_i(w_j) = \mathsf{keyword}_{i,j}\}$.
     (c) Adds dummy values to M_i until it is of size ℓ.
  2. Run $\begin{pmatrix}\perp\\ ES\end{pmatrix} \leftarrow \mathsf{OMC.Encrypt}\begin{pmatrix}\mathsf{sk}_{\mathsf{OMC}}, \cup_{i=1}^{\beta}\cup_{j=1}^{\ell} i||\mathsf{keyword}_{i,j}\\ 1^\lambda\end{pmatrix}$.
  3. Run $\begin{pmatrix}\sigma,\\ \mathsf{EM}\end{pmatrix} \leftarrow \mathsf{DOMapE.Encrypt}\begin{pmatrix}\mathsf{sk}, \mathsf{M}_1, ..., \mathsf{M}_\beta\\ \mathsf{pp}_{\mathsf{DOMapE}}\end{pmatrix}$.
  4. Denote $I_{\mathsf{c}} = \sigma$ and $I_{\mathsf{s}} = (ES, \mathsf{EM})$. Client sends I_{s} to server and keeps I_{c}.

- $\underline{\mathsf{OSE.Search}\begin{pmatrix}\mathsf{sk}, y, I_{\mathsf{c}}\\ \mathsf{EM}\end{pmatrix}:}$

  1. Client creates OMC set $EC = (1||\mathsf{LSH}_1(y), \cdots, \beta||\mathsf{LSH}_\beta(y))$.
  2. Run $\begin{pmatrix}\mathtt{Res}_{\mathsf{OMC}}\\ \perp\end{pmatrix} \leftarrow \mathsf{OMC.Search}\begin{pmatrix}\mathsf{sk}_{\mathsf{OMC}}, EC\\ ES\end{pmatrix}$.
     $$\begin{pmatrix}r, \sigma'\\ \mathsf{EM}'\end{pmatrix} \leftarrow \mathsf{DOMapE.Search}\begin{pmatrix}\mathsf{sk}_{\mathsf{DOMapE}}, \sigma, \mathtt{Res}_{\mathsf{OMC}}\\ \mathsf{EM}\end{pmatrix}.$$
  3. Denote $J = (\cup_{i=1}^{\delta} r_i) \setminus \perp, I'_{\mathsf{c}} = \sigma'$ and $I'_{\mathsf{s}} = (ES, \mathsf{EM}')$. Client receives J, I'_{c} and server receives I'_{s}.

Figure 10: OSE construction from OMC, DOMapE and LSH.

4.4 Oblivious membership check constructions

We discuss options to implement OMC. We briefly cover approaches based on Bloom filter lookups. In Section 4.4.2, we describe how to build OMC from private set intersection. This is the tool that we use for microbenchmarks. In our implementation, we use a Bloom filter to emulate an OMC.

4.4.1 Oblivious Bloom Filter Lookups

The client's set can be stored in a Bloom filter [Blo70] which is then stored on the server in an ORAM. The client will request the relevant bits from the ORAM. This prevents the client from having to store the entire Bloom Filter on their side, but requires them to request multiple ORAM accesses to query the relevant bits.

BlindSEER [PKV+14, FVK+15] built a tree of encrypted Bloom filters for general Boolean search. Search of each

node uses Garbled circuits to decide whether to proceed to children. One can use a single level of their tree as an OMC as long as only the client learns the response. This requires some modification as their system was optimized for circuits that output a bit, we would need the set of matching locations. Their system was evaluated on datasets with 10^8 records [FMC+15].

4.4.2 Building OMC from PSI and pseudorandom permutations

Private set intersection (PSI) [FNP04] is a form of secure multi-party computation where a client and server hold sets Y and X respectively. They run an interactive computation, at the end, the client learns $X \cap Y$. No other information is leaked. Current implementations of PSI depend on one of two tools: oblivious polynomial evaluations (OPE) and oblivious pseudorandom functions (OPRF).[6]

We show how to build OMC from honest-but-curious PSI as follows:

1. At initialization the client applies a pseudorandom permutation (PRP) to each element in the set X.
2. The client sends the set of elements (passed through the PRP) to the server.
3. Later when the client has a set Y, they apply the pseudorandom permutation to each element of Y, and uses the resulting values as their set for the PSI protocol.

In OMC, the simulator learns the size of both sets X, Y, using an ideal PSI, only the size of X is leaked to the server. Both the sizes of X and Y are global parameter, $\beta \cdot 2^{\mu}$ and β respectively. We now formalize the above, re-stating the PSI definition used for VOLE-PSI [RS21, Figure 5].

Definition 7 (Private Set Intersection (PSI))**.** *Let $\mathcal{X}$ denote a set. Let the client hold a set $X \subset \mathcal{X}$ and the server hold a set $Y \subseteq \mathcal{X}$. Consider $\begin{pmatrix} Z, \\ \perp \end{pmatrix} \leftarrow \mathsf{PSI}\begin{pmatrix} X, \\ Y \end{pmatrix}$, the PSI protocol between the client and the server. At the end, the client learns Z and the server learns nothing.*

Correctness: *Correctness is,*

$$\Pr\left[Z \neq X \cap Y \;\middle|\; \begin{pmatrix} Z, \\ \perp \end{pmatrix} \leftarrow \mathsf{PSI}\begin{pmatrix} X, \\ Y \end{pmatrix}\right] \leq \mathit{negl}(\lambda).$$

Security: PSI *is secure if for any PPT adversary $\mathcal{A}$, there exists a simulator* Sim*, such that the distributions $\mathsf{Real}_{\mathcal{A},q}$ and $\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q}$, described in Figure 11, are computationally indistinguishable. VOLE-PSI [RS21] uses a different security definition for PSI, their definition implies ours.*

Remark *We consider PSI where both X and Y are constant size sets so we ignore the case when a party provides too large a set.*

Our construction requires pseudorandom permutations.

