Proximity Searchable Encryption for the Iris Biometric

Sohaib Ahmad* Chloe Cachet† Luke Demarest‡ Benjamin Fuller§ Ariel Hamlin¶

August 19, 2021

Abstract

Biometric databases collect people's information and allow users to perform proximity searches (finding all records within a bounded distance of the query point) with few cryptographic protections. This work studies proximity searchable encryption applied to the iris biometric.

Prior work proposed inner product functional encryption as a technique to build proximity biometric databases (Kim et al., SCN 2018). This is because binary Hamming distance is computable using an inner product. This work identifies and closes two gaps to using inner product encryption for biometric search:

1. Biometrics naturally use long vectors often with thousands of bits. Many inner product encryption schemes generate a random matrix whose dimension scales with vector size and have to invert this matrix. As a result, setup is not feasible on commodity hardware unless we reduce the dimension of the vectors. We explore state of the art techniques to reduce the dimension of the iris biometric and show that all known techniques harm the accuracy of the resulting system. That is, for small vector sizes multiple unrelated biometrics are returned in the search. For length 64 vectors, at a 90% probability of the searched biometric being returned, 10% of stored records are erroneously returned on average.

   Rather than changing the feature extractor, we introduce a new cryptographic technique that allows one to generate several smaller matrices. For vectors of length 1024 this reduces time to run setup from 30 days to 4 minutes. At this vector length, for the same 90% probability of the searched biometric being returned, .02% of stored records are erroneously returned on average.

2. Prior inner product approaches leak distance between the query and all stored records. We refer to these as distance-revealing. We show a natural construction from function hiding, secret-key, predicate, inner product encryption (Shen, Shi, and Waters, TCC 2009). Our construction only leaks access patterns, and which returned records are the same distance from the query. We refer to this scheme as distance-hiding.

We implement and benchmark one distance-revealing and one distance-hiding scheme. The distance-revealing scheme can search a small (hundreds) database in 4 minutes while the distance-hiding scheme is not yet practical, requiring 4 hours.

Keywords: Searchable encryption, biometrics, proximity search, inner product encryption.

1 Introduction

Biometrics are measurements of physical phenomena of the human body. We focus on the *iris* biometric in this work. Iris data, like all biometric data is noisy, which means that two readings from the same iris are unlikely to be identical. *Feature extractors* convert such physical phenomena to a digital representation that is more stable but still noisy. The output of feature extractors is called a *template*. Biometric databases are used for both security critical applications (such as access control) and privacy critical applications (such as immigration). Let $\mathcal{D}$ be some distance metric and t be some distance threshold. Applications building on biometric templates require:

1. **Low False Reject Rate (FRR)** templates from the same biometric are within distance t with high probability, and

*University of Connecticut. Email sohaib.ahmad@uconn.edu
†University of Connecticut. Email chloe.cachet@uconn.edu
‡University of Connecticut. Email luke.h.demarest@gmail.com
§University of Connecticut. Email benjamin.fuller@uconn.edu
¶Khoury College of Computer Sciences, Northeastern University. Email ahamlin@ccs.neu.edu

2. **Low False Accept Rate (FAR)** templates from two different biometrics are within distance t with low probability.

Learning stored biometric templates enables an attacker to reverse this value into a convincing biometric [GRGB+12, MCYJ18, AF20], enabling *presentation* attacks [VS11, HWKL18, SDDN19] that can compromise users' accounts and devices. Since biometrics cannot be updated, such a compromise lasts a lifetime.

Searchable encryption [SWP00, CGKO11, BHJP14, FVY+17] enables servers to be queried without decrypting the data. For a distance metric $\mathcal{D}$, proximity searchable encryption returns all records that are within distance t. That is, for a dataset $x_1, ..., x_\ell$ for a query y, one should return all x_i such that $\mathcal{D}(x_i, y) \leq t$. Since biometric data is inherently noisy, proximity searchable encryption is a key tool to secure biometric databases while allowing queries.

Iris feature extractors usually produce binary vectors that are similar in Hamming distance (fingerprints are usually compared by set difference, faces with L2 norm). Kim et al. proposed to use secret-key, function-hiding *inner product encryption* or $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk}}$ for encrypted comparison of binary Hamming biometrics [KLM+16]. $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk}}$ allows computation of inner product without revealing underlying values. Inner product of vectors x, y in $\{-1, 1\}^n$ encodes Hamming distance:

$$\mathcal{D}(x, y) = (n - \langle x, y \rangle)/2.$$

More formally the functionality of $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk}}$ is as follows: one generates $\mathsf{sk} \leftarrow \mathsf{Setup}(\cdot)$ and has two algorithms $\mathsf{ct}_x \leftarrow \mathsf{Encrypt}(x, \mathsf{sk})$ and $\mathsf{tk}_y \leftarrow \mathsf{TokGen}(y, \mathsf{sk})$ such that one can use $\mathsf{Decrypt}$ (without sk) to learn $\langle x, y \rangle$. That is, $\mathsf{Decrypt}(\mathsf{ct}_x, \mathsf{tk}_y) = \langle x, y \rangle$. One can use $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk}}$ to build proximity search by encrypting $c_i \leftarrow \mathsf{Encrypt}(x_i, \mathsf{sk})$ and providing all c_i to the database server (additional data can be associated with x_i using traditional encryption). For queries y the client provides $\mathsf{tk}_y \leftarrow \mathsf{TokGen}(y, \mathsf{sk})$ to the server. The server can compute the inner product between the query and each stored record and should return all records with the appropriate inner product.

We identify and close two gaps in the use of inner product encryption to build proximity searchable encryption for the iris.

1.1 Our Contribution

Multi Random Projection Inner Product Encryption Daugman's seminal iris feature extractor [Dau05, Dau09] produces a vector of length $n = 1024$, the open source OSIRIS [ODGS16] system uses $n = 32768$ by default, and recent neural network feature extractors [AF19] use $n = 2048$.

The most efficient $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk}}$ schemes rely on *dual pairing vector spaces* [OT15] in bilinear groups. The secret key for such constructions is a random matrix $\mathbf{A} \in \mathbb{F}_q^{n \times n}$ and its inverse $\mathbf{A}^{-1}$; q is a large prime that is the order of the bilinear pairing. Setup for the scheme must invert a random $\mathbf{A} \in \mathbb{F}_q^{n \times n}$.

This operation is prohibitive for $n > 1000$, as is the case for iris feature extractors. For the most efficient known scheme which we call *Random Projection with Check* or RProjC [KLM+18], the authors' parallel implementation of key generation in FLINT [Har10] (on a modern 16 core machine), generating keys for $n = 240$ took roughly four hours. In our experiments, Setup time grows cubicly as expected.[1] Through interpolation, we estimate the time to generate keys for $n = 1024$ at 23 days.

While one can train feature extractors with smaller n, we show (in Section 3) that known techniques harm the quality of the biometric features, making the irises of different people appear similar. The false accept vs false reject rate tradeoff degrades, leaving the application with the choice of either not matching readings of the same iris or matching readings of difference individuals' irises. Both choices have consequences for the resulting application.

In Section 3.1 and Table 3, we show that for a small size dataset of 356 individuals using a feature extractor with $n = 64$ a distance t that enables a 90% true accept rate searching for an individual in the dataset returns 40 incorrect biometrics with an average query! By comparison when $n = 1024$, queries return .06 incorrect biometrics on average. Datasets with more individuals are not available; we expect this rate to be consistent across dataset sizes.

Section 4 introduces a new transform for inner product encryption that generates multiple matrices $\mathbf{A}_1, ..., \mathbf{A}_\sigma$ and their inverses during key generation where each $\mathbf{A}_i$ is a $(\lceil n/\sigma \rceil + 1) \times (\lceil n/\sigma \rceil + 1)$ instead of a single large pair $\mathbf{A}, \mathbf{A}^{-1}$. To hide partial information, both x and y are augmented when they are split into component vectors:

$$x'_i = 1 \mid\mid x_{i*\lceil n/\sigma \rceil}, ..., x_{i*\lceil n/\sigma \rceil + (\lceil n/\sigma \rceil - 1)}$$
$$y'_i = \zeta_i \mid\mid y_{i*\lceil n/\sigma \rceil}, ..., y_{i*\lceil n/\sigma \rceil + (\lceil n/\sigma \rceil - 1)}$$

[1] We have not evaluated sub-cubic matrix inversion in finite fields.

Scheme Name	Underlying IPE	IPE Type fh	sk	pred	Multi Random Proj Applied	Distance Hiding	Operation Time Setup	BuildIndex	Trapdoor	Search
RProjC	[KLM+18]	✓	✓	–	–	–	2×10^6	10.8	.07	235
MRProjC	[KLM+18]	✓	✓	–	✓	–	268	10.8	.08	241
MRProj	[BCSW19]	✓	✓	✓	✓	✓	268	10.8	22.4	12600

Table 1: Time (in seconds) for operations with $\ell = 356$ records stored at $n = 1024$. All algorithms are naturally parallelizable. Timing for the single base scheme is interpolated from smaller vector lengths. BuildIndex encrypts the dataset at initialization time, Trapdoor generates a search token, and Search finds the resulting indices.

for $i = 0, ..., \sigma - 1$ and $\zeta_0, ..., \zeta_{\sigma-1}$ is a linear secret sharing of 0 that is chosen in TokGen. The intuition is that any collection of $\sigma - 1$ or fewer components represents a random group element, so one cannot learn information about inner products between vector components. We show security of two prior IPE schemes with multi random projection (one in Section 4 and one in the full version of this work [Red]).

