Private Eyes: Zero-Leakage Iris Searchable Encryption

Julie Ha
Boston University
Boston, MA, United States
hajulie@bu.edu

Chloe Cachet
National Research Council of Canada
Montreal, QC, Canada
chloe.cachet@nrc-cnrc.gc.ca

Luke Demarest
Gonzaga University
Spokane, WA, United States
onlylukejohnson@gmail.com

Sohaib Ahmad
University of Connecticut
Storrs, CT, United States
sohaib.ahmad@uconn.edu

Benjamin Fuller
University of Connecticut
Storrs, CT, United States
benjamin.fuller@uconn.edu

Abstract

This work introduces *Private Eyes*, the first zero-leakage biometric database. The only 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 oblivious 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 which map keywords to values. One computes many LSHs of each record in the database and uses these hashes as keywords in the oblivious 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 oblivious map needs to answer disjunction queries efficiently.

We focus on the iris biometric which 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 solution 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.

For the largest tested parameters of a 5000 synthetic iris database, a search requires 18 rounds of communication and 25ms of parallel computation. Our scheme is implemented and open-sourced.

CCS Concepts

• **Security and privacy → Symmetric cryptography and hash functions**; **Mathematical foundations of cryptography**; **Management and querying of encrypted data**.

Keywords

Searchable Encryption; Biometrics; Proximity Search

CODASPY '25, Pittsburgh, PA, USA.

ACM ISBN 979-8-4007-1476-4/25/6
https://doi.org/10.1145/3714393.3726497

ACM Reference Format:
Julie Ha, Chloe Cachet, Luke Demarest, Sohaib Ahmad, and Benjamin Fuller. 2025. Private Eyes: Zero-Leakage Iris Searchable Encryption. In *Proceedings of the Fifteenth ACM Conference on Data and Application Security and Privacy (CODASPY '25), June 4–6, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3714393.3726497

1 Introduction

Biometrics are collected into large databases for search [13, 26, 31]. Learning stored biometric values enables attackers to break authentication and privacy for a user's lifetime [3, 36, 46, 61, 70, 74]. To reduce this risk, this article develops new searchable encryption techniques for biometric databases [23, 71]. See previous reviews of searchable encryption [10, 35, 50].

Consider a data owner outsourcing 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 (see [50] for an overview). Since biometrics cannot be replaced or revoked, we focus on a **zero-leakage system. A zero-leakage system reveals 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. Components can be replaced with higher-leakage alternatives for efficiency gains.

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. Our parameter analysis shows that these specificities of biometrics result in critical inefficiencies in previous designs. We propose a new design that avoids these pitfalls.

Our Contributions We present `PrivateEyes`, the first zero-leakage proximity search system for the iris. It is enabled by the following contributions:

(1) A two-stage design that 1) finds the non-null clauses (i.e. clauses that match something in the database), 2) obliviously searches these non-null clauses *only*. The first stage can be built with private set intersection (PSI) [32]. The second stage is built using oblivious tree traversal [77].
(2) A prototype implementation with evaluation on random data up to 25000 records and synthetic iris data up to 5000 records [15]. On synthetic data, search takes at most 25ms of parallel computation and 18 rounds of communication. 92% of the computation and 14 rounds are in the second stage, the focus of this work.

To scale beyond the sizes of available iris datasets 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, and Section 8 concludes. The Appendices present proofs and the generation of synthetic irises.

2 Design Overview

A database is a list of biometrics $\mathcal{DB} = w_1, \cdots, w_\ell$ where each $w_i \in \{0,1\}^n$. The goal of a biometric database is, given some biometric query w^*, to find all values $w_i \in \mathcal{DB}$ that are close enough to w^*. For the Hamming metric $\mathcal{D}$ and distance threshold t, that means to find all w_i such that $\mathcal{D}(w_i, w^*) \leq t$.[1]

Like prior work [33, 54, 59, 75], we combine locality-sensitive hashes (LSHs) [47] with an encrypted map.[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 keywords to values and are used to build inverted indices. For a map M, we use the notation $\mathsf{M}[\mathsf{keyw}] = \mathtt{value}$ to denote that keyw is associated with value.

For a database of size ℓ, parameter $\beta \in \mathbb{Z}^+$, LSH family $\mathcal{H}$, and maps $\mathsf{M}_1, \cdots, \mathsf{M}_\beta$, one can achieve proximity search as follows:

(1) Sample β LSHs, one per map: $\mathsf{LSH}_1, \cdots, \mathsf{LSH}_\beta \leftarrow \mathcal{H}$.
(2) In each map, **associate all records with the same LSH output to that LSH output**: for $j = 1, \cdots, \beta$, set
$$\mathsf{M}_j[v] = \{w_i | \mathsf{LSH}_j(w_i) = v\}.^3$$
(3) To search for value w^*, **retrieve all records with at least one LSH output in common with** w^*:
   (a) Compute $\mathsf{LSH}_1(w^*), \cdots, \mathsf{LSH}_\beta(w^*)$.
   (b) Retrieve $\cup_{j=1}^{\beta} \mathsf{M}_j[\mathsf{LSH}_j(w^*)]$.

Queries are disjunctions. Boldyreva and Tang [9] 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, **few of these LSHs 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 try to find a small number, called δ, of LSHs values that exist in some map, and then query exactly those δ maps in a way that hides which 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 detailed description of the approach, shown visually in Figure 1.

Oblivious Membership Check An object to check which LSHs' outputs, $\mathsf{LSH}_j(w^*)$, have matches. For an encrypted stored set X the *oblivious membership check* or OMC takes in a set W and returns a set W' of size δ. If $|W \cap X| \leq \delta$ then W' contains $W \cap X$ and $\delta - |W \cap X|$ dummy values. If $|W \cap X| > \delta$ then W' contains δ randomly selected values from $W \cap X$. In our system, X is the set of all LSHs values $X = \{(j, \mathsf{LSH}_j(w_i))\}_{i,j}$ and $W = \{(j, v) | \mathsf{LSH}_j(w^*) = v\}_j$ for some query w^*.

We build OMC using private set intersection and pseudorandom permutations. We benchmark this design using VolePSI [68]; the resulting implementation takes $2ms$ and 4 rounds of communication for our largest tested parameters.

For each parameter set, we manually find a size δ where if one only searches for δ items in the second stage, there is only a small degradation in accuracy.

Disjunctive Oblivious Map An object that searches for the disjunction of exactly δ items. These δ values form a set of LSHs that exist in the map[4]. Using an oblivious data structure with a constant number of queries 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.

We propose a DOMapE based on oblivious tree traversal, building on the design principles of Wang et al. [77]. In our approach, each map corresponds to a tree, so 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.

To build the encrypted database (Enc stage in figure 1) the client computes the set of all LSH values using the OMC (with their index j). They also create the maps associating LSH values to the corresponding biometric values and build the corresponding DOMapE. To search for query w^* (Find stage in figure 1), the client computes the corresponding LSH values and uses OMC.Find to find a subset of those values present in the maps. DOMapE.Find is then used to retrieve the corresponding non-null values from the maps. The inputs to both OMC.Find and DOMapE.Find are of constant size, yielding a zero-leakage solution.