Definition 8 (Pseudorandom Permutation (PRP))**.** *Let $F : \{0,1\}^{\lambda} \times \mathcal{X} \to \mathcal{X}$ be an efficient keyed permutation and $\mathcal{F}_n$ denote the set of all permutations on $\mathcal{X}$. F is a pseudorandom permutation if for any PPT adversary $\mathcal{A}$,*

$$\left| \begin{matrix} \Pr\left[\mathcal{A}^{F_k(\cdot)}(1^{\lambda}) = 1 || k \xleftarrow{\$} \{0,1\}^{\lambda}\right] - \\ \Pr\left[\mathcal{A}^{f(\cdot)}(1^{\lambda}) = 1 || f \xleftarrow{\$} \mathcal{F}_n\right] \end{matrix} \right| \leq \mathit{negl}(\lambda)$$

Construction 3 (OMC from PSI and PRP)**.** *Let* PSI *denote a PSI scheme and $F : \{0,1\}^{\lambda} \times \mathcal{X} \to \mathcal{X}$ be a pseudorandom permutation. Let X and Y be sets, such that $X \subseteq \mathcal{X}$ and $Y \subseteq \mathcal{X}$. Then we build* OMC *as in Figure 12.*

[6] Cristofaro et al. [CT09] construct efficient PSI procotols, under the restriction that the server can do some precomputation or the client is weak. Dong et al. [DCW13] present an efficient PSI protocol based on a variant of Bloom Filters called *garbled Bloom Filters*. Dachman-Soled et al. [DSMRY09] present an efficient PSI protocol utilizing secret sharing and Reed-Soloman codes. Malicious secure implementations of PSI utilizing OPE also depend on zero-knowledge proofs to prevent parties from deviating from the protocol.

$\underline{\mathsf{Real}_{\mathcal{A},n_X,n_Y}(1^\lambda)}$:

1. $(X, Y) \leftarrow \mathcal{A}(1^\lambda)$ where $X, Y \subset \mathcal{X}$.
2. If $|X| > n_X$ or $|Y| > n_Y$, return $\perp$.
3. Output ts of $\begin{pmatrix} Z, \\ \perp \end{pmatrix} \leftarrow \mathsf{PSI} \begin{pmatrix} X, \\ Y \end{pmatrix}$.

$\underline{\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},n_X,n_Y}(1^\lambda)}$:

1. $(X, Y) \leftarrow \mathcal{A}(1^\lambda)$ where $X, Y \subset \mathcal{X}$.
2. Output $\mathsf{ts} \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{param})$ where $\mathsf{param} = (|X| = \mathtt{csize}, |Y| = \mathtt{ssize})$.

Figure 11: Definition of PSI security.

Theorem 3. *Let* PSI *be a secure private set intersection scheme and F be a pseudorandom permutation, then Construction 3 describes a secure OMC with $\gamma - \mathtt{ssize}$, $\beta - \mathtt{qsize}$, and $\delta - \mathtt{rsize}$.*

Proof. Correctness is straightforward and follows from correctness of the PSI scheme and the fact that each element in F is a permutation. If there are more than δ elements in the intersection then δ are chosen randomly. However, the correctness guarantee only requires that non-$\perp$ elements returned be in the intersection. Security follows from the security of the PRP (values seen by the server are indistinguishable from random) and the security of the PSI scheme (server learns nothing about EC and $EC \cap ES$). Formally, we build the simulator $\mathsf{Sim}_{\mathsf{OMC}}$ as follows:

1. Receive inputs 1^λ, number of queries q and server's set size γ.
2. For $1 \leq i \leq \gamma$, sample a random value $x_i \xleftarrow{\$} \mathcal{X}$.
3. Define the server set as $ES = \{x_i\}_{i=1}^{\gamma}$.
4. For $1 \leq k \leq q$: run the PSI simulator $\mathsf{ts}^k \leftarrow \mathsf{Sim}_{\mathsf{PSI}}(1^\lambda, \mathsf{param})$ with $\mathsf{param} = (\beta = \mathtt{csize}, \gamma = \mathtt{ssize})$.
5. Output ES and $\mathsf{ts}^1, ..., \mathsf{ts}^q$.

We then use a hybrid argument to show security of OMC. Consider the following games:

1. $\mathsf{Real}_{\mathsf{OMC}}$,
2. Replace the PRP F by a random permutation $f \xleftarrow{\$} \mathcal{F}_n$,
3. $\mathsf{Sim}_{\mathsf{OMC}}$.

Games 1 and 2 are indistinguishable by the security of the PRP scheme, and games 2 and 3 are indistinguishable by the security of the PSI scheme. We note that sampling a random output value is equivalent to sampling a random input value as inputs are guaranteed to be a set. □

5 Datasets

We test and evaluate our implementation on three datasets:

ND-0405 dataset This dataset [PSO+09, BF16] is a superset of the NIST Iris Evaluation Challenge [PBF+08]. It consists of the readings of left and right irises from 356 individuals, each iris having at least 4 distinct readings. We use a state-of-the-art feature extractor called ThirdEye [AF19] to obtain 1024 bits feature vectors from the original iris readings. Since the left irises were used to train the feature extractor, we use the right ones for testing and evaluation. The first reading of each right iris is in the $\mathcal{DB}$; queries come from the remaining readings. The Hamming distance distributions are in Figure 13(a).

- $\mathsf{OMC.Setup}\begin{pmatrix}1^\lambda\\ \perp\end{pmatrix}$:

  1. Client randomly samples a PRP key $\mathsf{sk} \xleftarrow{\$} \{0,1\}^\lambda$.
  2. Client keeps sk.

- $\mathsf{OMC.Encrypt}\begin{pmatrix}\mathsf{sk}, X\\ \perp\end{pmatrix}$:

  1. Client initializes $ES = \emptyset$.
  2. For each $x_i \in X$, client sets $ES = ES \cup F(\mathsf{sk}, x_i)$, where F is a PRP.
  3. Client sends ES to server.

- $\mathsf{OMC.Search}\begin{pmatrix}\mathsf{sk}, Y\\ ES\end{pmatrix}$:

  1. For each $y_j \in Y$, client computes permutation $F(\mathsf{sk}, y_j)$.
  2. Client initializes a map M and inserts $\mathsf{M}[F(\mathsf{sk}, y_j)] = y_j$.
  3. Client sets $EC = \{F(\mathsf{sk}, y_j)\}$.
  4. Run $\begin{pmatrix}EC \cap ES,\\ \perp\end{pmatrix} \leftarrow \mathsf{PSI}\begin{pmatrix}EC,\\ ES\end{pmatrix}$.
  5. Client
     (a) Initializes $\mathtt{Res} = \emptyset$.
     (b) For each $c \in EC \cap ES$, set $\mathtt{Res} = \mathtt{Res} \cup \mathsf{M}[c]$.
     (c) If $|\mathtt{Res}| > \delta$ set $\mathtt{Res}$ to be δ random elements of $\mathtt{Res}$.
     (d) If $|\mathtt{Res}| < \delta$ append elements from $\perp_1, ...$ to make $\mathtt{Res}$ size δ.
     (e) Outputs $\mathtt{Res}$.