We implemented two versions of proximity search building on this form of $\mathsf{IPE}_{\mathsf{fh,sk}}$. The first is a direct application of the RProjC [KLM+18] scheme and the second is our new *multi random projection* version, called *Multi Random Projection with Check* or MRProjC. To benchmark, we encrypted a single reading of each individual ($\ell = 356$) from the ND0405 dataset [PSO+09, BF16] which is a superset of the NIST Iris Evaluation Challenge [PBF+08]. Queries are drawn from other readings in the ND0405 dataset. This performance is summarized in Table 1 with search taking approximately 4 minutes. Our multi random projection technique reduces time for Setup by **four orders of magnitude** while only slightly increasing the time of creating the search token and search. This *multi random projection* technique makes proximity searchable encryption on a 350 biometric dataset feasible.

Distance Hiding Proximity Search By design, the proximity search from $\mathsf{IPE}_{\mathsf{fh,sk}}$ for any searched value y, allows the server to compute the distance [KLM+18] between y and *all* stored records.[2] This establishes a geometry on the space of stored records. If the server has side information on the stored records x_i, they may be able to reconstruct global geometry from the local geometry revealed by pairwise distances [PBDT05, AEG+06]. While we are not aware of any leakage abuse attacks directly against proximity search, there are attacks against k-nearest neighbor databases [KPT19, KE19].[3] Distance allows one to easily compute the k-nearest points (with some error) so attacks that can exploit this leakage apply. Like most leakage abuse attacks, the efficacy of these attacks depends on what the adversary knows about the stored data. We discuss these attacks more in Section 7.

For applications where such leakage is unacceptable (or the adversary has side information on the encrypted data), we show a transform from a *predicate* version of inner product encryption to proximity search that does not reveal pairwise distance. A predicate IPE scheme produces ciphertexts c_x and tokens tk_y which allow one to effectively check if $\langle x, y\rangle = 0$ (instead of revealing the inner product). Barbosa et al. [BCSW19] recently proposed such a scheme that is a modification of Kim et al.'s construction [KLM+18]. Their construction simply removes the group elements that allow one to check the inner product, so we call this *Random Projection* or RProj. We call such a scheme an $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ scheme. $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ allows one to test if the inner product is equal to some value i as follows: add an $n + 1^{th}$ element as -1 to x, denoted x', and create $y_i = y||i$. Then,

$$\langle x', y_i\rangle = (\langle x \;||\; \text{-}1, y \;||i\rangle = 0) \Leftrightarrow (\langle x, y\rangle = i)\,.$$

One can check all values in a set $\mathcal{I}$ by generating a tokens tk_{y_i} for each $i \in \mathcal{I}$. Setting $\mathcal{I} = \{n - 2 * 0, ..., n - 2 * t\}$, yields a proximity check (these tokens are permuted before being sent to server).

We call this construction *Multi Random Projection* or MRProj. The simplicity and generality of this construction is an advantage, it immediately benefits from efficiency improvements in inner product encryption and can be built from multiple computational assumptions. However, the size of tk_y and search time grow linearly with t. For the iris t is usually around $.3n$.

Since the server can see if the same tk_{y_i} matches different records, when two records are both within distance t, the server learns if they match the same distance (but not the specific distance). Thus, the resulting proximity scheme leaks two pieces of information:

[2]Some prior work allows computation of approximate distance [KIK12] using locality sensitive hashes [IM98], allowing the server to see how many hashes match, the number of matches approximates distance.

[3]Here we focus on attacks that apply to proximity searchable encryption. There is a rich history of *leakage abuse attacks* against different types of searchable encryption [IKK12,CGPR15,KKNO16,WLD+17,GSB+17,GLMP18,KPT19,MT19,KE19,KPT20,FMC+20].

Access Pattern [IKK12, CGPR15] The set of records returned by the query. If x_i and x_j are both returned by a query it must be the case that $\mathcal{D}(x_i, x_j) \leq 2t$. Preventing attacks that only require access pattern usually requires oblivious RAM [GKL+20] and its high storage and communication overhead.

Distance Equality Leakage For a database of records $x_1, ..., x_\ell$ for a searched value y if there are multiple records x_i, x_j such that $\mathcal{D}(x_i, y) \leq t$ and $\mathcal{D}(x_j, y) \leq t$ our scheme additionally reveals if $\mathcal{D}(x_i, y) = \mathcal{D}(x_j, y)$.

No information is leaked about data that is not returned (beyond that it was not returned). Biometrics are well spread, so one does not expect readings of two biometrics to be close to a query. As mentioned, the vector size has a large impact on the number of improper records that will be returned by a query (recall for $n = 64$, 40 improper records are returned, when $n = 1024$, .06 improper records are returned). Since MRProj only leaks when multiple records are returned it is critical to ensure an accurate system, underscoring the importance of our multi random projection approach enabling Setup for large n where high correctness is possible.

In RProjC and MRProjC, the server learns the pairwise distance between the query y and all records x_i. So in that setting, n only affects correctness, not security.

The search complexity of MRProj is roughly a multiplicative of $t \approx .3n$ slower than for MRProjC. See the difference in concrete timing in Table 1. For $n = 1024$ this corresponds to a $t \approx 307$, the measured multiplicative overhead is only 52.5. Closing this performance gap is the main open problem resulting from this work; MRProj is not fast enough. In Section 8 we present avenues for improving search efficiency.

Organization The rest of this work is organized as follows, Section 2 describes mathematical and cryptographic preliminaries, Section 3 describes the n vs accuracy tradeoff for the iris and its impact on security, Section 4 introduces the multi random projection technique, Section 5 shows that $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ suffices to build proximity search, Section 6 discusses our implementation, Section 7 reviews further related work and Section 8 concludes.

2 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. 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 $q = q(\lambda) \in \mathbb{N}$ be a prime, then $\mathbb{G}_q$ denotes a cyclic group of order q. Let x denote a vector over $\mathbb{Z}_q$ such that $x = (x_1, \cdots, x_n) \in \mathbb{Z}_q^n$, the dimension of vectors should be apparent from context. Consider vectors $x = (x_1, \cdots, x_n)$ and $v = (v_1, \cdots, v_n)$, their inner-product is denoted by $\langle x, v \rangle = \sum_{i=1}^{n} x_i v_i$. Let X be a matrix, then X^T denotes its transpose.

Hamming distance is defined as the distance between the bit vectors x and y of length n: $\mathcal{D}(x, y) = |\{i \,|\, x_i \neq y_i\}|$. We note that if a vector over $\{0, 1\}$ as is encoded as $x_{\pm 1,i} = 1$ if and $x_i = 1$ and $x_{\pm 1,i} = -1$ if $x_i = 0$ then it is true that $\langle x_{\pm 1}, y_{\pm 1} \rangle = n - 2\mathcal{D}(x, y)$.

Definition 1 (Asymmetric Bilinear Group)**.** *Suppose $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ are three groups (respectively) of prime order q with generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $g_T \in \mathbb{G}_T$ respectively. We denote a value x encoded in $\mathbb{G}_1$ with either g_1^x or $[x]_1$, we denote $\mathbb{G}_2$ similarly. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a non-degenerate (i.e. $e(g_1, g_2) \neq 1$) bilinear pairing operation such that for all $x, y \in \mathbb{Z}_q$, $e([x]_1, [y]_2) = e(g_1, g_2)^{xy}$. We assume the group operations in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ and the pairing operation e are efficiently computable, then $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ defines an asymmetric bilinear group.*

Let $\mathcal{G}_{abg}$ be an algorithm that takes input 1^λ and outputs a description of an asymmetric bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ with security parameter λ.

2.1 Inner Product Encryption

Secret-key predicate encryption with function privacy supporting inner products queries was first proposed by Shen et al. [SSW09]. This primitive allows one to check if the inner product between vectors is zero or not. The scheme they presented is both attribute and function hiding, meaning that an adversary running the decryption algorithm gains no knowledge on either the attribute or the predicate.