We implemented and analyzed this construction on datasets of up to 25000 records. Since available iris datasets are not this large, we created larger synthetic iris datasets up to 5000 records using

[1] This functionality differs from k-nearest neighbors where the goal is to retrieve the k closest records [9]. There have been leakage abuse attacks against k-nearest neighbor systems that reveal access pattern [52, 53, 56] and resulting systems [18]. These attacks do not apply to our leakage profile.

[2] We won't discuss works that use encrypted maps but require work proportional to the total number of close points, making them impractical for biometrics [8, 57, 76].

[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 keyword. This condition is satisfied for the accuracy regimes discussed in this work.

[4] Except for dummy values added when the total number of matches is less than δ.

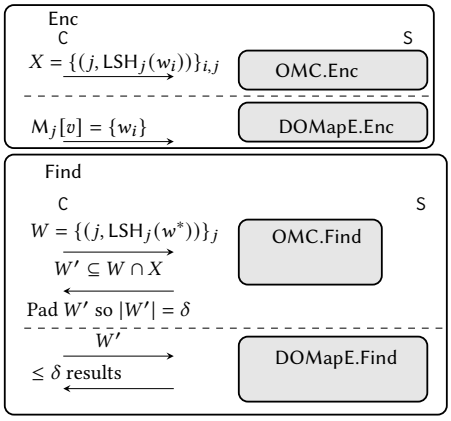


Figure 1: System Overview Composing OMC **and** DOMapE**. In** Find **one checks which LSH values exist in the database using the** OMC**, finding at most δ candidates. If needed** OMC **pads this set of candidates with dummy values to reach $|W'| = \delta$.** DOMapE **is designed to have no leakage if the size of the disjunction is constant (δ). See Construction 2 for a formal description.**

generative adversarial networks or GANs [22, 42] and random data to 25K records. For all tested parameters we have $\delta < 75$ (in contrast to a number of LSHs $\beta > 600$), so a server could reasonably process the ORAM requests at each tree level in parallel. Thus, we report on both parallel and sequential time.

Comparison with prior work Cachet et al. [1] 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, search took 4 minutes on a dataset of size 356. For the solution with less leakage, the search took 1 hour [14].

Barni et al. [4] and Blanton et al. [6] evaluated their fingerprints identification systems on small datasets of 320 readings. They respectively achieved search times of 16 and 0.45 seconds for the server. Their protocols use Garbled circuits and only a single round of communication. Blanton et al. [6] also proposed an iris identification system that compares two 2048 bits iris readings in 0.15s. For a database of size 356 (our smallest setting) this would amount to 53s. For face recognition, SciFi [63] online search runtime is linear in the size of the database: for 100 faces representations search takes 31s. Erkin et al. [28]'s system takes roughly 40s to search over a dataset of size 320. As a reminder on 5000 irises our search is 25ms with 18 rounds. For a network delay of 100ms, this corresponds to an overall time of $\approx$ 2s.

Recent work [17, 38, 73] extends PSI to the setting where one considers items a match if their distance is small. A naive use of fuzzy PSI tells the client if the biometric exists in the database but not which item it matches. This issue can be solved using labeled PSI, [19] which associates a label with each set value $x \in X$. This is the approach used by Uzun et al. [73]. Uzun et al. [73] evaluate 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 [39] 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 uses $\beta = 64$ by averaging multiple biometric readings, such techniques can be applied in our setting as well. Averaging readings is not standard in the biometric literature so we do not do it.

More work is needed to scale zero-leakage biometric databases. Our cryptographic storage overhead is around 5000 for all tested parameters. The required number of LSHs grows with the database size. Scaling to 10^6 irises requires 93000 LSHs (using linear interpolation over the number of required LSHs for synthetic data with an $r^2 = .88$) for an overall storage of 93 billion LSH outputs (roughly 500TB for the cryptographic object). In our current testing, we load the whole cryptographic object in memory with each query, which is not possible for larger sizes.

3 Preliminaries

Let λ be the security parameter, we use $\mathtt{poly}(\lambda)$ and $\mathtt{negl}(\lambda)$ 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 $x \leftarrow S$ denote sampling x uniformly at random from the finite set S. Let $\perp_1, \perp_2, \ldots$ be a sequence of distinguished unique symbols. These symbols are allowed inputs to algorithms. For a map M, let M.Keywords output the set of all stored keywords.

For interactive protocol Prot between a client C and a server S we use notation $\begin{pmatrix} o_\mathsf{C} \\ o_\mathsf{S} \end{pmatrix} \leftarrow \mathtt{Prot} \begin{pmatrix} i_\mathsf{C} \\ i_\mathsf{S} \end{pmatrix}$ with $i_\mathsf{C}, o_\mathsf{C}, i_\mathsf{S}, o_\mathsf{S}$ denoting the client's and the server's inputs and outputs respectively. Protocols are written from the perspective of the client with underlying interactive algorithms indicating the server's role.

Hamming distance is defined as the distance between the bit vectors x and y of length n, $\mathcal{D}(x, y) = |\{i \mid x_i \neq y_i\}|$. The fractional Hamming distance is $\mathcal{D}(x, y)/n$.

Definition 3.1 (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 $\mathcal{D}(x, y) \leq t$, $\Pr_{\mathsf{LSH} \leftarrow \mathcal{H}}[\mathsf{LSH}(x) = \mathsf{LSH}(y)] \geq 1 - p_1$ and
(2) If $\mathcal{D}(x, y) \geq ct$, $\Pr_{\mathsf{LSH} \leftarrow \mathcal{H}}[\mathsf{LSH}(x) = \mathsf{LSH}(y)] \leq 1 - p_2$.

For x, y if $\mathcal{D}(x, y) \leq t$ they are called near, if $\mathcal{D}(x, y) \geq ct$ they are called 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 and use the ND-0405 dataset [11, 66]. The average distance between same irises is $t/n \approx .21$, using a state-of-the-art feature extractor [2]. Section 5 presents the datasets used in this work. Our discussion applies to other biometrics with substantive noise such as the face [5].

If one only uses the AND of α LSHs and considers it a match when at least one LSH out of β corresponds, 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 a search query, 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 $65 \le \beta \le 80$ 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 $680 \le \beta \le 835$ 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 8(a)), requiring β to be larger as we show in Table 1.

We call an LSH match *good* if it ensures the query results in a true accept and *bad* otherwise. For the ND-0405 dataset,[6] with a larger $\beta = 225$ and $\alpha = 15$, the average number of total LSH matches is 23.4.

3.2 Cryptographic Definitions

This work relies on *oblivious RAM* (ORAM) [40, 41] to achieve zero-leakage. In our constructions, we consider static datasets and ignore write queries. Our ORAM definition reflects this choice. As we discuss in Section 6, this definition is also satisfied by private information retrieval schemes (with appropriate encryption). We discuss other considerations for dynamic data at the end of Section 6.

Definition 3.2 (Oblivious RAM). An Oblivious RAM (ORAM) scheme is two protocols, Setup and Access:

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

[5] Deng et al. [27, Figure 6] show analogous statistics for the face.