Figure 12: Construction of OMC from PSI and PRP.

Synthetic dataset Available irises datasets are of limited size, often no more than a few hundreds irises (356 individuals for ND-0405). Real world systems would store thousands to millions individuals, depending on the application. Our solution is to generate *synthetic* irises templates, that mimic actual ones. As can be seen in Figure 13b, synthetic data same and different distributions are similar to the ND-0405 ones. The details on synthetic data generation are in Appendix A. The high level approach is a generative adversarial network (GAN) [GPAM+20] as in prior approaches on synthetic iris generation [AF20].

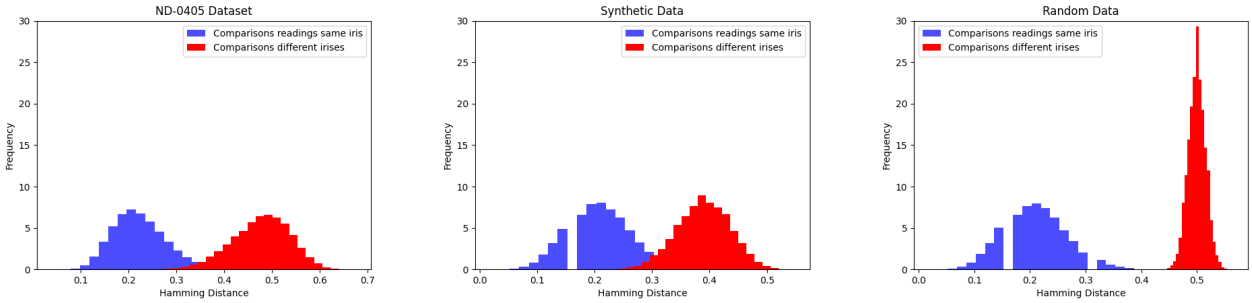
Random dataset This dataset is made from randomly generated 1024 bits vectors. The Hamming distance between two vectors is close to 0.5 with a small variance. This is visible in the red histogram from Figure 13c.

Random and synthetic queries generation Contrary to the ND0405 dataset [BF16], the random and synthetic datasets do not include queries.

We generate queries from a distribution that resembles the one for ND-0405. We use the common observation that like irises comparisons have a distribution close to a binomial across different feature extractors [Dau09,Dau05,SSF19]. From Figure 13a, we extract the mean, $\mu = 0.21$, and the standard deviation, $\sigma = 0.056$. This yields a distribution $B(n, \mu)/n$, the binomial distribution for $n = 53$. This is because for $B(n, \mu)$ it is true that $\sigma^2 = \mu(1-\mu)n$. Thus, by linearity of expectation for $B(n, \mu)/n$ it is true that $\sigma^2 = \mu(1-\mu)/n$, thus one can compute $n = \lceil \mu(1-\mu)/\sigma^2 \rceil = \lceil 52.9 \rceil = 53$.

We can then generate queries for the random and synthetic datasets as follows:

1. First we generate a binomial distribution using the mean and standard deviation of the same iris distribution for the ND-0405.



(a) Histogram of comparisons for ND0405 dataset.

(b) Histogram of comparisons for synthetic dataset.

(c) Histogram of comparisons for random data.

Figure 13: Histograms of Hamming distance between readings of the same iris (in blue) and different irises (in red). Different irises are stored in the database and queries are drawn from a different reading of an iris in the database. The gaps in the synthetic and random blue histograms are caused by the query generation technique used (see paragraph on random and synthetic queries generation) in Section 5.

2. For each feature vector in the dataset, we create a corresponding query by sampling an error fraction from the $\texttt{frac} \leftarrow B(53, 0.21)/53$.

3. We flip the corresponding number of error bits in the original feature vector, that is, $\texttt{frac} * 1024$.

Using this technique, we obtain the same iris distributions (in blue) for synthetic and random data shown in Figures 13b and 13c. Since there are only 54 possible outcomes for a fraction of error bits, this leads to discontinuities in the histograms presented in Figures 13b and 13c.

6 Implementation

We present an open-source implementation of our algorithms including the LSH parameter finding, tree building, and oblivious search [CHF24]. This implementation is in Python 3.9 and uses the PathORAM [SDS+18b] module [Hac18]. Our experiments use a Bloom filter cache on the client as an OMC to focus on the performance of the developed DOMapE. We evaluate an OMC candidate based on PSI in Section 7. Our implementation supports two main conclusions.

One can set a $\delta < \beta$ size of the query to DOMapE that supports a high true accept rate. For a query, we define *bad matches* to be the number of LSH matches that only result in incorrectly returned records; setting δ to be 1 more than the 95% of this value. See Table 2 for a comparison of true accept rate for the setting when $\delta = \beta$ and when $\delta < \beta$. In all analyzed parameters $\delta/\beta < .06$. The value δ is higher for real and synthetic data than for random data; this is due to the larger variance of distances between readings of different irises. This increases the number of false matches.

The two stage DOMapE approach improves search performance. While setup takes several hours, search completes in at most a couple of minutes on all tested parameters. For these parameters we execute at most 1000 ORAM accesses. See further timing discussion in Section 7.2.

We see two mechanisms for achieving better performance as datasets grow:

1. Use ORAM that supports sending multiple queries in parallel (this is a weaker object than a parallel ORAM [WST12, BCP16, CLT16]).

2. Group LSHs into groups that are placed into a single RAM and analyze the required δ for each group. As $\delta/\beta \approx .06$ one can use standard concentration bounds to argue about δ for each group as long as the data and queries are independent of the LSHs.

Dataset Modifications Our implementation does not allow for insertion after the initial building of the tree. Using ORAM one can rebuild the trees using techniques of Wang et al. [WNL+14].

Dataset size	Dataset type	α	β nmaps	# FA	Matches Avg bad	Max bad	Avg good	TAR without OMC	δ qsize	with OMC
356	random	15	630	7.5	7.2	16	32.1	0.98	13	0.94
356	ND	18	850	14.4	16.8	55	21.6	0.95	37	0.89
356	synthetic	18	850	15.5	12.2	37	22.6	0.96	26	0.92
1000	random	18	850	3.6	3.5	11	21.2	0.96	8	0.96
1000	synthetic	19	1000	3.7	22.7	82	20.2	0.95	47	0.96
2500	random	19	1000	5.7	5.6	13	25	0.94	11	0.89
2500	synthetic	21	1200	16	38.4	569	6.6	0.92	56	0.85
5000	random	20	1200	6.9	6.7	14	21.7	0.92	12	0.87
5000	synthetic	22	1300	21.4	44.7	578	6.9	0.91	72	0.83

Table 2: TAR/FAR and the number of matches for random, ND0405, and synthetic datasets of different sizes.