Definition 2 (Secret key predicate encryption)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, $\mathcal{M}$ be the set of attributes and $\mathcal{F}$ be a set of predicates. We define $\mathsf{PE} = (\mathsf{PE.Setup}, \mathsf{PE.Encrypt}, \mathsf{PE.TokGen}, \mathsf{PE.Decrypt})$, a secret-key predicate encryption scheme, as follows:*

1. Draws $\beta \xleftarrow{\$} \{0,1\}$,
2. Computes $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{PE.Setup}(1^\lambda)$, sends pp to $\mathcal{A}$,
3. For $1 \le i \le s$, $\mathcal{A}$ chooses $x_i^{(0)}, x_i^{(1)} \in \mathcal{M}$,
4. For $1 \le j \le r$, $\mathcal{A}$ chooses $f_j^{(0)}, f_j^{(1)} \in \mathcal{F}$,
5. Denote $R := \left(x_1^{(0)}, x_1^{(1)}\right), \cdots, \left(x_r^{(0)}, x_r^{(1)}\right)$, $S := \left(f_1^{(0)}, f_1^{(1)}\right), \cdots, \left(f_s^{(0)}, f_s^{(1)}\right)$.
6. $\mathcal{A}$ sends R and S to $\mathcal{C}$,
7. $\mathcal{A}$ loses the game if R and S are not admissible,
8. $\mathcal{A}$ receives $C^{(\beta)} := \{ct_i^{(\beta)} \leftarrow \mathsf{PE.Encrypt}\left(\mathsf{sk}, x_i^{(\beta)}\right)\}_{i=1}^r$ and $T^{(\beta)} := \{tk_j^{(\beta)} \leftarrow \mathsf{PE.TokGen}\left(\mathsf{sk}, f_j^{(\beta)}\right)\}_{j=1}^s$
9. $\mathcal{A}$ returns $\beta' \in \{0,1\}$,
10. Her *advantage* is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Exp}_{IND}^{\mathsf{PE}}}(\lambda) = \left| \Pr[\mathcal{A}(1^\lambda, T^{(0)}, C^{(0)}) = 1] - \Pr[\mathcal{A}(1^\lambda, T^{(1)}, C^{(1)}) = 1] \right|$

Figure 1: Definition of $\mathsf{Exp}_{IND}^{\mathsf{PE}}$ for predicate encryption.

- $\mathit{PE.Setup}(1^\lambda) \to (\mathit{sk}, \mathit{pp})$,
- $\mathit{PE.Encrypt}(\mathit{sk}, x) \to \mathit{ct}_x$,
- $\mathit{PE.TokGen}(\mathit{sk}, f) \to \mathit{tk}_f$,
- $\mathit{PE.Decrypt}(\mathit{pp}, \mathit{tk}_f, \mathit{ct}_x) \to b$.

We require the scheme to have the following properties:

Correctness: *For any $x \in \mathcal{M}, f \in \mathcal{F}$,*

$$\Pr\left[f(x) = b \;\middle|\; \begin{array}{c} \mathit{ct}_x \leftarrow \mathit{PE.Encrypt}(\mathit{sk}, x) \\ \mathit{tk}_f \leftarrow \mathit{PE.TokGen}(\mathit{sk}, f) \\ b \leftarrow \mathit{PE.Decrypt}(\mathit{pp}, \mathit{tk}_f, \mathit{ct}_x) \end{array} \right] \ge 1 - \mathit{negl}(\lambda).$$

Security of admissible queries: *Let $r = \mathtt{poly}(\lambda)$ and $s = \mathtt{poly}(\lambda)$. Any PPT adversary $\mathcal{A}$ has only $\mathit{negl}(\lambda)$ advantage in the Exp_{IND}^{PE} game (defined in Figure 1). Token and encryption queries must meet the following admissibility requirements, $\forall j \in [1, r], \forall i \in [1, s]$,*

$$\mathit{PE.Decrypt}(\mathit{pp}, \mathit{tk}_j^{(0)}, \mathit{ct}_i^{(0)}) = \mathit{PE.Decrypt}(\mathit{pp}, \mathit{tk}_j^{(1)}, \mathit{ct}_i^{(1)}).$$

The above definition is called *full security* in the language of Shen, Shi, and Waters [SSW09]. Note that this definition is selective (not adaptive), as the adversary specifies two sets of plaintexts and functions apriori. The relevant primitive for us is $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk},\mathsf{pred}}$ which uses the above definition restricted to the class of predicates $\mathcal{F} = \{f_y \mid y \in \mathbb{Z}_q^n\}$ be the set of predicates such that for all vectors $x \in \mathbb{Z}_q^n$, $f_y(x) = 1$ when $\langle x, y \rangle = 0$, $f_{y,t}(x) = 0$ otherwise. We use $(\mathsf{IPE.Setup}, \mathsf{IPE.Encrypt}, \mathsf{IPE.TokGen}, \mathsf{IPE.Decrypt})$ to refer to the corresponding tuple of algorithms.

2.2 Proximity searchable encryption

In this section we define *proximity searchable encryption (PSE)*, a variant of searchable encryption that supports proximity queries.

Definition 3 (History)**.** *Let $X \in \mathcal{M}$ be a list of keywords drawn from space $\mathcal{M}$, let $\mathcal{F}$ be a class of predicates over $\mathcal{M}$. An m-query history over $\mathcal{W}$ is a tuple $\mathit{History} = (X, F)$, with $F = (f_1, \cdots, f_m)$ a list of m predicates, $f_i \in \mathcal{F}$.*

Definition 4 (Access pattern)**.** *Let $X \in \mathcal{M}$ be a list of keywords. The access pattern induced by an m-query history $\mathit{History} = (X, F)$ is the tuple $\mathit{AccPatt}(\mathit{History}) = (f_1(X), \cdots, f_m(X))$*

1. Draws $\beta \xleftarrow{\$} \{0,1\}$,
2. Computes $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{PSE.Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$.
3. $\mathcal{A}$ chooses and outputs $\mathsf{History}^{(0)}, \mathsf{History}^{(1)}$.
4. $\mathcal{A}$ loses the game if $\mathsf{AccPatt}(\mathsf{History}^{(0)}) \neq \mathsf{AccPatt}(\mathsf{History}^{(1)}) \vee \ \mathsf{DisEq}(\mathsf{History}^{(0)}, \mathsf{History}^{(1)}) = 0$
5. $\mathcal{A}$ receives $I^{(\beta)}$ and $Q^{(\beta)}$.
6. $\mathcal{A}$ outputs $\beta' \in \{0,1\}$
7. Her advantage in the game is: $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Exp}_{IND}^{\mathsf{PSE}}}(\lambda) = \left| \ \Pr[\mathcal{A}(1^\lambda, I^{(0)}, Q^{(0)}) = 1] - \Pr[\mathcal{A}(1^\lambda, I^{(1)}, Q^{(1)}) = 1] \ \right|$

Figure 2: Definition of $\mathsf{Exp}_{IND}^{\mathsf{PSE}}$ for proximity searchable encryption.

Definition 5 (Distance Equality)**.** *Let* $\mathsf{History}^{(0)}$ *and* $\mathsf{History}^{(1)}$ *be two* m*-query histories for predicates of the type* $f_{y,t}(x) = (\mathcal{D}(x,y) \overset{?}{\leq} t)$*. Let,* $\mathsf{DisEq}(\mathsf{History}^{(0)}, \mathsf{History}^{(1)}) = 1$ *if and only if for each* j *it is true that*

$$\left\{ (i,k) \middle| \begin{matrix} (\mathcal{D}(x_i^{(0)}, y_j^{(0)}) = \mathcal{D}(x_k^{(0)}, y_j^{(0)}) \wedge \mathcal{D}(x_i^{(1)}, y_j^{(1)}) \neq \mathcal{D}(x_k^{(1)}, y_j^{(1)})) \\ \vee \\ (\mathcal{D}(x_i^{(0)}, y_j^{(0)}) \neq \mathcal{D}(x_k^{(0)}, y_j^{(0)}) \wedge \mathcal{D}(x_i^{(1)}, y_j^{(1)}) = \mathcal{D}(x_k^{(1)}, y_j^{(1)})) \end{matrix} \right\},$$

is the empty set.

Definition 6 (Proximity Searchable Encryption)**.** *Let*

- $\lambda \in \mathbb{N}$ *be the security parameter,*
- $\mathcal{DB} = (M_1, \cdots, M_\ell)$ *be a database of size* ℓ*,*
- *Keywords* $X = (x_1, \cdots, x_\ell)$*, such that* $x_i \in \mathbb{Z}_q^n$ *relates to* M_i*,*
- $\mathcal{F} = \{f_{y,t} \mid y \in \mathbb{Z}_q^n, \ t \in N\}$ *be a family of predicates such that, for a keyword* $x \in \mathbb{Z}_q^n$*,* $f_{y,t}(x) = 1$ *if* $\mathcal{D}(x,y) \leq t$*,* 0 *otherwise.*

The tuple of algorithms $\mathsf{PSE} = (\mathsf{PSE.Setup}, \mathsf{PSE.BuildIndex}, \mathsf{PSE.Trapdoor}, \mathsf{PSE.Search})$ *defines a proximity searchable encryption scheme:*

- $\mathsf{PSE.Setup}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pp})$*,*
- $\mathsf{PSE.BuildIndex}(\mathsf{sk}, X) \rightarrow I_X$*,*
- $\mathsf{PSE.Trapdoor}(\mathsf{sk}, f_{y,t}) \rightarrow \mathsf{tk}_{y,t}$*,*
- $\mathsf{PSE.Search}(\mathsf{pp}, Q_{y,t}, I_X) \rightarrow J_{X,y,t}$*.*