[6] This uses the following experiment:
(1) Storage of a single feature extracted reading for the eye for each of the 356 persons. Sample $\beta = 225$ LSHs of size $\alpha = 15$.
(2) Let $w_1', \ldots, w_{356}'$ be the second stored template in the ND-0405 dataset.
(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.
(3) For $1 \le i \le q$:
  (a) $y_i \leftarrow \mathcal{A}(\mathsf{ts}_{i-1})$,
  (b) $\begin{pmatrix}v_i, \sigma_i\\ \mathsf{EM}_i\end{pmatrix} \leftarrow \mathsf{Access}\begin{pmatrix}\sigma_{i-1}, y_i\\ \mathsf{EM}_{i-1}\end{pmatrix}$.
  (c) Let ts_i be the server's view.
(4) Output $(\mathsf{ts}_0, \cdots, \mathsf{ts}_q)$.

$\underline{\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q}(1^\lambda)}$: Output $(\mathsf{ts}_0, ..., \mathsf{ts}_q) \leftarrow \mathsf{Sim}(q, |\mathsf{Mem}|, 1^\lambda)$.

Figure 2: Definition of Ideal and Real for ORAM security.

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) For $1 \le i \le q$:
  (a) Run $y_i \leftarrow \mathcal{A}(\mathsf{ts}_{i-1})$.
  (b) Run $\begin{pmatrix}v_i, \sigma_i,\\ \mathsf{EM}_i\end{pmatrix} \leftarrow \mathsf{Access}\begin{pmatrix}\sigma_{i-1}, y_i\\ \mathsf{EM}_{i-1}\end{pmatrix}$.

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 $\mathsf{negl}(\lambda)$.

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 the distributions $\mathsf{Real}_{\mathcal{A},q}$ and $\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q}$, described in Figure 2, are computationally indistinguishable.

The above is an adaptive simulation definition of ORAM [37], 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 3.3 (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 OSE = (Setup, Enc, Find) have the following format:

$$\begin{pmatrix}\mathsf{sk}\\ \mathsf{pp}\end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix}1^\lambda, \mathsf{param}\\ 1^\lambda, \mathsf{param}\end{pmatrix},$$
$$\begin{pmatrix}I_\mathsf{C},\\ I_\mathsf{S}\end{pmatrix} \leftarrow \mathsf{Enc}\begin{pmatrix}\mathsf{sk}, \mathcal{DB}\\ \mathsf{pp}\end{pmatrix}, \begin{pmatrix}J, I_\mathsf{C}'\\ I_\mathsf{S}'\end{pmatrix} \leftarrow \mathsf{Find}\begin{pmatrix}\mathsf{sk}, y, I_\mathsf{C},\\ \mathsf{pp}, I_\mathsf{S}\end{pmatrix}.$$

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.

Security: Let $q = \mathsf{poly}(\lambda)$, $\mathcal{L}_\mathsf{OSE} = \{\mathcal{L}_\mathsf{Enc}, \mathcal{L}_\mathsf{Find} = \perp\}$ be the leakage profile of OSE's algorithms. For any PPT adversary $\mathcal{A}$, there exists a simulator Sim such that the distributions $\mathsf{Real}_{\mathcal{A},q,\mathsf{param}}$ and $\mathsf{Ideal}_{\mathcal{A},\mathsf{Sim},q,\mathsf{param}}$, described in Figure 3, are computationally indistinguishable.

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

(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 $\begin{pmatrix}I_{\mathsf{C},1},\\ I_{\mathsf{S},1}\end{pmatrix} \leftarrow \mathsf{Enc}\begin{pmatrix}\mathsf{sk}, \mathcal{DB}\\ \mathsf{pp}\end{pmatrix}$.
Let ts_1 be the server's view.
(4) For $1 \le j \le q$:
(a) $y^j \leftarrow \mathcal{A}(\mathsf{ts}_j)$.
(b) $\begin{pmatrix}J_j, I_{\mathsf{C},j+1}\\ I_{\mathsf{S},j+1}\end{pmatrix} \leftarrow \mathsf{Find}\begin{pmatrix}\mathsf{sk}, y_j, I_{\mathsf{C},j}\\ \mathsf{pp}, I_{\mathsf{S},j}\end{pmatrix}$.
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) $\mathsf{ts}_0 \leftarrow \mathsf{Sim}(1^\lambda, q, \mathsf{param})$.
(2) $\mathcal{DB} \leftarrow \mathcal{A}(\mathsf{ts}_0)$.
(3) $(\mathsf{ts}_1, ..., \mathsf{ts}_{q+1}) \leftarrow \mathsf{Sim}(\mathcal{L}_{\mathsf{Enc}}(\mathcal{DB}))$.
(4) Output $(\mathsf{ts}_0, ...\mathsf{ts}_{q+1})$.

Figure 3: Definition of Ideal and Real for OSE security.

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

Definition 3.4 (Oblivious Proximity Search). Consider Definition 3.3 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 $J_{\mathcal{DB},\mathsf{near},y} := \{w_i | \mathcal{D}(w_i, y) \le t\}$. Let $q = \mathsf{poly}(\lambda)$ and $\epsilon > 0$. For all $\mathcal{DB}$ and all $y_1, ..., y_q$ define:

$$\begin{pmatrix}\mathsf{sk}\\ \mathsf{pp}\end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix}1^\lambda\\ 1^\lambda\end{pmatrix},$$
$$\begin{pmatrix}I^1_{\mathsf{C}},\\ I^1_{\mathsf{S}}\end{pmatrix} \leftarrow \mathsf{Enc}\begin{pmatrix}\mathsf{sk}, \mathcal{DB}\\ \mathsf{pp}\end{pmatrix}, \begin{pmatrix}J_j, I_{\mathsf{C},j+1}\\ I_{\mathsf{S},j+1}\end{pmatrix} \leftarrow \mathsf{Find}\begin{pmatrix}\mathsf{sk}, y_j, I_{\mathsf{C},j}\\ \mathsf{pp}, I_{\mathsf{S},j}\end{pmatrix}$$

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

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

Definition 3.4 doesn't limit the number of false matches. Furthermore, in Section 4, we never show that our construction satisfies approximate correctness. 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 4.2). 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 are 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 3.5 (Oblivious Membership Check). Let $\mathsf{OMC} = (\mathsf{Enc}, \mathsf{Find})$ 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 3.3 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

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

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

Finally, we define the second stage of our system:

Definition 3.6 (Disjunctive Oblivious Map with Encryption). Let $\mathsf{param} = (\beta, \delta)$, such that $\beta, \delta \in \mathbb{N}$ and $\delta \le \beta$ and let $\mu \in \mathbb{N}$. Let $\mathsf{DOMapE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Find})$ 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 3.3 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 $1 \le i \le \beta$, $|\mathsf{M}.\mathsf{Keywords}| \le 2^\mu$. Set $\mathcal{DB} = \mathsf{M}_1, \cdots, \mathsf{M}_\beta$. Let $\epsilon > 0$, $q, \beta, \delta = \mathsf{poly}(\lambda)$ and $\delta \le \beta$. 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 $1 \le j \le q$:

$$\begin{pmatrix}\mathsf{sk}\\ \mathsf{pp}\end{pmatrix} \leftarrow \mathsf{Setup}\begin{pmatrix}1^\lambda, \beta, \delta\\ 1^\lambda\end{pmatrix},$$
$$\begin{pmatrix}\sigma_1,\\ \mathsf{EM}_1\end{pmatrix} \leftarrow \mathsf{Enc}\begin{pmatrix}\mathsf{sk}, \mathcal{DB}\\ \mathsf{pp}\end{pmatrix}, \begin{pmatrix}r_j, \sigma_{j+1},\\ \mathsf{EM}_{j+1}\end{pmatrix} \leftarrow \mathsf{Find}\begin{pmatrix}\mathsf{sk}, \sigma_j, y_j\\ \mathsf{pp}, \mathsf{EM}_j\end{pmatrix}.$$

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

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

4 Oblivious Proximity Search for Biometrics

This section presents our technical solutions, focusing on the design of DOMapE. We describe possible constructions of OMC in Section 4.2. The most relevant related work is by Boldyreva and Tang [9], 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 [67], we present a tree based construction. In this work, we only consider static data. For static data, B-trees and skip lists are equivalent data structures [55]. However, updates and the resulting performance differ.

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

$$\mathsf{M}_j[\mathsf{keyw}] = \{w_i | \mathsf{LSH}_j(w_i) = v\}.$$

(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 keywords y_i are placed into the map, sorted (lexographically) and used as nodes in a binary tree (along with $\{w_i\}$). Internal nodes have store the minimum keyword 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.

To turn this into an oblivious search algorithm, one can place each tree 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 corresponds to Boldyreva and Tang's approach and is shown in Figure 4(a) . 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. This design is shown visually in Figure 4(b), with each shaded region representing a separate ORAM. With this approach, each shaded region sees exactly one read for each search query. This organization allows higher levels of the tree to serve as the position map of their child eliminating the need for a recursive ORAM [77].

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 prevent this. However, we retain a separate ORAM for each level of the trees. This is shown visually in Figure 4(c) and also described by the ApplyO 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. In this design, parents store the position map of children, enabling non-recursive ORAM.

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* $\mathsf{O} = (\mathsf{O.Setup}, \mathsf{O.Access})$ *be an oblivious RAM and let O_i denote its instantiation for level $0 \leq i \leq \mu$. Consider the* DOMapE *construction shown in Figure 6.*

Theorem 4.1. *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 $\beta -$ nmaps, $\mu -$ msize, and $\delta -$ qsize.*

The proof of Theorem 4.1 is in the full version of this work [44, Theorem 1].

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

For a database $\mathcal{DB} = (w_1, ..., w_\ell)$, define $\mathsf{OSE} = (\mathsf{OSE.Setup}, \mathsf{OSE.Enc}, \mathsf{OSE.Find})$ *as in Figure 7.*

Theorem 4.2. *Let* DOMapE *and* OMC *be as in Construction 2. Then Construction 2 is an oblivious searchable encryption scheme with leakage $\mathcal{L}_{\mathsf{Enc}}(\mathsf{M}_1, ...\mathsf{M}_\beta) = \mu$.*

Theorem 4.2's proof is straightforward and is deferred to the full version of this work [44, Theorem 2]. Roughly, it shows that the composition of OMC and the DOMapE is a disjunctive map that takes in size β queries and returns δ records. Theorem 4.2 does not handle correctness. Since there is an overlap between the histograms for real data in Figure 8 one cannot make strong correctness claims. We evaluate correctness empirically in Section 7.

4.2 Oblivious membership check constructions

We discuss options to implement OMC. We briefly cover approaches based on Bloom filter lookups. In Appendix A, 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 local Bloom filter to emulate an OMC.

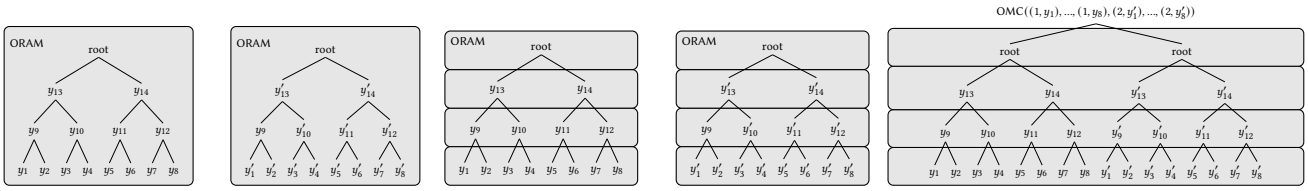
Oblivious Bloom Filter Lookups The client's set can be stored in a Bloom [7], Cuckoo [29], or XOR [43] filters which is then stored on the server in an ORAM. The client will request the relevant bits from the ORAM. This prevents the client from having to store the entire filter on their side, but requires them to request multiple ORAM accesses to query the relevant bits. BlindSEER [30, 64] 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 [34].

5 Datasets

We test and evaluate our implementation on three datasets:

ND-0405 dataset This dataset [11, 66] is a superset of the NIST Iris Evaluation Challenge [65]. It consists of the readings of left and right irises from 356 individuals, each iris having at least 4 distinct readings. We use the ThirdEye feature extractor [2] to obtain 1024 bits feature vectors from the original iris images. 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 8(a).

Synthetic dataset Available irises datasets are of limited size, often no more than a few hundreds irises (356 individuals for ND-0405).



(a) Boldyreva and Tang ORAM organization [9]. (b) First optimization. (c) Our design.

Figure 4: ORAM Organization Strategies. Each shaded region represents data stored together in a single oblivious RAM. In (a) one uses a separate ORAM for each LSH. In (b) we split this to one ORAM per tree level. Then (c) is our design of DOMapE **where each level across binary trees is stored in a single ORAM. This allows one to hide which trees are accessed.**

BIndex(M, μ):

(1) Sort map M using the comparator $\leq$.
Let Leaves $= (x_i, \mathsf{M}[x_i])$ be the sorted result.
(2) Pad Leaves to length 2^μ with pairs $(\perp_i, \perp_i)$.
(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.

ApplyO$\begin{pmatrix} \mathsf{Tr}_1, ..., \mathsf{Tr}_\beta \\ 1^\lambda \end{pmatrix}$:

(1) For $j \in [0, \mu]$:
(a) $\mathsf{Level}_j = \perp$. For all $\mathsf{Tr}_{i\in[\beta]}$, $\mathsf{Level}_j = \mathsf{Level}_j \,||\, \mathsf{Level}(\mathsf{Tr}_i, j)$.
(b) $\begin{pmatrix} \sigma_j, \\ \mathsf{EM}_j \end{pmatrix} \leftarrow \mathsf{O.Setup}\begin{pmatrix} 1^\lambda, \mathsf{Level}_j \\ , \perp \end{pmatrix}$
(2) Denote $\Sigma := \{\sigma_j\}_{j=0}^{\mu}$ and $\mathsf{EM} = \{\mathsf{EM}_j\}_{j=0}^{\mu}$.