In the static setting, one can use private information retrieval (PIR) [CG97, CKGS98] with encryption. At retrieval, single server computational PIR and PathORAM [SDS+18a] with "large" blocks of size $\Omega(\log^2 N)$ both achieve communication complexity of $O(\log N)$, with N the number of blocks. However, time efficiency would probably suffer from this change. Traditional PIR schemes require work $\Theta(|\mathcal{DB}|)$ on the server. Doubly efficient PIR (DEPIR) [BIM00] preserves the communication efficiency of regular PIR but with $o(|\mathcal{DB}|)$ server work. To achieve this DEPIR relies on a server pre-processing stage which is allowed in our model. While ORAM can be built from symmetric encryption. DEPIR constructions are based on ring LWE [LMW23] or a non-standard secretly permuted Reed-Muller codes assumption [BIPW17, CHR17].

7 Evaluation

Evaluation is split into two parts: 1) parameter analysis and accuracy, and 2) efficiency of the resulting cryptographic construction. Our parameter analysis focuses on the TAR and number of matches. Our efficiency analysis focuses on network roundtrips, storage, and single-threaded computation time.

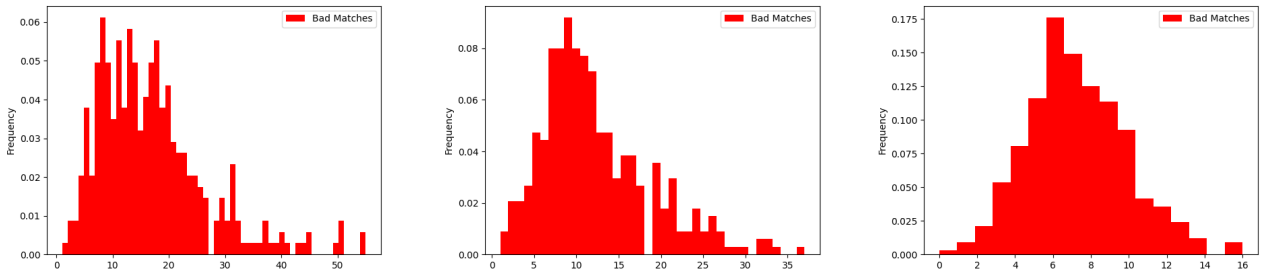
7.1 Accuracy - Parameter analysis

Each experiment is conducted on each dataset. Recall the relevant parameters: α, the length of the extended LSH, $\beta -$ `nmaps`, and $\delta -$ `qsize`.

Finding α, β, δ The first part of the experiment was a manual search across α, β, measuring the TAR and number of bad matches. Selected parameters had TAR of at least 90%. The average number of bad matches was at most 10 for random data and at most 50 for ND and synthetic data. Once α, β were selected we recorded the histogram of bad matches and set δ to be one more than the 95% of this histogram (Fig. 14).

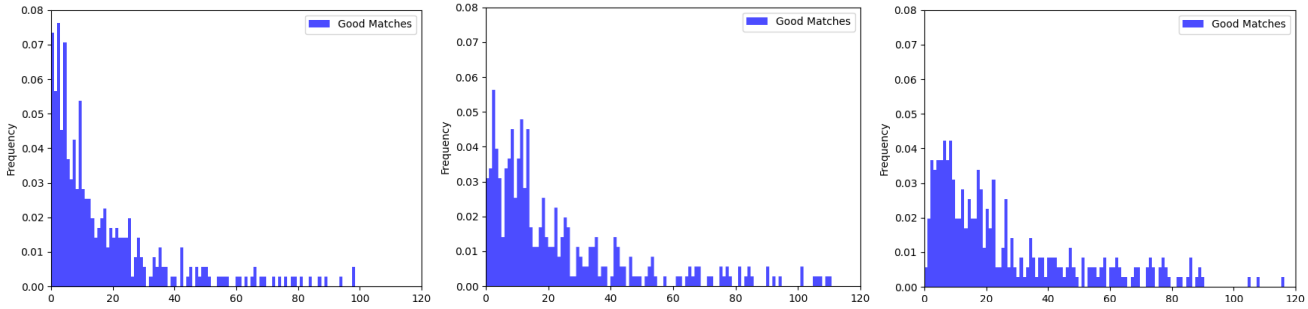
Measuring accuracy We then measured accuracy for a search that queries all β maps and one that only queries δ maps. For these tests, we only measure the TAR to understand the impact of restricting the number of searched values on accuracy.

Discussion In proximity search, high TAR, requires capturing the tail of comparisons between different readings of the same iris (shown in Figure 13). For example, for distance $t = .21n$ and a TAR of .01, Section 3.1 proposed $\beta = 65$ and $\alpha = 13$. Table 2 shows that even for random data, we require $\beta = 630$ and $\alpha = 15$. These parameters increase further on the ND and Synthetic datasets. Parameters vary between random and synthetic data due to a larger variance in the Hamming distance histograms, shown in Figure 13. This is in contrast to the histogram of good matches across datasets in Figure 15 which are consistent across datasets.This leads to an increase in the selected δ. Across dataset sizes, δ for synthetic data is about 5 times δ for random data. The ND and synthetic data statistics align well. This gives some indication that parameters for larger synthetic dataset sizes would yield comparable performance on real irises. Across all parameter settings $\delta/\beta < .06$ validating the overall design.



(a) ND-0405 (b) Synthetic Data (c) Random Data

Figure 14: Number of bad matches across datasets of size 356.



(a) ND-0405 (b) Synthetic Data (c) Random Data

Figure 15: Number of good matches across datasets of size 356.

Restricting to only δ accesses in the DOMapE has an effect on the TAR of the system. Note that the real system does not know which matches are good so even in the case when many matches are bad it is possible for a single good match to be included in the arbitrarily selected δ traversals.The worst degradation of TAR is for synthetic data with 5000 records where TAR drops from .91 to .83.

7.2 Speed - Cryptographic Efficiency

The implementation was tested on a RHEL 7.9 machine with an Intel(R) Xeon(R) CPU E5-2630 v3 with 32 cores at 2.40GHz, 64 GB (4x16GB) 2400MHz DDR4 RDIMM ECC RAM, and Two 2TB SATA 7.2K RPM HDD in Raid1. Results are in Table 3. We did not model network delay in our experiments, because the underlying PathORAM implementation does not batch ORAM requests into a roundtrip.