We require the scheme to have the following properties:

Correctness *Define* $J_{X,y,t} = \{i | f_{y,t}(x_i) = 1, x_i \in X\}$*.* PSE *is correct if for all* X *and* $f_{y,t} \in \mathcal{F}$*:*

$$\Pr\left[J' = J_{X,y,t} \middle| \begin{matrix} I_X \leftarrow \mathsf{PSE.BuildIndex}(\mathsf{sk}, X) \\ Q_{y,t} \leftarrow \mathsf{PSE.Trapdoor}(\mathsf{sk}, f_{y,t}) \\ J' \leftarrow \mathsf{PSE.Search}(\mathsf{pp}, Q_{y,t}, I_X) \end{matrix} \right] \geq 1 - \mathsf{negl}(\lambda).$$

Security for Admissible Queries *Any PPT adversary* $\mathcal{A}$ *has only* $\mathsf{negl}(\lambda)$ *advantage in the experiment* $\mathsf{Exp}_{IND}^{\mathsf{PSE}}$ *defined in Figure 2, for* $\ell = \mathtt{poly}(\lambda)$ *and* $m = \mathtt{poly}(\lambda)$*.*

3 Iris Statistics and Leakage

This section introduces iris feature extractors and shows that reducing the length of the feature extractor harms the uniqueness of the resulting biometric. Reduced uniqueness harms both the correctness (because the wrong set of irises is returned) and security of the MRProj construction (because the server learns information about returned irises). Daugman [Dau05, Dau09] introduced the seminal iris processing pipeline. This pipeline assumes a near infrared camera. Iris images in near infrared are believed to be independent from the visible light pattern; the near-infrared iris pattern is epigenetic, irises of identical twins are believed to be independent [Dau09, HBF10]. Traditional iris recognition consists of three phases:

Segmentation takes the image and identifies which pixels should be included as part of the iris. This produces a $\{0,1\}$ matrix of the same size as the input image with 1s corresponding to iris pixels.

Normalization takes the variable size set of iris pixels and maps them to a fixed size rectangular array. This can roughly be thought of as unrolling the iris.

Feature Extraction transforms the rectangular array into a fixed number of features. In Daugman's original work this consisted of convolving small areas of the rectangle with a 2D wavelet. Modern feature extractors are usually convolutional neural networks.

In identification systems the tradeoff is between FRR and FAR. FRR is how frequently readings of the *same* biometric are regarded as different. FAR is how frequently readings of *different* biometrics are regarded as the same. As described above, when one wishes to match a biometric y against a database one considers matches as the set $\{x_i | \mathcal{D}(x_i, y) \leq t\}$ for some metric $\mathcal{D}$ and distance parameter t. Selecting a small t increases FRR and reduces FAR. Before investigating the dependence on feature vector length and the FRR/FAR tradeoff we introduce the feature extractor and dataset used in this analysis.

Feature Extractor For the feature extractor, we use the recent pipeline called ThirdEye [AF18, AF19], which is publicly available [Ahm20]. The software produces a 1024 dimensional real valued feature vector. We convert this to a binary vector by setting $f'_i = 1$ if $f_i > \mathsf{Exp}[f_i]$ where the $\mathsf{Exp}[f_i]$ is the expectation of the individual feature, otherwise $f'_i = 0$. We train the feature extractor as specified in [AF19].

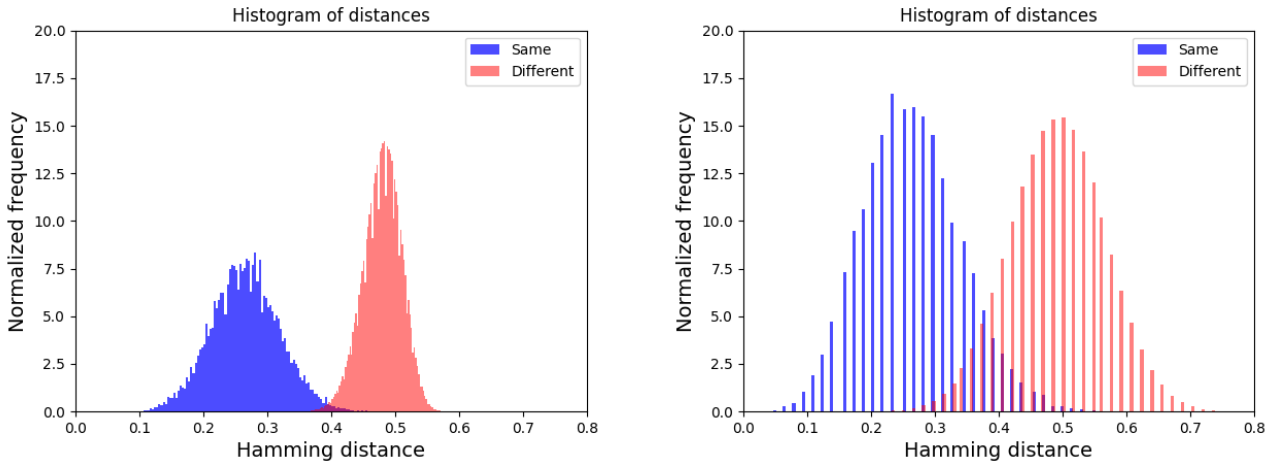
Biometric Database There are many iris datasets collected across a variety of conditions. In this work we use the NotreDame 0405 dataset [PSO+09, BF16] which is a superset of the NIST Iris Evaluation Challenge [PBF+08]. This dataset consists of images from 356 biometrics (we consider left and right eyes as separate biometrics) with 64964 images in total.[4] Figure 3(a) shows the histograms for the testing portions of the feature extractor outputs. The blue histogram contains comparisons between different readings of the same biometric while the red histogram contains comparisons between different biometrics. Let $t' = t/1024$ be the fractional Hamming distance, the FRR is the fraction of the blue histogram to the right of t' and the FAR is the fraction of the red histogram to the left of t'. There is overlap between the red and blue histogram indicating that there is a tradeoff between FRR and FAR.

3.1 Performance of Biometric Identification with Small Dimension

The efficiency of IPE based proximity search critically depends on the number of features n (see Table 4). In our experiments we estimate Setup for $n = 1024$ for the schemes of Kim et al. [KLM+18] and Barbosa et al. [BCSW19] to take 30 days on a modern server machine (see details in Section 6). It is tempting to consider statistical methods to produce feature vectors of reduced size. We show this comes at a cost to the quality of the resulting feature vectors. This motivates our approach to reduce the complexity of Setup in Section 4. Our analysis consists of two major parts:

1. We compare different mechanisms for reducing the size of feature vectors using $n = 64$ as the target dimension.
2. Using the best feature reduction mechanism we compare the FRR/FAR tradeoff for $n < 1024$, showing direct impacts for the correctness and security of the resulting biometric search.

[4] We obtained similar results with the IITD dataset [KP10] which consists of 224 persons and 2240 images. The IITD dataset is considered "easier" than the ND0405 dataset because images are collected in more controlled environments leading to less noise and variation between images.



(a) ND 0405 Histogram with $n = 1024$

(b) ND 0405 Histogram with $n = 64$

Figure 3: Hamming distance distribution for images from the same iris in blue, and different irises in red. Histograms are produced using ThirdEye [AF19]. Resulting histograms for the ND 0405 dataset. Figure 3(a) shows the histogram when $n = 1024$ with a small overlap between distances comparisons of the same iris and different irises. This overlaps is increased substantially when $n = 64$ in in Figure 3b). Figure 3b) is produced using the E method.

3.1.1 Dimensionality Reduction Method

We consider four different mechanisms for dimension reduction and consider their impact on FRR/FAR. For all techniques, the most important phenomena is that variance of Different comparisons increases as the sample size decreases.[5] Compare Figure 3(a) and Figure 3(b). This makes the tails of Same and Different wider leading to worse identification. The four mechanisms we consider are:[6]

Random Sample Select a random subset of positions of size 64 and use this as the feature extractor. We denote this technique by R-64 (for random).

Error Rate Minimization Hollingsworth et al. [HBF08] and Bolle et al. [BPCR04] propose the concept of "fragile bits" which are more likely to be susceptible to bit flips. Their work is based on the Gabor based feature extractor (described at the beginning of this section) while ThirdEye [AF19] is a convolutional neural network.

We utilize 64 most "stable" bits which have the least probability of flipping or have the lowest variance. Results for this approach are shown in Table 2 and denoted by S-64 (for stable).

Surprisingly, this approach is worse than random sampling. We believe this approach to be appropriate for the Gabor based feature extractor since it produces large number of noisy features due to noise in different readings of an iris. This is in contrast to our feature extractor which outputs a succinct feature vector where the CNN tries to make individuals features independent.