Level(Tr, j): Return all nodes at level j in Tr.

Figure 5: Build tree index and apply ORAM algorithms.

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 8b, synthetic data same and different distributions are similar to the ND-0405 ones. The details on synthetic data generation are in Appendix B. The high level approach is a generative adversarial network (GAN) [42] as in prior approaches on synthetic iris generation [3].

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 8c.

Random and synthetic queries generation Contrary to the ND0405 dataset [11], 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 [24, 25, 69]. From Figure 8a, 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 then generate queries for the random and synthetic datasets as follows:

(1) Generate a binomial distribution using the mean and standard deviation of the same iris distribution for the ND-0405.
(2) For each feature vector in the dataset, create a corresponding query by sampling an error fraction from the $\texttt{frac} \leftarrow B(53, 0.21)/53$.
(3) Flip the number of error bits, $\texttt{frac} * 1024$, in the feature vector.

Using this technique, we obtain the same iris distributions (in blue) for synthetic and random data shown in Figures 8b and 8c. There are only 54 possible outcomes for a fraction of error bits, this leads to discontinuities in the histograms presented in Figures 8b and 8c.

6 Implementation

We present an open-source implementation of our algorithms including the LSH parameter finding, tree building, and oblivious search [15]. This implementation is in Python 3.10 and uses the PathORAM [72] module [45]. Our experiments use a Bloom filter cache on the client as an OMC to focus on the performance of the developed DOMapE. We separately evaluate an OMC candidate based on PSI in Section 7. Our implementation supports two main conclusions.

(1) 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 1 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

$\begin{pmatrix}\beta, \nu, \delta\\ \beta, \nu, \delta\end{pmatrix} \leftarrow \text{Setup}\begin{pmatrix}1^\lambda, \beta, \nu, \delta\\ 1^\lambda\end{pmatrix}$

$\text{Enc}\begin{pmatrix}\mathsf{M}_1, \cdots, \mathsf{M}_\beta\\ 1^\lambda\end{pmatrix}$:

(1) Let $\eta_i = |\mathsf{M}_i|$, 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) C initiates $\begin{pmatrix}\Sigma,\\ \mathsf{EM}\end{pmatrix} \leftarrow \text{ApplyO}\begin{pmatrix}\mathsf{Tr}_1, ..., \mathsf{Tr}_\beta\\ 1^\lambda\end{pmatrix}$.
(4) Server receives EM and client keeps Σ.

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

C does:

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

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

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

(4) For i in 1 to ν:

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

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

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

Figure 6: DOMAPE Construction. The BIndex **algorithm is shown in Figure 5.**

different irises. This increases the number of false matches. As one exception, we set δ for our 25K random parameters heuristically based on the smaller dataset sizes.

(2) The two stage DOMapE approach improves search performance. While setup takes several hours, parallel search is at most 35ms. Without the two stage approach parallel search would require a server with hundreds of threads. See further timing discussion in Section 7.2.

Dataset Modifications Our implementation does not allow for insertion after the initial building of the tree. With ORAM one can rebuild the trees using techniques of Wang et al. [77].

Alternative to ORAM In the static setting, one can use private information retrieval (PIR) [20, 21] with encryption. At retrieval, single server computational PIR and PathORAM [72] 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) [5] 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. DEPIR

$\text{OSE.Setup}\begin{pmatrix}1^\lambda\\ 1^\lambda\end{pmatrix}$:

(1) Client samples $\mathsf{LSH}_1, ..., \mathsf{LSH}_\beta \leftarrow \mathcal{H}(1^\lambda)$ and runs

$\begin{pmatrix}\mathsf{sk}_{\mathsf{OMC}}\\ \perp\end{pmatrix} \leftarrow \text{OMC.Setup}\begin{pmatrix}1^\lambda\\ \perp\end{pmatrix}$,

$\begin{pmatrix}\mathsf{sk}_{\mathsf{DOMapE}}\\ \mathsf{pp}_{\mathsf{DOMapE}}\end{pmatrix} \leftarrow \text{DOMapE.Setup}\begin{pmatrix}1^\lambda\\ 1^\lambda\end{pmatrix}$.

(2) Denote $\mathsf{pp} = \mathsf{pp}_{\mathsf{DOMapE}}, \mathsf{sk} = (\mathsf{sk}_{\mathsf{OMC}}, \mathsf{sk}_{\mathsf{DOMapE}}, \mathsf{LSH}_1, ..., \mathsf{LSH}_\beta)$.
(3) Client sends pp to server and keeps sk.

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

(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)

$\begin{pmatrix}\perp\\ ES\end{pmatrix} \leftarrow \text{OMC.Enc}\begin{pmatrix}\mathsf{sk}_{\mathsf{OMC}}, \cup_{i=1}^{\beta} \cup_{j=1}^{\ell} i || \mathsf{keyword}_{i,j}\\ 1^\lambda\end{pmatrix}$,

$\begin{pmatrix}\sigma,\\ \mathsf{EM}\end{pmatrix} \leftarrow \text{DOMapE.Enc}\begin{pmatrix}\mathsf{sk}, \mathsf{M}_1, ..., \mathsf{M}_\beta\\ \mathsf{pp}_{\mathsf{DOMapE}}\end{pmatrix}$.

(3) Denote $I_{\mathsf{C}} = \sigma$ and $I_{\mathsf{S}} = (ES, \mathsf{EM})$.

$\texttt{OSE.Find}\begin{pmatrix}\mathsf{sk}, y, I_{\mathsf{C}}\\ I_{\mathsf{S}}\end{pmatrix}$:

(1) Let $I_s = (ES, \mathsf{EM})$.
(2) Client creates OMC set $EC = (1 || \mathsf{LSH}_1(y), \cdots, \beta || \mathsf{LSH}_\beta(y))$.
(3)

$\begin{pmatrix}\mathsf{Res}_{\mathsf{OMC}}\\ \perp\end{pmatrix} \leftarrow \text{OMC.Search}\begin{pmatrix}\mathsf{sk}_{\mathsf{OMC}}, EC\\ ES\end{pmatrix}$,

$\begin{pmatrix}r, \sigma'\\ \mathsf{EM}'\end{pmatrix} \leftarrow \text{DOMapE.Search}\begin{pmatrix}\mathsf{sk}_{\mathsf{DOMapE}}, \sigma, \mathsf{Res}_{\mathsf{OMC}}\\ \mathsf{EM}\end{pmatrix}$.