Storage efficiency Feature vectors are 1024 bit vectors, so 5K irises is 640 KB. The unprotected (same structure as DOMapE but without ORAM) index takes approximately 122.3 MB.[7] This represents a storage increase factor of around 22 between raw data and unprotected index. As shown in Table 3, for our encrypted storage this amounted to 35.6 GB in storage. ORAM increases storage again approximately 291 times. As we discuss in the Conclusion, one can more efficiently pack ORAM blocks using trees with a branching factor > 2.

Time efficiency The time to build the encrypted index is largely dominated by the ORAM setup time so we only report the later (column "O.Init"). For small datasets (356 records), ORAM setup takes between 1 and 4 hours, respectively for random and real data. For larger datasets of size 5000, ORAM setup takes almost 2 days for random data and approximately 11 days for synthetic data. While these timings are not prohibitive yet, they will be for very large datasets. We believe that part of this problem is caching of the large data structures to virtual memory and

[7]Internal node consists of an LSH number, a 22 LSH values, and 2 child identifiers (either the node id or the position of the child in the next ORAM). Leaf nodes consist of a single 32 node identifier.

Dataset size, ℓ	Dataset type	α	β nmaps	δ qsize	# Queries	ORAM Reads	# Roundtrips sequential	# Roundtrips parallel	Time (s) O.Init	Time (s) Search	Time (s) O.Read	Size (GB)
356	random	15	630	13	356	117	334	18	3.8×10^3	11	.051	1.1
356	ND	18	850	37	356	333	666	18	13.8×10^3	39	.062	1.1
356	synthetic	18	850	26	356	234	468	18	12.3×10^3	27	.061	1.1
1000	random	18	850	8	500	80	160	20	11×10^3	17	.122	2.2
1000	synthetic	19	1000	47	500	470	940	20	13×10^3	113	.135	2.2
2500	random	19	1000	11	500	132	262	24	59×10^3	90	.353	8.9
2500	synthetic	21	1200	56	500	672	1344	24	202×10^3	553	.420	17.8
5000	random	20	1200	12	500	156	312	26	166×10^3	235	.757	35.6
5000	synthetic	22	1300	72	500	936	1872	26	902×10^3	1530	.816	35.6

Table 3: Efficiency results. O.Init is time to initialize all ORAMs. O.Read is average read time (across ORAM layers). Search is time per query and includes tree traversals. Size EDB denotes the size of the ORAM files that are stored on the server (OMC storage is ignored since it is much smaller). Sequential number of roundtrips is $2 * \#\text{ORAM Reads}$ and Parallel Rounds trips is $\lceil \log_2 \ell \rceil * 2$. All timing numbers are averaged across the number of queries in # Queries.

the use of a spinning disk hard drive that perform poorly with ORAM workloads [WST12, Section 2.3]. To check this hypothesis, we ran the synth dataset of size 356 on a M1 Mac mini desktop with 16GB of memory and a 2TB SSD, and observed a 4 fold reduction in ORAM setup time.

Search time (per query) requires < 1 min for datasets of size 356, reaching a few minutes for larger datasets (approximately 4 minutes for 5000 random and 26 minutes for 5000 synthetic). Although these timings for larger datasets are not ideal, they are a vast improvement compared to previous works (see discussion in Introduction.). Furthermore, if one had to search all β trees, search time would increase by a factor of at least $1/.06 \approx 16$.

Network Round Trips We report on two figures, the number of round trips using a purely sequential PathORAM implementation and the number of roundtrips if one is able to fully batch all requests at the same level. For the largest synthetic parameter sizes, if one assumes a fast network with $60ms$ responses and unbounded bandwidth then network delays result in 1.56 seconds in parallel rounds trips, but slower $1s$ responses results in 26 seconds. If one assumes sequential round trips and $60ms$ responses the network delay alone is $112s$. For comparison, we note that our local ORAM read operation took $.849s$ on this dataset.

Impact of ORAM implementation Our underlying ORAM module is not parallel [SDS+18a]. Since ORAM accesses dominate the timings, using a better optimized implementation would positively affect efficiency. Maas et al. [MLS+13] proposed a secure processor relying on Path ORAM to obfuscate its memory access trace. They built an hardware implementation of their approach on a commercially available FPGA-based server. Their evaluation shows ORAM accesses take $30\mu s$ for an ORAM size of 4 GB and block size of 4 kB. As comparison, in our implementation an ORAM access takes 62 ms for a total size of ORAM files of 1.1 GB. Recall that our scheme uses one ORAM per tree level. This is 2000 times slower than in Maas et al.'s work.

Many existing ORAM schemes including PathORAM naturally supported batched read/write operations where the client keeps a larger stash. In the case of PathORAM, the client repeatedly reads and writes a "random" path on a tree. One can naturally perform all reads first and then perform all writes, simulating the intermediate storage that would be held by the server. Parallel ORAM is a more complex solution when the reads come from different clients [WST12].

Evaluation of OMC implementation using private set intersection On the same hardware as the rest of the evaluation we deployed the VolePSI implementation [RS21]. We deployed this with a server set of size 6.5 million items and a client set of 1300 items. This corresponds to the largest set of parameters in Table 3. VolePSI is based on OT extension and requires a setup phase. We benchmarked 32 PSI iterations with the first taking 766ms and the rest taking 2ms of computation. We note that VolePSI requires 7 messages of communication. These results justify the focus on the design of DOMapE.

8 Conclusion

This work presents *Private Eyes* the first zero-leakage biometric database. Our system is tested with response times in minutes on databases of thousands of irises. Our construction combines LSHs and oblivious maps. The unique aspect of our design is the recognition and mitigation of the cryptographic inefficiencies caused by the high noise in biometric data. In particular, we use the statistics of biometric data to create our two-stage approach which filters which LSHs to query using a lighter-weight membership checking primitive before the heavy-weight oblivious map.

We used binary trees, one can use trees with a higher branching factor (or skiplists as in [BC14]) to reduce the number of ORAM lookups. Ideally, each node would correspond to a single ORAM block which is commonly a multiple of 256 bytes. Our current estimate is that internal nodes account for ≤ 128 bits of storage out of the 256 byte block size. As such, one could make the tree into a 18-ary tree (32 bits for LSH number, 32 bits for left most child, and $22 + 32$ bits for each additional comparison node). This would reduce the depth of the trees and required number of round trips by a factor of $\log_2 18 \approx 4$.