Error Delta Maximization This approach uses bits which maximize the difference between the means of the intra and inter class distributions. That is, these are bits where the difference between intra class and inter class error is the highest. That is, we select the bits that maximize the following difference:

$$\max_i \left(\Pr_{x,y \leftarrow \text{Different}}[x_i \neq y_i] - \Pr_{x,y \leftarrow \text{Same}}[x_i \neq y_i] \right)$$

The intuition is that bits are the most useful as they maximize the difference in probability of error between the same and different comparisons. The hope is to overcome the weakness of the prior approach which did

[5] This is consistent with previous observations that sampling from the iris red histogram behaves similarly to a binomial distribution where the number of trials is proportional the included entropy of the iris [SSF19].

[6] For all experiments we computed the mechanism four times and report the average in Table 2.

FRR at Size	False Accept Rate 0	.01	.02	.03	.04	.05	.06	.07	.08	.09	.10
1024	.50	.03	.02	.01	.01	.01	.01	.01	.01	0	0
R-64	.99	.38	.29	.24	.22	.18	.17	.16	.14	.13	.12
S-64	1	.61	.61	.51	.41	.41	.41	.32	.32	.32	.26
E-64	.97	.30	.24	.18	.14	.14	.10	.10	.10	.07	.07
T-64	.96	.27	.16	.13	.13	.09	.09	.06	.06	.06	.04

Table 2: FRR for different output sizes and probabilities of leakage for the ND0405 datasets. Summary of false reject rates for queries drawn from *Same* distribution. We vary a threshold t, report the false reject rate (FRR) when allowing for the corresponding FAR. The original $n = 1024$ system is presented for comparison.

not consider the entropy of bits across different biometrics. The top 64 bits are used. This approach is denoted by E-64 (for error). This approach improves over both R and S techniques.

Training Network Lastly, we train the ThirdEye architecture [AF19] from scratch to output a smaller feature vector of size $n = 64$ for both datasets. Essentially we train a new feature extractor on the same training data to reduce dimensions. The feature extractor remains the same but is now constrained to learn 64 features. This is achieved by changing the number of neurons in the second last layer of our convolutional neural network. We can expect this to perform better than random sampling since the feature extractor is explicitly learning to classify using 64 features. We use T (for train) to denote this technique.

Results are summarized in Table 2. The E and T techniques outperform the R and S techniques. Going forward we use the E dimensionality reduction technique for the rest of this work because it is simpler to compute for different vector sizes.

3.1.2 Impact of reducing n

We now show that decreasing n using the E method hurts the identification quality of the iris biometric. First we note that an FRR of $\leq .10$ requires a distance tolerance of $t \geq .3n$ (see the histograms in Figure 3). However, comparisons between different irises are tightly centered around $t = .5n$. This means for a dataset $\{x_i\}_{i=1}^{\ell}$ for most pairs x_i, x_j there exists some value x^* such that $\mathcal{D}(x_i, x^*) \leq t$ and $\mathcal{D}(x_j, x^*) \leq t$. This means for most pairs x_i, x_j, there is some query that will cause them both to be returned.

The goal of this subsection is to understand behavior on actual queries. We consider a distribution over x^* of different readings of individuals stored in the dataset to see how frequently multiple records are returned. Recall that multiple records being returned impacts the system correctness for both the MRProjC and MRProj constructions. It additionally affects leakage for MRProj. For these analysis we consider the ND-0405 dataset with the E mechanism for reducing the size of a feature vector (see the previous subsection).

We consider correctness of the system at different feature vector lengths n. We select a random reading of each biometric to represent the *encrypted dataset*. We first select a t that yields at most $\leq 10\%$ FRR (for comparisons of the same iris on the training dataset). We then use the following procedure:

1. Initialize matrix $C_{i,j} = 0^{356 \times 356}$.
2. Pick $\mathcal{I} \subset \{1, ..., 356\}$ of size 150 randomly.
3. For each i in $\mathcal{I}$:
   (a) Select 3 random readings of iris i, denoted x_i^* (removing reading that is encrypted):[7]
   (b) For all j if $\mathcal{D}(x_i^*, x_j) \leq t$ and $\mathcal{D}(x_i^*, x_i) \leq t$ increment $C_{i,j}$.
4. Compute $\texttt{ACount} = \sum_{i=0}^{355} \left(\sum_{j=0, j \geq i}^{355} C_{i,j} \right) / (3 * 150)$.

[7]Every iris in the ND0405 dataset has at least 4 readings so this is the maximum number of queries that will have an equal number of readings from the size 150 subset.

	Vector Length							
ACount	64	96	128	256	384	512	768	1024
Avg.	40.8	34.5	13.0	6.03	3.86	1.03	.53	.06
σ^2	.75	.74	.42	.23	.17	.083	.076	.019

Table 3: Effect of dimensionality reduction on the correctness and security of the resulting biometric search system. ACount is the average number of improperly records when searching for a biometric that is in the dataset. All feature extractors with $n < 1024$ use the E method to select features.

The value ACount represents how frequently a record of a different biometric would be returned by an in use search system. For both correctness and security considers one desires ACount to be as close to 0 as possible. We ran this experiment 40 times and report the mean and standard deviation of ACount in Table 3. As one can see keeping a vector size of $n = 1024$ has a three order of magnitude reduction in the average number of improperly returned records, underscoring the importance of inner product encryption to work with large n.

Leakage on readings of the same iris There are two types of biometric databases, those which associate a single reading x_i of a biometric with each record r_i and those where multiple readings of a biometric $x_{i,1}, ..., x_{i,k}$ are associated with a single record. Until now, we've implicitly assumed that the database has only one reading of a biometric. We now briefly consider the implications of leakage between readings of the same biometric. That is, $x_{i,1}, ..., x_{i,k}$ are readings from the same biometric and associated with a record r_i in the biometric database. First note that $x_{i,\alpha}$ and $x_{i,\beta}$ are likely to be close together (because readings of the same biometric are similar).

One may able to infer information about $x_{i,1}, ..., x_{i,k}$ from access pattern and distance equality leakage. One may be able to learn the relative positioning of the different readings by which values $\mathcal{I}$ are return by a query y (if it is not all values). Similarly, we expect the adversary to learn distance equality leakage for the entire set $x_{i,1}, ..., x_{i,k}$. Both of these leakage profiles allow an adversary to construct geometry of a biometric's different readings. This may allow the adversary to determine the type of noise present in that individual's biometric. It may be possible to use noise rates to draw conclusions about sensitive attributes about the corresponding person. Biometric systems frequently demonstrate systemic bias [DRD+20]. As one example most datasets draw from volunteer undergraduates students. Systems accuracy varies based on sensitive attributes such as gender, race, and age (see [DRD+20, Table 1]). Thus one may be able to infer sensitive attributes based on the relative size of $|\mathcal{I}|/k$.

If one stores multiple readings, it seems important to use cryptographic techniques to hide such leakage. A potential solution is to instead store a single reading that is the average of the multiple readings [ZD08] and make other values associated data that are not searchable.

4 Multi Random Projection IPE

As described in the Introduction, we show a general technique improving Setup efficiency for IPE schemes where ciphertexts and tokens are projected into dual vector spaces by a pair of matrices $\mathbf{A}, \mathbf{A}^{-1}$. We call this *multi random projection* technique. The key idea is to create multiple pairs of matrices of smaller dimension for subvectors of the inputs. These independent encodings are then combined with an additive secret sharing of 0 in the encryption so that computation with ciphertexts and tokens is only useful when using all of the components. In this section we show security of the technique when applied to the RProj scheme of Barbosa et al. [BCSW19, Section 4].[8]

Construction The construction is in Figure 4. We first argue correctness and then security. For security we show the scheme satisfies a stronger simulation based definition of security, as in the work of Barbosa et al. [BCSW19].

[8]Functional encryption for orthogonality (OFE) as defined by Barbosa et al. is equal to predicate inner product encryption, as defined in this work.

$\underline{\mathsf{Setup}(1^\lambda, n, \sigma)}$:

1. Sample $(G_1, G_2, G_T, q, e) \leftarrow \mathcal{G}_{abg}$ and randomly sample generators $g_1 \in G_1$ and $g_2 \in G_2$.
2. For $1 \leq \ell \leq \sigma$, randomly samples an invertible square matrix $\mathbb{B}_\ell \in \mathbb{Z}_q^{N \times N}$ and sets $\mathbb{B}_\ell^* = (\mathbb{B}_\ell^{-1})^T$, with $N = n/\sigma + 1$.
3. Outputs $\mathsf{pp} = (G_1, G_2, G_T, q, e, n, \sigma)$ as public parameters and $\mathsf{sk} = (g_1, g_2, \{\mathbb{B}_\ell, \mathbb{B}_\ell^*\}_{\ell=1}^{\sigma})$.

$\underline{\mathsf{TokGen}(\mathsf{pp}, \mathsf{sk}, y)}$:

1. Sample $\alpha \xleftarrow{\$} \mathbb{Z}_q$.
2. Splits y into σ subvectors y_ℓ of size n/σ.
3. For $1 \leq \ell \leq \sigma$, defines $y'_\ell = 1 \mid\mid y_\ell$ and sets $\mathsf{tk}_\ell = [\ \alpha \cdot (y'_\ell)^T \cdot \mathbb{B}_\ell\ ]_1$, a vector in G_1.
4. Outputs $\mathsf{tk} = (\mathsf{tk}_1, \cdots, \mathsf{tk}_\sigma)$.