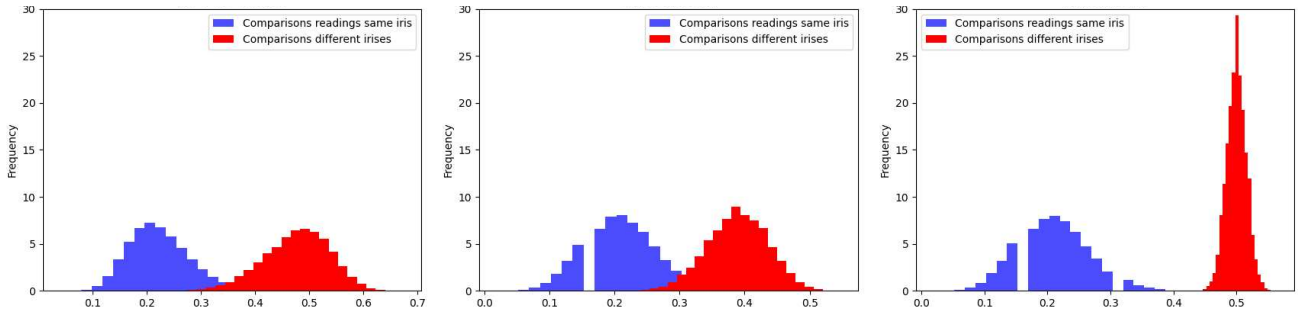
(4) Denote $J = (\cup_{i=1}^{\delta} r_i) \setminus \perp, I'_{\mathsf{C}} = \sigma'$ and $I'_{\mathsf{S}} = (ES, \mathsf{EM}')$.
(5) Client receives J, I'_{C} and server stores I'_{S}.

Figure 7: OSE **construction from** OMC, DOMapE **and** LSH.

constructions are based on ring LWE [58] or a non-standard secretly permuted Reed-Muller codes assumption [12, 16]. Currently, the ring LWE DEPIR is asymptotically efficient but inefficient in practice [62].

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.



(a) Histogram of comparisons for ND0405 dataset. (b) Histogram of comparisons for synthetic dataset. (c) Histogram of comparisons for random data.

Figure 8: 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.

7.1 Accuracy - Parameter analysis

Each experiment is conducted on each dataset. Recall the relevant parameters: α, the length of the extended LSH, β – `nmaps`, and δ – `qsize`.

Finding parameters The first part of the experiment was a manual search across parameters 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.

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.

As one exception, due to slow speed and high memory overhead for the 25K random dataset, we picked $\delta = 20$ based on our parameters from smaller datasets, this dataset naturally had a larger number of bad matches but still displayed a high TAR of .91 when restricted to $\delta = 20$ maps.

Discussion In proximity search, high TAR requires capturing the tail of comparisons between different readings of the same iris (shown in Figure 8). For example, for distance $t = .21n$ and a FAR of .01, Section 3.1 proposed $\beta = 65$ and $\alpha = 13$. Table 1 shows that even for random data, we require $\beta = 630$ and $\alpha = 15$. These parameters increase further on the ND and Synthetic 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. Restricting to only δ accesses in the DOMapE does harm TAR. The worst degradation 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 AMD Ryzen Threadripper PRO 7995WX CPU with 96 cores and 768 GB of RAM, running Ubuntu 22.04. Results are in Table 2. We did not model network delay.

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 2, 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 hours, while larger datasets take days.

Parallel search time, the sum of the max required time to complete each ORAM read at each level, remains under 35ms on all tested parameters. We note that parallel search time is less than sequential search time divided by δ as sequential search time includes reading all ORAMs from disk at the beginning of each search and writing them back to disk at the end of each search. This is done with every query. Only searching δ trees is critical to process the ORAM reads in parallel on a moderately powerful modern server.

Network Round Trips We report on two figures, the number of round trips using a purely sequential non-recursive PathORAM implementation and the number of roundtrips if one is able to fully batch all requests at the same level. One can store the position map for ORAMs in the prior tree level [77], enabling non-recursive constructions. For the largest synthetic parameter sizes, if one assumes a fast network with $60ms$ responses and unbounded bandwidth then network delays result in .84 seconds in parallel rounds trips, but slower $1s$ responses results in 14 seconds.

[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	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
25000	random	22	3500	8.7	57.2	277	6.1	-	20	0.91

Table 1: TAR/FAR and the number of matches for random, ND0405, and synthetic datasets of different sizes. For 25K dataset, δ was set manually (not 95% of bad matches).

Dataset size, ℓ	Dataset type	α	β nmaps	δ qsize	# Queries	ORAM Reads	# Roundtrips seq.	par.	Time (s) O.Init	seq.	par.	Size EDB (GB)
356	random	15	630	13	356	117	118	10	1.1×10^3	.65	.013	1.1
356	ND	18	850	37	356	333	334	10	1.5×10^3	.91	.013	1.1
356	synthetic	18	850	26	356	234	235	10	1.6×10^3	.85	.013	1.1
1000	random	18	850	8	500	80	81	11	3.1×10^3	1.06	.015	2.2
1000	synthetic	19	1000	47	500	470	471	11	3.6×10^3	1.51	.016	2.2
2500	random	19	1000	11	500	132	133	13	16×10^3	3.44	.018	8.9
2500	synthetic	21	1200	56	500	672	673	13	20×10^3	4.52	.020	17.8
5000	random	20	1200	12	500	156	157	14	42×10^3	7.41	.021	35.6
5000	synthetic	22	1300	72	500	936	937	14	45×10^3	8.82	.023	35.6
25000	random	22	3500	20	100	300	301	16	552×10^3	7.89	.035	285

Table 2: 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 $1 +$ #ORAM Reads and Parallel Rounds trips is $\lceil \log_2 \ell \rceil + 1$. All timing numbers are averaged across the number of queries in # Queries.

Using a batched ORAM implementation 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 [78].

Evaluation of OMC implementation using private set intersection On the same hardware as the rest of the evaluation we deployed the VolePSI implementation [68]. To test the largest synthetic parameters, 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 2. 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. To get the results in our abstract and introduction, we add 2ms to Table 2 and four rounds of communication. These results justify the focus on the design of DOMapE.

8 Conclusion

Private Eyes was tested with parallel response times of at most 35ms 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. The statistics of biometric data inspired a 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 but one could use trees with a higher branching factor (or skiplists as in [8]) 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. was supported by NSF Award #1950600. C.C. was supported by NSF Award #2141033. L.D. was supported by the Harriott Fellowship. S.A. is supported by a fellowship from Synchrony Inc. and the State of Connecticut. B.F. was 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

[1] Ahmad, S., Cachet, C., Demarest, L., Fuller, B., and Hamlin, A. Proximity searchable encryption for the iris biometric. In *AsiaCCS* (2022). https://ia.cr/2020/1174.

[2] Ahmad, S., and Fuller, B. Thirdeye: Triplet-based iris recognition without normalization. In *IEEE BTAS* (2019).

[3] Ahmad, S., and Fuller, B. Resist: Reconstruction of irises from templates. In *IJCB* (2020), IEEE, pp. 1–10.

[4] Barni, M., Bianchi, T., Catalano, D., Di Raimondo, M., Donida Labati, R., Failla, P., Fiore, D., Lazzeretti, R., Piuri, V., Scotti, F., and Piva, A. Privacy-preserving fingercode authentication. In *Proceedings of the 12th ACM Workshop on Multimedia and Security* (New York, NY, USA, 2010), Association for Computing Machinery, p. 231–240.

[5] Beimel, A., Ishai, Y., and Malkin, T. Reducing the servers computation in private information retrieval: Pir with preprocessing. In *Advances in Cryptology — CRYPTO 2000* (Berlin, Heidelberg, 2000), M. Bellare, Ed., Springer Berlin Heidelberg, pp. 55–73.