Acknowledgements

The authors thank the anonymous reviewers for their help in improving the manuscript. The authors also thank Mayank Varia for helpful discussions and Mike Roseluk for helping provide options for the evaluation of private set intersection protocols. J.H. is supposed by NSF Award #1950600. C.C. is supported by NSF Award #2141033. L.D. is supported by the Harriott Fellowship. S.A. is supported by a fellowship from Synchrony Inc. and the State of Connecticut. B.F. is supported by NSF Awards # 2141033 and 2232813 and ONR Grant N00014-19-1-2327.

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19020700008. This material is based upon work supported by the Defense Advanced Research Projects Agency, DARPA, under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government.

The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

[ACD+22] Sohaib Ahmad, Chloe Cachet, Luke Demarest, Benjamin Fuller, and Ariel Hamlin. Proximity searchable encryption for the iris biometric. In *AsiaCCS*, 2022. `https://ia.cr/2020/1174`.

[AF18] Sohaib Ahmad and Benjamin Fuller. Unconstrained iris segmentation using convolutional neural networks. In *Asian Conference on Computer Vision*, pages 450–466. Springer, 2018.

[AF19] Sohaib Ahmad and Benjamin Fuller. Thirdeye: Triplet-based iris recognition without normalization. In *IEEE International Conference on Biometrics: Theory, Applications and Systems*, 2019.

[AF20] Sohaib Ahmad and Benjamin Fuller. Resist: Reconstruction of irises from templates. In *IJCB*, pages 1–10. IEEE, 2020.

[Ahm20] Sohaib Ahmad. Sohaib ahmad github, 2020. Accessed: 2020-07-23.

[BBC+10] Mauro Barni, Tiziano Bianchi, Dario Catalano, Mario Di Raimondo, Ruggero Donida Labati, Pierluigi Failla, Dario Fiore, Riccardo Lazzeretti, Vincenzo Piuri, Fabio Scotti, and Alessandro Piva. Privacy-preserving fingercode authentication. In *Proceedings of the 12th ACM Workshop on Multimedia and Security*, page 231–240, New York, NY, USA, 2010. Association for Computing Machinery.

[BBOH96] Christopher M Brislawn, Jonathan N Bradley, Remigius J Onyshczak, and Tom Hopper. The FBI compression standard for digitized fingerprint images. In *Proc. SPIE*, volume 2847, pages 344–355, 1996.

[BC14] Alexandra Boldyreva and Nathan Chenette. Efficient fuzzy search on encrypted data. In *International Workshop on Fast Software Encryption*, pages 613–633. Springer, 2014.

[BCP16] Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious parallel ram and applications. In *Theory of Cryptography Conference*, pages 175–204. Springer, 2016.

[BF16] Kevin W Bowyer and Patrick J Flynn. The ND-IRIS-0405 iris image dataset. *arXiv preprint arXiv:1606.04853*, 2016.

[BG11] Marina Blanton and Paolo Gasti. Secure and efficient protocols for iris and fingerprint identification. In Vijay Atluri and Claudia Diaz, editors, *Computer Security – ESORICS 2011*, pages 190–209, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[BHJP14] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR)*, 47(2):1–51, 2014.

[BIM00] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: Pir with preprocessing. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, pages 55–73, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 662–693, Cham, 2017. Springer International Publishing.

[Blo70] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[BT21] Alexandra Boldyreva and Tianxin Tang. Privacy-preserving approximate k-nearest-neighbors search that hides access, query and volume patterns. *PoPETS*, 2021.

[CAD+23] Chloe Cachet, Sohaib Ahmad, Luke Demarest, Serena Riback, Ariel Hamlin, and Benjamin Fuller. Multi random projection inner product encryption, applications to proximity searchable encryption for the iris biometric. *Information and Computation*, 293:105059, 2023.

[CCD+20] Hao Chen, Ilaria Chillotti, Yihe Dong, Oxana Poburinnaya, Ilya Razenshteyn, and M Sadegh Riazi. Sanns: Scaling up secure approximate {k-Nearest} neighbors search. In *USENIX Security*, pages 2111–2128, 2020.

[CFR23] Anrin Chakraborti, Giulia Fanti, and Michael K Reiter. {Distance-Aware} private set intersection. In *USENIX Security*, pages 319–336, 2023.

[CG97] Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). In *STOC*, STOC '97, page 304–313, New York, NY, USA, 1997. Association for Computing Machinery.

[CGKO11] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19:895–934, 01 2011.

[CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *CCS*, pages 668–679, 2015.

[CHF24] Chloe Cachet, Julie Ha, and Benjamin Fuller. An implementation of oblivious proximity searchable encryption. `https://github.com/hajulie/searchable_biometric`, 2024.

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled psi from fully homomorphic encryption with malicious security. In *CCS*, pages 1223–1237, 2018.

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 694–726, Cham, 2017. Springer International Publishing.

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, nov 1998.

[CLT16] Binyi Chen, Huijia Lin, and Stefano Tessaro. Oblivious parallel ram: improved efficiency and generic constructions. In *Theory of Cryptography Conference*, pages 205–234. Springer, 2016.

[CT09] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear computational and bandwidth complexity. Cryptology ePrint Archive, Paper 2009/491, 2009. `https://eprint.iacr.org/2009/491`.

[CWD+18] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[Dau05] John Daugman. Results from 200 billion iris cross-comparisons. 01 2005.

[Dau09] John Daugman. How iris recognition works. In *The essential guide to image processing*, pages 715–739. Elsevier, 2009.

[Dau14] John Daugman. 600 million citizens of India are now enrolled with biometric id,". *SPIE newsroom*, 7, 2014.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 789–800, 2013.

[DGXZ19] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, pages 4690–4699, 2019.

[DSMRY09] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Efficient robust private set intersection. In *International Conference on Applied Cryptography and Network Security*, pages 125–142. Springer, 2009.

[EFG+09] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In Ian Goldberg and Mikhail J. Atallah, editors, *Privacy Enhancing Technologies*, pages 235–253, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[FMC+15] Benjamin Fuller, Darby Mitchell, Robert Cunningham, Uri Blumenthal, Patrick Cable, Ariel Hamlin, Lauren Milechin, Mark Rabe, Nabil Schear, Richard Shay, et al. Security and privacy assurance research (spar) pilot final report. Technical report, MIT Lincoln Laboratory Lexington United States, 2015.