$\underline{\mathsf{Encrypt}(\mathsf{pp}, \mathsf{sk}, x)}$:

1. Samples $\beta \xleftarrow{\$} \mathbb{Z}_q$.
2. Splits x into σ subvectors x_ℓ of size n/σ.
3. For $1 \leq \ell \leq \sigma - 1$, samples $\zeta_\ell \xleftarrow{\$} \mathbb{Z}_q$ then sets $\zeta_\sigma = -\sum_{\ell=1}^{\sigma-1} \zeta_\ell$.
4. For $1 \leq \ell \leq \sigma$ defines $x'_\ell = \zeta_\ell \mid\mid x_\ell$ and sets $\mathsf{ct}_\ell = [\ \beta \cdot (x'_\ell)^T \cdot \mathbb{B}_\ell^*\ ]_2$, a vector in G_2.
5. Outputs $\mathsf{ct} = (\mathsf{ct}_1, \cdots, \mathsf{ct}_\sigma)$.

$\underline{\mathsf{Decrypt}(\mathsf{pp}, \mathsf{tk}, \mathsf{ct})}$:

Computes $\left(\Pi_{\ell=1}^{\sigma} \Pi_{i=1}^{N} e(\mathsf{tk}_\ell[i], \mathsf{ct}_\ell[i])\right)$ and returns $\top$ if the results is equal to $1 \in \mathbb{G}_T$, $\perp$ otherwise.

Figure 4: Construction of MRProj.

$\underline{\mathsf{Real}_{\mathsf{IPE},\mathcal{A}}(1^\lambda)}$	$\underline{\mathsf{Ideal}_{\mathsf{IPE},\mathcal{A}}(1^\lambda)}$
$(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{IPE.Setup}(1^\lambda)$	$(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{IPE.Setup}(1^\lambda)$
$b \leftarrow \mathcal{A}^{\mathsf{IPE.TokGen}(\mathsf{sk},\cdot),\mathsf{IPE.Encrypt}(\mathsf{sk},\cdot)}(1^\lambda)$	$b \leftarrow \mathcal{A}^{\mathcal{S}(\Phi(\cdot))}(1^\lambda)$
Output b	Output b

Figure 5: Definition of experiment $\mathsf{Exp}_{SIM}^{\mathsf{IPE}}$. Φ denotes the information leakage received by the simulator $\mathcal{S}$ such that $\Phi(i,j) = f_{y_j}(x_i)$ for all i, j.

Correctness First note that $\langle x, y \rangle = \sum_{\ell=1}^{\sigma} \langle x_\ell, y_\ell \rangle$, and thus

$$\begin{aligned}\Pi_{\ell=1}^{\sigma} \Pi_{i=1}^{N} e(\mathsf{tk}_\ell[i], \mathsf{ct}_\ell[i]) &= g_T^{\sum_{\ell=1}^{\sigma} \beta \cdot (x'_\ell)^T \cdot \mathbb{B}_\ell^* \cdot \mathbb{B}_\ell^T \cdot \alpha \cdot (y'_\ell)} \\ &= g_T^{\sum_{\ell=1}^{\sigma} \beta \cdot (x'_\ell)^T \cdot \alpha \cdot (y'_\ell)} = g_T^{\alpha\beta \sum_{\ell=1}^{\sigma} \zeta_\ell + \langle x_\ell, y_\ell \rangle} \\ &= g_T^{\alpha\beta \cdot \langle x, y \rangle + \alpha\beta \cdot \sum_{\ell=1}^{\sigma} \zeta_\ell} = g_T^{\alpha\beta \cdot \langle x, y \rangle}\end{aligned}$$

If $\langle x, y \rangle = 0$ then $\Pi_{\ell=1}^{\sigma} \Pi_{i=1}^{N} e(\mathsf{tk}_\ell[i], \mathsf{ct}_\ell[i]) = e(g_1, g_2)^0 = 1$, which is the identity element in $\mathbb{G}_T$ and is easily detectible and $\top \leftarrow \mathsf{Decrypt}(\mathsf{pp}, \mathsf{tk}, \mathsf{ct})$ with probability 1. However, if $\langle x, y \rangle \neq 0$, then the probability that $\top \leftarrow \mathsf{Decrypt}(\mathsf{pp}, \mathsf{tk}, \mathsf{ct})$ is $\Pr[\alpha\beta \cdot \langle x, y \rangle = 0] \leq \frac{2}{q}$.

Definition 7 (Simulation-based security)**.** *Let $\mathsf{IPE} = (\mathsf{IPE.Setup}, \mathsf{IPE.TokGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ be a predicate IPE scheme over $\mathbb{Z}_q^n$. Then IPE is SIM-secure if for all PPT adversaries $\mathcal{A}$, there exist a simulator $\mathcal{S}$ such that for the experiment $\mathsf{Exp}_{SIM}^{\mathsf{IPE}}$ described in figure 5, the advantage of $\mathcal{A}$ ($\mathsf{adv}_{\mathcal{A}}^{\mathsf{Exp}_{SIM}^{\mathsf{IPE}}}$) is*

$$\left|\ \Pr[1 \leftarrow \mathsf{Real}_{\mathsf{IPE},\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathsf{Ideal}_{\mathsf{IPE},\mathcal{A}}(1^\lambda)]\ \right|$$

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

Kim et. al. [KLM+16, Remark 2.5] show that the simulation based security above implies the indistinguishability based definition, so we argue that the scheme in Figure 4 satisfies Definition 7 which implies Definition 2.

Theorem 1. *In the Generic Group Model for asymmetric bilinear groups the construction in Figure 4 is a secure $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk},\mathsf{pred}}$ scheme according to Definition 7 for the family of predicates $\mathcal{F} = \{f_y | y \in \mathbb{Z}_q^n\}$ such that for all vectors $x \in \mathbb{Z}_q^n$, $f_y(x) = (\langle x, y \rangle \stackrel{?}{=} 0)$.*

Proof of Theorem 1. This scheme has the security as the original $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ scheme from [BCSW19] for the simulation based security definition. We note that that scheme of Barbosa et al. [BCSW19] builds on the work Kim et al. [KLM+16] and our proof uses similar definitions of formal variables. The scheme works by having a challenger interact with a simulator S and two oracles, $\mathcal{O}'_{\mathsf{TokGen}}$ and $\mathcal{O}'_{\mathsf{Encrypt}}$, in the ideal scheme and a pair of oracles, $\mathcal{O}_{\mathsf{TokGen}}$ and $\mathcal{O}_{\mathsf{Encrypt}}$, in the real scheme. For this proof, we will build the simulator $\mathcal{S}$ which can correctly simulate the distribution of tokens and ciphertexts only using the predicate evaluation on whether the inner product of the two vectors is 0. This information is supplied to the simulator by the oracles $\mathcal{O}'_{\mathsf{TokGen}}$ and $\mathcal{O}'_{\mathsf{Encrypt}}$ to match the functionality of the encryption scheme.

Inner-product collection: Let i, j be shared counters between the token generation and encryption oracles. Let $x^{(i)} \in \mathbb{Z}_q^n$ and $y^{(j)} \in \mathbb{Z}_q^n$ denote respectively the adversary's i^{th} query to the token generation oracle and j^{th} query to the encryption oracle. The collection of mappings $\mathcal{C}_{\mathsf{ip}}$ is defined as

$$\mathcal{C}_{\mathsf{ip}} = \begin{cases} (i,j) \to 0 & \text{if } \langle x^{(i)}, y^{(j)} \rangle = 0 \\ (i,j) \to 1 & \text{otherwise.} \end{cases}$$

Formal variables: The simulator constructs formal variables for the unknowns of the system in order to respond as correctly as possible. Let Q be the maximum number of queries made by an adversary. Let σ and N be as in the construction in Figure 4. For all $i \in [Q]$, $\ell \in [\sigma]$ and $k \in [N]$, let $\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}, \hat{x}^{(i)}_{\ell,k}, \hat{y}^{(i)}_{\ell,k}$ represent the hidden variables $\alpha^{(i)}, \beta^{(i)}, x^{(i)}_{\ell,k}, y^{(i)}_{\ell,k}$, let $\hat{b}_{\ell,k,m}$ and $\hat{b}^*_{\ell,k,m}$ represent the entry in position (k, m) of the $\mathbb{B}_\ell$ and $\mathbb{B}^*_\ell$ matrices respectively, let $\hat{\zeta}^{(i)}_\ell$ be the formal variables for $\zeta^{(i)}_\ell$ where the simulator tracks the constraints that for each $i \in [Q]$, $\sum_{\ell=1}^{\sigma} \hat{\zeta}^{(i)}_\ell = 0$ and let $\hat{s}^{(i)}_{\ell,m}$ and $\hat{t}^{(i)}_{\ell,m}$ represent formal polynomials as constructed below,

$$\hat{s}^{(i)}_{\ell,m} = \sum_{k=1}^{N} \hat{y}'^{(i)}_{\ell,k} \cdot \hat{b}_{\ell,k,m} = \hat{b}_{\ell,1,m} + \sum_{k=2}^{N} \hat{y}^{(i)}_{\ell,k\text{-}1} \cdot \hat{b}_{\ell,k,m} \tag{1}$$

$$\hat{t}^{(i)}_{\ell,m} = \sum_{k=1}^{N} \hat{x}'^{(i)}_{\ell,k} \cdot \hat{b}^*_{\ell,k,m} = \hat{\zeta}^{(i)}_\ell \cdot \hat{b}^*_{\ell,1,m} + \sum_{k=2}^{N} \hat{x}^{(i)}_{\ell,k\text{-}1} \cdot \hat{b}^*_{\ell,k,m} \tag{2}$$