[6] Blanton, M., and Gasti, P. Secure and efficient protocols for iris and fingerprint identification. In *Computer Security – ESORICS 2011* (Berlin, Heidelberg, 2011), V. Atluri and C. Diaz, Eds., Springer Berlin Heidelberg, pp. 190–209.

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

[8] Boldyreva, A., and Chenette, N. Efficient fuzzy search on encrypted data. In *International Workshop on Fast Software Encryption* (2014), Springer, pp. 613–633.

[9] Boldyreva, A., and Tang, T. Privacy-preserving approximate k-nearest-neighbors search that hides access, query and volume patterns. *PoPETS* (2021).

[10] Bösch, C., Hartel, P., Jonker, W., and Peter, A. A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR) 47*, 2 (2014), 1–51.

[11] Bowyer, K. W., and Flynn, P. J. The ND-IRIS-0405 iris image dataset. *arXiv preprint arXiv:1606.04853* (2016).

[12] Boyle, E., Ishai, Y., Pass, R., and Wootters, M. Can we access a database both locally and privately? In *Theory of Cryptography* (Cham, 2017), Y. Kalai and L. Reyzin, Eds., Springer International Publishing, pp. 662–693.

[13] Brislawn, C. M., Bradley, J. N., Onyshczak, R. J., and Hopper, T. The FBI compression standard for digitized fingerprint images. In *Proc. SPIE* (1996), vol. 2847, pp. 344–355.

[14] Cachet, C., Ahmad, S., Demarest, L., Riback, S., Hamlin, A., and Fuller, B. Multi random projection inner product encryption, applications to proximity searchable encryption for the iris biometric. *Information and Computation 293* (2023), 105059.

[15] Cachet, C., Ha, J., and Fuller, B. An implementation of oblivious proximity searchable encryption. https://github.com/hajulie/searchable_biometric, 2024.

[16] Canetti, R., Holmgren, J., and Richelson, S. Towards doubly efficient private information retrieval. In *Theory of Cryptography* (Cham, 2017), Y. Kalai and L. Reyzin, Eds., Springer International Publishing, pp. 694–726.

[17] Chakraborti, A., Fanti, G., and Reiter, M. K. Distance-Aware private set intersection. In *USENIX Security* (2023), pp. 319–336.

[18] Chen, H., Chillotti, I., Dong, Y., Poburinnaya, O., Razenshteyn, I., and Riazi, M. S. SANNS: Scaling up secure approximate k-Nearest neighbors search. In *USENIX Security* (2020), pp. 2111–2128.

[19] Chen, H., Huang, Z., Laine, K., and Rindal, P. Labeled psi from fully homomorphic encryption with malicious security. In *CCS* (2018), pp. 1223–1237.

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

[21] Chor, B., Kushilevitz, E., Goldreich, O., and Sudan, M. Private information retrieval. *J. ACM 45*, 6 (nov 1998), 965–981.

[22] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. Generative adversarial networks: An overview. *IEEE signal processing magazine 35*, 1 (2018), 53–65.

[23] Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security 19* (01 2011), 895–934.

[24] Daugman, J. Results from 200 billion iris cross-comparisons.

[25] Daugman, J. How iris recognition works. In *The essential guide to image processing*. Elsevier, 2009, pp. 715–739.

[26] Daugman, J. 600 million citizens of India are now enrolled with biometric id,". *SPIE newsroom 7* (2014).

[27] Deng, J., Guo, J., Xue, N., and Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. In *CVPR* (2019), pp. 4690–4699.

[28] Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., and Toft, T. Privacy-preserving face recognition. In *Privacy Enhancing Technologies* (Berlin, Heidelberg, 2009), I. Goldberg and M. J. Atallah, Eds., Springer Berlin Heidelberg, pp. 235–253.

[29] Fan, B., Andersen, D. G., Kaminsky, M., and Mitzenmacher, M. D. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies* (2014), pp. 75–88.

[30] Fisch, B. A., Vo, B., Krell, F., Kumarasubramanian, A., Kolesnikov, V., Malkin, T., and Bellovin, S. M. Malicious-client security in blind seer: a scalable private dbms. In *2015 IEEE Symposium on Security and Privacy* (2015), IEEE, pp. 395–410.

[31] Foundation, E. F. Mandatory national ids and biometric databases.

[32] Freedman, M. J., Nissim, K., and Pinkas, B. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques* (2004), Springer, pp. 1–19.

[33] Fu, Z., Wu, X., Guan, C., Sun, X., and Ren, K. Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics and Security 11*, 12 (2016), 2706–2716.

[34] Fuller, B., Mitchell, D., Cunningham, R., Blumenthal, U., Cable, P., Hamlin, A., Milechin, L., Rabe, M., Schear, N., Shay, R., et al. Security and privacy assurance research (spar) pilot final report. Tech. rep., MIT Lincoln Laboratory Lexington United States, 2015.

[35] Fuller, B., Varia, M., Yerukhimovich, A., Shen, E., Hamlin, A., Gadepally, V., Shay, R., Mitchell, J. D., and Cunningham, R. K. SoK: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy (SP)* (2017), IEEE, pp. 172–191.

[36] Galbally, J., Ross, A., Gomez-Barrero, M., Fierrez, J., and Ortega-Garcia, J. From the iriscode to the iris: A new vulnerability of iris recognition systems. *Black Hat Briefings USA 1* (2012).

[37] Garg, S., Mohassel, P., and Papamanthou, C. Tworam: efficient oblivious ram in two rounds with applications to searchable encryption. In *Annual International Cryptology Conference* (2016), Springer, pp. 563–592.

[38] Garimella, G., Rosulek, M., and Singh, J. Structure-aware private set intersection, with applications to fuzzy matching. In *Annual International Cryptology Conference* (2022), Springer, pp. 323–352.

[39] Gentry, C. Fully homomorphic encryption using ideal lattices. In *STOC* (2009), pp. 169–178.

[40] Goldreich, O. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing* (1987), pp. 182–194.

[41] Goldreich, O., and Ostrovsky, R. Software protection and simulation on oblivious rams. *Journal of the ACM (JACM) 43*, 3 (1996), 431–473.

[42] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM 63*, 11 (2020), 139–144.

[43] Graf, T. M., and Lemire, D. Xor filters: Faster and smaller than bloom and cuckoo filters. *Journal of Experimental Algorithmics (JEA) 25* (2020), 1–16.

[44] Ha, J., Cachet, C., Demarest, L., Ahmad, S., and Fuller, B. Private eyes: Zero-leakage iris searchable encryption. Cryptology ePrint Archive, Paper 2023/736, 2023. https://eprint.iacr.org/2023/736.