[FMC+20] Francesca Falzon, Evangelia Anna Markatou, David Cash, Adam Rivkin, Jesse Stern, and Roberto Tamassia. Full database reconstruction in two dimensions. In *CCS*, pages 443–460, 2020.

[FNP04] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.

[Fou] Electronic Frontier Foundation. Mandatory national ids and biometric databases.

[FP22] Francesca Falzon and Kenneth G. Paterson. An efficient query recovery attack against a graph encryption scheme. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian D. Jensen, and Weizhi Meng, editors, *Computer Security – ESORICS 2022*, pages 325–345, Cham, 2022. Springer International Publishing.

[FVK+15] Ben A Fisch, Binh Vo, Fernando Krell, Abishek Kumarasubramanian, Vladimir Kolesnikov, Tal Malkin, and Steven M Bellovin. Malicious-client security in blind seer: a scalable private dbms. In *2015 IEEE Symposium on Security and Privacy*, pages 395–410. IEEE, 2015.

[FVY+17] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K Cunningham. SoK: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 172–191. IEEE, 2017.

[FWG+16] Zhangjie Fu, Xinle Wu, Chaowen Guan, Xingming Sun, and Kui Ren. Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics and Security*, 11(12):2706–2716, 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GLMP18] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 315–331, 2018.

[GMP16] Sanjam Garg, Payman Mohassel, and Charalampos Papamanthou. Tworam: efficient oblivious ram in two rounds with applications to searchable encryption. In *Annual International Cryptology Conference*, pages 563–592. Springer, 2016.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *Journal of the ACM (JACM)*, 43(3):431–473, 1996.

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 182–194, 1987.

[GPAM+20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[GPP23] Zichen Gui, Kenneth G. Paterson, and Sikhar Patranabis. Rethinking searchable symmetric encryption. In *IEEE Security and Privacy*, 2023. `https://eprint.iacr.org/2021/879`.

[GRGB+12] Javier Galbally, Arun Ross, Marta Gomez-Barrero, Julian Fierrez, and Javier Ortega-Garcia. From the iriscode to the iris: A new vulnerability of iris recognition systems. *Black Hat Briefings USA*, 1, 2012.

[GRS22] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Structure-aware private set intersection, with applications to fuzzy matching. In *Annual International Cryptology Conference*, pages 323–352. Springer, 2022.

[GSB+17] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 655–672. IEEE, 2017.

[Hac18] Gabriel Hackebeil. Path oram python module, 2018.

[HWKL18] Yi Huang, Adams Kong Wai-Kin, and Kwok-Yan Lam. From the perspective of CNN to adversarial iris images. In *BTAS*, pages 1–10. IEEE, 2018.

[IKK12] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In *NDSS*, volume 20, page 12. Citeseer, 2012.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[JM19] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. In *ICLR*, 2019.

[KB15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[KE19] Evgenios M Kornaropoulos and Petros Efstathopoulos. The case of adversarial inputs for secure similarity approximation protocols. In *EuroS&P*, pages 247–262. IEEE, 2019.

[KIK12] Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. Efficient similarity search over encrypted data. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1156–1167. IEEE, 2012.

[KKM+22] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Thomas Schneider, Amos Treiber, and Michael Yonli. Sok: Cryptanalysis of encrypted search with leaker–a framework for leakage attack evaluation on real-world data. In *EuroS&P*, pages 90–108. IEEE, 2022.

[KKNO16] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1329–1340, 2016.

[KPT19a] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Data recovery on encrypted databases with k-nearest neighbor query leakage. In *IEEE S&P*, pages 1033–1050. IEEE, 2019.

[KPT19b] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution. Cryptology ePrint Archive, Report 2019/441, 2019.

[KPT20] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. The state of the uniform: attacks on encrypted databases beyond the uniform query distribution. In *IEEE S&P*, pages 1223–1240. IEEE, 2020.

[KYV+17] Naman Kohli, Daksha Yadav, Mayank Vatsa, Richa Singh, and Afzel Noore. Synthetic iris presentation attack using idcgan. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 674–680. IEEE, 2017.

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic ram computation from ring lwe. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 595–608, New York, NY, USA, 2023. Association for Computing Machinery.

[LMWY20] Kasper Green Larsen, Tal Malkin, Omri Weinstein, and Kevin Yeo. Lower bounds for oblivious near-neighbor search. In *SODA*, pages 1116–1134. SIAM, 2020.

[LN+96] Michael Lamoureux, Bradford Nickerson, et al. On the equivalence of b-trees and deterministic skip list. 1996.

[LPW+20] Qin Liu, Yu Peng, Jie Wu, Tian Wang, and Guojun Wang. Secure multi-keyword fuzzy searches with enhanced service quality in cloud computing. *IEEE Transactions on Network and Service Management*, 18(2):2046–2062, 2020.

[LWW+10] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[MCYJ18] Guangcan Mai, Kai Cao, Pong C Yuen, and Anil K Jain. On the reconstruction of face images from deep face templates. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1188–1202, 2018.

[MHN13] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[MLS+13] Martin Maas, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Krste Asanovic, John Kubiatowicz, and Dawn Song. Phantom: Practical oblivious computation in a secure processor. In *CCS*, CCS '13, page 311–324, New York, NY, USA, 2013. Association for Computing Machinery.

[MT19] Evangelia Anna Markatou and Roberto Tamassia. Full database reconstruction with access and search pattern leakage. In *International Conference on Information Security*, pages 25–43. Springer, 2019.

[OPJM10] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, and Boaz Moskovich. SCiFI - a system for secure face identification. In *2010 IEEE Symposium on Security and Privacy*, pages 239–254, 2010.

[PBF+08] P Jonathon Phillips, Kevin W Bowyer, Patrick J Flynn, Xiaomei Liu, and W Todd Scruggs. The iris challenge evaluation 2005. In *BTAS*, pages 1–8. IEEE, 2008.

[PKV+14] Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos Keromytis, and Steve Bellovin. Blind seer: A scalable private dbms. In *IEEE S&P*, pages 359–374. IEEE, 2014.

[PSO+09] P Jonathon Phillips, W Todd Scruggs, Alice J O'Toole, Patrick J Flynn, Kevin W Bowyer, Cathy L Schott, and Matthew Sharpe. FRVT 2006 and ICE 2006 large-scale experimental results. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):831–846, 2009.

[Pug90] William Pugh. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.

[RS21] Peter Rindal and Phillipp Schoppmann. Vole-psi: Fast oprf and circuit-psi from vector-ole. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 901–930, Cham, 2021. Springer International Publishing.