Then the universe of formal variables is $\mathcal{U} = \mathcal{R} \cup \mathcal{T}$, where

$$\mathcal{R} = \left\{ \hat{\alpha}^{(i)}, \hat{\beta}^{(i)} \right\}_{i \in [Q]} \cup \left\{ \hat{s}^{(i)}_{\ell,m} \ , \ \hat{t}^{(i)}_{\ell,m} \right\}_{i \in [Q],\ \ell \in [\sigma],\ m \in [N]}$$

and

$$\mathcal{T} = \left\{ \hat{\alpha}^{(i)}, \hat{\beta}^{(i)} \right\}_{i \in [Q]} \cup \left\{ \hat{x}'^{(i)}_{\ell,k}, \hat{y}'^{(i)}_{\ell,k}, \hat{\zeta}^{(i)}_\ell \right\}_{i \in [Q], \ell \in [\sigma], k \in [N]} \cup \left\{ \hat{b}_{\ell,k,m} \ , \ \hat{b}^*_{\ell,k,m} \right\}_{\ell \in [\sigma], m, k \in [N]}$$

Specification of the simulator Let $\mathcal{A}$ be a PPT adversary that makes at most $Q = \mathtt{poly}(\lambda)$ queries to the oracles. The simulator $\mathcal{S}$ starts by initializing an empty set of inner products $\mathcal{C}_{\mathsf{ip}}$ and three empty tables T_1, T_2, T_T which map handles to the polynomials over the variables of $\mathcal{R}$. The state of the simulator consists of these four objects, $(\mathcal{C}_{\mathsf{ip}}, T_1, T_2, T_T)$, which are updated after each query received. The simulator $\mathcal{S}$ answers the adversary's queries as follows.

Token generation queries: On input $x^{(i)} \in Z_q^n$, $\mathcal{O}'_{\mathsf{TokGen}}$ sends the collection $\mathcal{C}'_{\mathsf{ip}}$ to the simulator. $\mathcal{S}$ updates $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}'_{\mathsf{ip}}$. For $1 \le \ell \le \sigma$, $1 \le m \le N$, $\mathcal{S}$ generates a new handle $h_{\ell,m} \xleftarrow{\$} \{0,1\}^\lambda$ and adds the mapping $h_{\ell,m} \to \hat{\alpha}^{(i)} \cdot \hat{s}^{(i)}_{\ell,m}$ to T_1. $\mathcal{S}$ then sets $\mathsf{tk}_\ell = h_{\ell,1}, \cdots, h_{\ell,N}$. Finally, $\mathcal{S}$ returns the token $\mathsf{tk} = (\mathsf{tk}_1, \cdots, \mathsf{tk}_\sigma)$.

Encryption queries: On input $y^{(i)} \in Z_q^n$, $\mathcal{O}'_{\mathsf{Encrypt}}$ sends the collection $\mathcal{C}'_{\mathsf{ip}}$ to the simulator. $\mathcal{S}$ updates $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}'_{\mathsf{ip}}$. For $1 \le \ell \le \sigma$, $1 \le m \le N$, $\mathcal{S}$ generates a new handle $h_{\ell,m} \xleftarrow{\$} \{0,1\}^\lambda$ and adds the mapping $h_{\ell,m} \to \hat{\beta}^{(i)} \cdot \hat{t}^{(i)}_{\ell,m}$ to T_2. $\mathcal{S}$ sets $\mathsf{ct}_\ell = h_{\ell,1}, \cdots, h_{\ell,N}$. Finally, $\mathcal{S}$ returns the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \cdots, \mathsf{ct}_\sigma)$.

Addition oracle queries: Given $h_1, h_2 \in \{0,1\}^\lambda$, $\mathcal{S}$ verifies that formal polynomials p_1, p_2 exist in table T_τ, $\tau \in \{1, 2, T\}$ such that $h_1 \to p_1$ and $h_2 \to p_2$. If it is not the case $\mathcal{S}$ returns $\perp$. If a handle for $(p_1 + p_2)$ already exists in T_τ $\mathcal{S}$ returns it. Otherwise, $\mathcal{S}$ generates a new handle $h \xleftarrow{\$} \{0,1\}^\lambda$, adds the mapping $h \to (p_1 + p_2)$ to T_τ and returns h.

Pairing oracle queries: Given $h_1, h_2 \in \{0,1\}^\lambda$, $\mathcal{S}$ verifies that formal polynomials p_1, p_2 exist in tables T_1 and T_2 respectively, such that $h_1 \to p_1$ in T_1 and $h_2 \to p_2$ in T_2. If it is not the case $\mathcal{S}$ returns $\perp$. If a handle for $(p_1 \cdot p_2)$ already exists in T_T $\mathcal{S}$ returns it. Otherwise, $\mathcal{S}$ generates a new handle $h \xleftarrow{\$} \{0,1\}^\lambda$, adds the mapping $h \to (p_1 \cdot p_2)$ to T_T and returns h.

Zero-testing oracle queries: Given $h \in \{0,1\}^\lambda$, $\mathcal{S}$ verifies that formal polynomials p exists in T_τ, $\tau \in \{1, 2, T\}$, such that $h \to p$. If it is not the case $\mathcal{S}$ returns $\perp$. $\mathcal{S}$ then works as follows.

1. It "canonicalizes" the polynomial p by expressing it as a sum of products of formal variables in $\mathcal{T}$ with $\mathtt{poly}(\lambda)$ terms.
2. If $\tau \in \{1, 2\}$ and p is the zero polynomial, $\mathcal{S}$ outputs "zero". Otherwise if outputs "non-zero".
3. If $\tau = T$ the simulator decomposes p into the form

$$p = \sum_{i,j=1}^{Q} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot \left(p_{i,j}\left(\{\hat{s}^{(i)}_{\ell,m},\ \hat{t}^{(j)}_{\ell,m}\}_{\ell \in [\sigma], m \in [N]} \right) + f_{i,j}\left(\{\hat{s}^{(i)}_{\ell,m},\ \hat{t}^{(j)}_{\ell,m}\}_{\ell \in [\sigma], m \in [N]} \right) \right) \tag{3}$$

   where for $1 \le i, j \le Q$, $p_{i,j}$ is defined as

$$p_{i,j} = c_{i,j} \cdot \left(\sum_{\ell,m=1}^{\sigma,N} \hat{s}^{(i)}_{\ell,m}\ \hat{t}^{(j)}_{\ell,m} \right)$$

   where $c_{i,j} \in \mathbb{Z}_q$ is the coefficient of the term $\hat{s}^{(i)}_{1,1} \hat{t}^{(j)}_{1,1}$, and $f_{i,j}$ consists of the remaining terms.
4. If for all $1 \le i, j \le Q$, $(i, j) = 0$ in $\mathcal{C}_{\mathsf{ip}}$ (corresponding to a zero inner product) and $f_{i,j}$ does not contain any non-zero term, $\mathcal{S}$ outputs "zero". Otherwise it outputs "non-zero".

Correctness of the simulator As in the original proof, the simulator's responses to token generation, encryption and group oracle queries are distributed identically as in the real experiment. We now have to show correctness of the simulator's answers to zero-testing oracle queries.

1. We first need to show that the canonicalization process in step 1 is efficient. Since the adversary can only obtain handles to new monomials using token generation and encryption queries, the monomials are all over formal variables in $\mathcal{R}$. Also, since the adversary can make Q queries at most, the polynomial p they can build and submit to the zero-testing oracle has at most $\mathtt{poly}(Q)$ terms and degree 2.
   Then using Equations 1 and 2, the formal polynomial p can be expressed as a polynomial over formal variables in $\mathcal{T}$. Since p has degree at most 2 over variables in $\mathcal{R}$, it can be expressed as a sum of at most $\mathtt{poly}(Q, n)$ monomials over variables in $\mathcal{T}$ and has degree at most $\mathtt{poly}(n)$. Since both the polynomial over $\mathcal{R}$ and the canonical polynomial over $\mathcal{T}$ are polynomially-sized, this is efficient.
2. For $\tau = 1$, the only monomials the adversary can obtain are responses to token generation queries. Then the

canonical polynomial is of the form

$$\begin{aligned} p &= \sum_{i=1}^{Q} \hat{\alpha}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c_{\ell,m}^{(i)} \cdot \hat{s}_{\ell,m}^{(i)} \right) \\ &= \sum_{i=1}^{Q} \hat{\alpha}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c_{\ell,m}^{(i)} \sum_{k=1}^{N} \hat{y}_{\ell,k}^{\prime(i)} \cdot \hat{b}_{\ell,k,m} \right) \\ &= \sum_{i=1}^{Q} \hat{\alpha}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c_{\ell,m}^{(i)} \left(\hat{b}_{\ell,1,m} + \sum_{k=2}^{N} \hat{y}_{\ell,k}^{(i)} \cdot \hat{b}_{\ell,k,m} \right) \right) \end{aligned}$$

where $c_{1,1}^{(i)}, \cdots, c_{\sigma,N}^{(i)} \in \mathbb{Z}_q$.