[45] Hackebeil, G. Path oram python module, 2018.
[46] Huang, Y., Wai-Kin, A. K., and Lam, K.-Y. From the perspective of CNN to adversarial iris images. In *BTAS* (2018), IEEE, pp. 1–10.
[47] Indyk, P., and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC* (1998), pp. 604–613.
[48] Ioffe, S., and Szegedy, C. 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* (2015), ICML'15, JMLR.org, p. 448–456.
[49] Jolicoeur-Martineau, A. The relativistic discriminator: a key element missing from standard GAN. In *ICLR* (2019).
[50] Kamara, S., Kati, A., Moataz, T., Schneider, T., Treiber, A., and Yonli, M. Sok: Cryptanalysis of encrypted search with leaker–a framework for leakage attack evaluation on real-world data. In *EuroS&P* (2022), IEEE, pp. 90–108.
[51] Kohli, N., Yadav, D., Vatsa, M., Singh, R., and Noore, A. Synthetic iris presentation attack using idcgan. In *2017 IEEE International Joint Conference on Biometrics (IJCB)* (2017), IEEE, pp. 674–680.
[52] Kornaropoulos, E. M., Papamanthou, C., and Tamassia, R. Data recovery on encrypted databases with k-nearest neighbor query leakage. In *IEEE S&P* (2019), IEEE, pp. 1033–1050.
[53] Kornaropoulos, E. M., Papamanthou, C., and Tamassia, R. The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution. In *2020 IEEE Symposium on Security and Privacy (SP)* (2020), IEEE, pp. 1223–1240.
[54] Kuzu, M., Islam, M. S., and Kantarcioglu, M. Efficient similarity search over encrypted data. In *2012 IEEE 28th International Conference on Data Engineering* (2012), IEEE, pp. 1156–1167.
[55] Lamoureux, M., Nickerson, B., et al. On the equivalence of b-trees and deterministic skip list.
[56] Larsen, K. G., Malkin, T., Weinstein, O., and Yeo, K. Lower bounds for oblivious near-neighbor search. In *SODA* (2020), SIAM, pp. 1116–1134.
[57] Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., and Lou, W. Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM, 2010 Proceedings IEEE* (2010), IEEE, pp. 1–5.
[58] Lin, W.-K., Mook, E., and Wichs, D. 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* (New York, NY, USA, 2023), STOC 2023, Association for Computing Machinery, p. 595–608.
[59] Liu, Q., Peng, Y., Wu, J., Wang, T., and Wang, G. Secure multi-keyword fuzzy searches with enhanced service quality in cloud computing. *IEEE Transactions on Network and Service Management 18*, 2 (2020), 2046–2062.
[60] Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (2013), vol. 30, p. 3.
[61] Mai, G., Cao, K., Yuen, P. C., and Jain, A. K. On the reconstruction of face images from deep face templates. *IEEE transactions on pattern analysis and machine intelligence 41*, 5 (2018), 1188–1202.
[62] Okada, H., Player, R., Pohmann, S., and Weinert, C. Towards practical doubly-efficient private information retrieval. In *Financial Cryptography* (2024).
[63] Osadchy, M., Pinkas, B., Jarrous, A., and Moskovich, B. SCiFI - a system for secure face identification. In *2010 IEEE Symposium on Security and Privacy* (2010), pp. 239–254.
[64] Pappas, V., Krell, F., Vo, B., Kolesnikov, V., Malkin, T., Choi, S. G., George, W., Keromytis, A., and Bellovin, S. Blind seer: A scalable private dbms. In *IEEE S&P* (2014), IEEE, pp. 359–374.
[65] Phillips, P. J., Bowyer, K. W., Flynn, P. J., Liu, X., and Scruggs, W. T. The iris challenge evaluation 2005. In *BTAS* (2008), IEEE, pp. 1–8.
[66] Phillips, P. J., Scruggs, W. T., O'Toole, A. J., Flynn, P. J., Bowyer, K. W., Schott, C. L., and Sharpe, M. FRVT 2006 and ICE 2006 large-scale experimental results. *IEEE transactions on pattern analysis and machine intelligence 32*, 5 (2009), 831–846.
[67] Pugh, W. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM 33*, 6 (1990), 668–676.
[68] Rindal, P., and Schoppmann, P. Vole-psi: Fast oprf and circuit-psi from vector-ole. In *Advances in Cryptology – EUROCRYPT 2021* (Cham, 2021), A. Canteaut and F.-X. Standaert, Eds., Springer International Publishing, pp. 901–930.
[69] Simhadri, S., Steel, J., and Fuller, B. Cryptographic authentication from the iris. In *International Conference on Information Security* (2019), Springer, pp. 465–485.
[70] Soleymani, S., Dabouei, A., Dawson, J., and Nasrabadi, N. M. Adversarial examples to fool iris recognition systems. In *2019 International Conference on Biometrics (ICB)* (2019), IEEE, pp. 1–8.
[71] Song, D. X., Wagner, D., and Perrig, A. Practical techniques for searches on encrypted data. In *IEEE S&P* (2000), IEEE, pp. 44–55.
[72] Stefanov, E., Dijk, M. V., Shi, E., Chan, T.-H. H., Fletcher, C., Ren, L., Yu, X., and Devadas, S. Path oram: an extremely simple oblivious ram protocol. *Journal of the ACM (JACM) 65*, 4 (2018), 1–26.
[73] Uzun, E., Chung, S. P., Kolesnikov, V., Boldyreva, A., and Lee, W. Fuzzy labeled private set intersection with applications to private real-time biometric search. In *USENIX Security* (2021), pp. 911–928.
[74] Venugopalan, S., and Savvides, M. How to generate spoofed irises from an iris code template. *IEEE Transactions on Information Forensics and Security 6*, 2 (2011), 385–395.
[75] Wang, B., Yu, S., Lou, W., and Hou, Y. T. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *IEEE INFOCOM 2014-IEEE conference on computer communications* (2014), IEEE, pp. 2112–2120.
[76] Wang, J., Ma, H., Tang, Q., Li, J., Zhu, H., Ma, S., and Chen, X. Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. *Comput. Sci. Inf. Syst. 10*, 2 (2013), 667–684.
[77] Wang, X. S., Nayak, K., Liu, C., Chan, T. H., Shi, E., Stefanov, E., and Huang, Y. Oblivious data structures. In *CCS* (2014), pp. 215–226.
[78] Williams, P., Sion, R., and Tomescu, A. Privatefs: A parallel oblivious file system. In *CCS* (2012), pp. 977–988.
[79] Yadav, S., Chen, C., and Ross, A. 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 Building OMC from PSI and pseudorandom permutations

Private set intersection (PSI) [32] 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. In the full version of this work [44] 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.

B Synthetic Data Generation

We briefly describe the neural network used to produce our synthetic templates, see our full version [44]. Our synthetic templates are built using a generative adversarial network or GAN. Yadav et al. [79] uses RaSGAN (relativistic average standard GAN) [49] 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. [51] use the DCGAN architecture to generate synthetic irises. Synthetic irises can be viewed as irises that must closely resemble bonafide irises as discussed in [51, 79].

We follow the approach of RESIST [3] which takes inspiration from synthetic data generation to invert iris templates into realistic looking images. We use a recently proposed relativistic average discriminator [49] as our discriminator. SYNTH architecture is a small neural network having only dense (fully connected) layers, there are five generator layers, and three discriminator layers. Each layer is followed by a LeakyReLU [60] activation and a batch normalization [48] layer. The last layers of both sub-networks are unique, the generator has a tanh activation while the discriminator has a Sigmoid activation.