[SDDN19] Sobhan Soleymani, Ali Dabouei, Jeremy Dawson, and Nasser M Nasrabadi. Adversarial examples to fool iris recognition systems. In *2019 International Conference on Biometrics (ICB)*, pages 1–8. IEEE, 2019.

[SDS+18a] Emil Stefanov, Marten Van Dijk, Elaine Shi, T.-H. Hubert Chan, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: An extremely simple oblivious ram protocol. *J. ACM*, 65(4), apr 2018.

[SDS+18b] Emil Stefanov, Marten Van Dijk, Elaine Shi, T-H Hubert Chan, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: an extremely simple oblivious ram protocol. *Journal of the ACM (JACM)*, 65(4):1–26, 2018.

[SSF19] Sailesh Simhadri, James Steel, and Benjamin Fuller. Cryptographic authentication from the iris. In *International Conference on Information Security*, pages 465–485. Springer, 2019.

[SWP00] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE S&P*, pages 44–55. IEEE, 2000.

[UCK+21] Erkam Uzun, Simon P Chung, Vladimir Kolesnikov, Alexandra Boldyreva, and Wenke Lee. Fuzzy labeled private set intersection with applications to private real-time biometric search. In *USENIX Security*, pages 911–928, 2021.

[VS11] Shreyas Venugopalan and Marios Savvides. How to generate spoofed irises from an iris code template. *IEEE Transactions on Information Forensics and Security*, 6(2):385–395, 2011.

[WLD+17] Guofeng Wang, Chuanyi Liu, Yingfei Dong, Hezhong Pan, Peiyi Han, and Binxing Fang. Query recovery attacks on searchable encryption based on partial knowledge. In *International Conference on Security and Privacy in Communication Systems*, pages 530–549. Springer, 2017.

[WMT+13] Jianfeng Wang, Hua Ma, Qiang Tang, Jin Li, Hui Zhu, Siqi Ma, and Xiaofeng Chen. Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. *Comput. Sci. Inf. Syst.*, 10(2):667–684, 2013.

[WNL+14] Xiao Shaun Wang, Kartik Nayak, Chang Liu, TH Hubert Chan, Elaine Shi, Emil Stefanov, and Yan Huang. Oblivious data structures. In *CCS*, pages 215–226, 2014.

[WST12] Peter Williams, Radu Sion, and Alin Tomescu. Privatefs: A parallel oblivious file system. In *CCS*, pages 977–988, 2012.

Layer	OutputSize	Kernels
Input	1x128	-
G1	64	64
G2	128	128
G3	64	64
G4	128	128
G5	1024	1024
D1	16	16
D2	128	128
D3	1	1

Table 4: Generator architecture

[WYLH14] Bing Wang, Shucheng Yu, Wenjing Lou, and Y Thomas Hou. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *IEEE INFOCOM 2014-IEEE conference on computer communications*, pages 2112–2120. IEEE, 2014.

[YCR19] Shivangi Yadav, Cunjian Chen, and Arun Ross. Synthesizing iris images using RaSGAN with application in presentation attack detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

A Synthetic Data Generation

We now describe the neural network used to produce our synthetic templates. Our synthetic templates are built using a generative adversarial network or GAN. A GAN trains two networks in competition, a generator which should produce synthetic irises and a discriminator which classifies irises as real or synthetic. Yadav et al. [YCR19] uses RaSGAN (relativistic average standard GAN) [JM19] to generate synthetic irises for the purpose of studying their effects on presentation attack detection (PAD) algorithms. Irises from the RaSGAN perform well against PAD and follow real iris statistics well. Kohli et al. [KYV$^+$17] use the DCGAN architecture to generate synthetic irises. Synthetic irises can be viewed as irises that must closely resemble bonafide irises as discussed in [YCR19, KYV$^+$17].

Our synthetic data generator is also trained using the ND-0405 dataset [BF16]. We follow the approach of RESIST [AF20] which takes inspiration from synthetic data generation to invert iris templates into realistic looking images.

We denote the network as `SYNTH`. In a GAN formulation, noise is sampled from a multivariate normal distribution ($\mathbb{P}_z$) with a mean of 0 and a variance of 1. The generator converts this noise vector into a synthetic template. Let $\mathbb{P}_y$ denote the distribution of real templates and $\mathbb{P}_{\hat{y}}$ denote the distribution of synthetic templates. We use a recently proposed relativistic average discriminator [JM19] as our discriminator. To build up to the relativistic discriminator we first start with the original GAN loss functions:

$$L(D) = -\mathbb{E}_{y\sim\mathbb{P}_y}[\log(D(y))] - \mathbb{E}_{\hat{y}\sim\mathbb{P}_{\hat{y}}}[\log(1 - D(\hat{y}))]$$
$$L(G) = -\mathbb{E}_{\hat{y}\sim\mathbb{P}_{\hat{y}}}[\log(D(\hat{y}))].$$

$L(D)$ is called the discriminator loss and $L(G)$ is called the generator loss, y is an actual template and $\hat{y}$ is a synthetic template generated by the generator.

The $L(D)$ and $L(G)$ losses are minimized using gradient descent. The generator and discriminator play a zero sum game. The generator weights are updated based on how good its synthetic irises are while the discriminator weights are updated on how well it differentiates between real and synthetic templates.

The last layer of the generator is the hyberbolic tangent (tanh) with output ranging from -1 to 1, this is done to generate binary synthetic templates by substituting 0,1 with -1,1. The real templates are also converted to -1,1 for conformity.

Architecture `SYNTH` architecture is a small neural network having only dense (fully connected) layers as shown in Table 4 where G_x are generator layers and D_x are discriminator layers. Each layer is followed by a LeakyReLU [MHN13] activation and a batch normalization [IS15] layer. The last layers of both sub-networks are unique, the generator has a tanh activation while the discriminator has a Sigmoid activation.

Training `SYNTH` is trained in two stages. First, the generator produces a synthetic template and second, the discriminator outputs how real this synthetic template is. This training is done till convergence of the weights of both networks. Both training stages use the Adam optimizer [KB15]. We randomly flip 2% bits of a real template as noise to aid in network convergence. The networks is trained over 100 epochs. Each epoch having 100 steps. Each step updates weights once by either 1) discriminating a pair of vectors or 2) generating a single synthetic template.

Once trained the `SYNTH` network can produce an unbounded number of distinct templates. We described how to produce different readings from the same template in Section 5.