Notice that the sum $\hat{b}_{\ell,1,m} + \sum_{k=2}^{N} \hat{y}_{\ell,k}^{(i)} \cdot \hat{b}_{\ell,k,m}$ can never be the identically zero polynomial over the formal variables $\{\hat{b}_{\ell,k,m}\}_{\ell\in[\sigma],\ k,m\in[N]}$. This holds irrespective of the actual values of the adversary's query $x^{(i)}$. Since all $\{\hat{\alpha}^{(i)}\}_{i\in[Q]}$ and $\{\hat{b}_{\ell,k,m}\}_{\ell\in[\sigma],\ k,m\in[N]}$ are sampled uniformly and independently in the real game and the polynomial p has degree $\mathtt{poly}(n) = \mathtt{poly}(\lambda)$, then by the Schwartz-Zippel lemma [KLM+16, Lemma 2.9], p evaluates to non-zero with overwhelming probability. This implies that the simulator is correct with overwhelming probability.

3. For $\tau = 2$, the only monomials the adversary can obtain are responses to ciphertexts queries. Then the canonical polynomial is of the form

$$\begin{aligned} p &= \sum_{i=1}^{Q} \hat{\beta}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c_{\ell,m}^{(i)} \cdot \hat{t}_{\ell,m}^{(i)} \right) \\ &= \sum_{i=1}^{Q} \hat{\beta}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c_{\ell,m}^{(i)} \sum_{k=1}^{N} \hat{x}_{\ell,k}^{\prime(i)} \cdot \hat{b}_{\ell,k,m}^{*} \right) \\ &= \sum_{i=1}^{Q} \hat{\beta}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c_{\ell,m}^{(i)} \left(\zeta_{\ell}^{(i)} \cdot \hat{b}_{\ell,1,m}^{*} + \sum_{k=2}^{N} \hat{x}_{\ell,k}^{(i)} \cdot \hat{b}_{\ell,k,m}^{*} \right) \right) \end{aligned}$$

where $c_{1,1}^{(i)}, \cdots, c_{\sigma,N}^{(i)} \in \mathbb{Z}_q$. Notice that the sum $\zeta_{\ell}^{(i)} \cdot \hat{b}_{\ell,1,m}^{*} + \sum_{k=2}^{N} \hat{x}_{\ell,k}^{(i)} \cdot \hat{b}_{\ell,k,j}^{*}$ can only be the identically zero polynomial over the formal variables $\{\hat{b}_{\ell,k,m}^{*}\}_{\ell\in[\sigma],\ k,m\in[N]}$ if $\zeta_{\ell}^{(i)} = 0$ which happens with negligible probability. Again, this holds irrespective of the adversary's queries $y^{(1)}, \cdots, y^{(Q)}$ and p is not the identically zero polynomial over the formal variables $\{\hat{\beta}^{(i)}\}_{i\in[Q]}$ and $\{\hat{b}_{\ell,k,m}^{*}\}_{\ell\in[\sigma],\ k,m\in[N]}$. Since all $\hat{b}_{\ell,k,m}^{*}$ are independent from one another (since $\hat{b}_{\ell,k,m}$ was sampled uniformly and independently), then again by Schwartz-Zippel lemma p evaluates to non-zero with overwhelming probability, the simulator is correct with overwhelming probability.

4. For $\tau = T$, the only polynomials the adversary can obtain are products of two polynomials, one from each base group. Then the polynomial p can be decomposed into a sum of monomials that each contain $\alpha^{(i)}$ and $\beta^{(j)}$ for some $i, j \in [Q]$. Then $\mathcal{S}$ can regroup terms for each $i, j \in [Q]$ and obtain Equation 3. If $f_{i,j}$ does not contain

any term, then p is of the form

$$\begin{aligned}
p &= \sum_{i,j=1}^{Q} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot c_{i,j} \cdot \left(\sum_{\ell,m=1}^{\sigma,N} \hat{s}_{\ell,m}^{(i)} \; \hat{t}_{\ell,m}^{(j)} \right) \\
&= \sum_{i,j=1}^{Q} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot c_{i,j} \cdot \left(\sum_{\ell,m=1}^{\sigma,N} \left(\sum_{k=1}^{N} \hat{y}_{\ell,k}^{\prime(i)} \cdot \hat{b}_{\ell,k,m} \right) \cdot \left(\sum_{k=1}^{N} \hat{x}_{\ell,k}^{\prime(j)} \cdot \hat{b}_{\ell,k,m}^{*} \right) \right) \\
&= \sum_{i,j=1}^{Q} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot c_{i,j} \cdot \left(\sum_{\ell=1}^{\sigma} (\hat{x}_{\ell}^{\prime(j)})^{T} \cdot \mathbb{B}_{\ell}^{*} \cdot \mathbb{B}_{\ell}^{T} \cdot \hat{y}_{\ell}^{\prime(i)} \right) \\
&= \sum_{i,j=1}^{Q} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot c_{i,j} \cdot \left(\sum_{\ell=1}^{\sigma} \zeta_{\ell}^{(i)} + \langle \hat{x}_{\ell}^{(j)}, \hat{y}_{\ell}^{(i)} \rangle \right) \\
&= \sum_{i,j=1}^{Q} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot c_{i,j} \cdot \langle \hat{x}^{(j)}, \hat{y}^{(i)} \rangle
\end{aligned}$$

Then, it is obvious that p is the zero polynomial when all (i, j) inner products are zero, which can be known by checking if $(i, j) \to 0$ in $\mathcal{C}_{\mathsf{ip}}$.
Now suppose that for some $i, j \in [Q]$ the polynomial $f_{i,j}$ contains at least one term. Then we claim that $f_{i,j}$ cannot be the identically zero polynomial over the formal variables $\{ \hat{b}_{\ell,k,m} \}_{\ell \in [\sigma], k,m \in [N]}$, irrespective of the adversary's choice of admissible queries. We refer the reader to the original work [KLM+16, Section 3] for a detailed proof of this claim. Then by the Schwartz-Zippel lemma, p evaluates to non-zero with overwhelming probability when $f_{i,j}$ contains at least one term.

□

5 Building distance hiding PSE

As mentioned in Section 2, Hamming distance can be calculated using the inner product between the two biometric vectors. As such, we can use a range of possible inner product values as the distance threshold.

Predicate function-hiding secret key inner product encryption [SSW09], or $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$, allows one to test if the inner product between two vectors is equal to zero. By appending a value to the first vector and -1 to the second vector, we can support equality testing for non-zero values. Generating several tokens or ciphertexts, one per distance in the range, allows to test if the inner product is below the chosen threshold.
We show that one can use $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ to construct PSE for Hamming distance. At a high level, each keyword is encoded as a {-1, 1} vector and -1 is appended to it, which in turn is encrypted with $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$. Keywords are similarly encoded but this time a distance from the range is appended to them, and tokens generated as part of the underlying $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ scheme.

Construction 1 (Proximity Searchable Encryption)**.** *Fix the security parameter $\lambda \in \mathbb{N}$. Let $\mathsf{IPE}_{\mathsf{fh,sk,pred}} = (\mathsf{IPE.Setup}, \mathsf{IPE.TokGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ be a predicate function-hiding secret key IPE scheme over $\mathbb{Z}_q^{n+1}$. Let $x_i \in \mathbb{Z}_q^n$ and $X = (x_1, \cdots, x_\ell)$ be the list of keywords. Let $\mathcal{F}$ be the set of all predicates such that for any $x_i \in X$, $f_{y,t}(x_i) = 1$ if the Hamming distance between x_i and the query vector $y \in \mathbb{Z}_q^n$ is less or equal to some chosen threshold $t \in \mathbb{Z}_q$, $f_{y,t}(x_i) = 0$ otherwise. Figure 6 is a proximity searchable encryption scheme for the Hamming distance.*

Theorem 2 (PSE main theorem)**.** *Let $\mathsf{IPE}_{\mathsf{fh,sk,pred}} = (\mathsf{IPE.Setup}, \mathsf{IPE.TokGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ be an IND-secure function-hiding inner product predicate encryption scheme over $\mathbb{Z}_q^{n+1}$. Then there exists $\mathsf{PSE} = (\mathsf{PSE.Setup}, \mathsf{PSE.BuildIndex}, \mathsf{PSE.Trapdoor}, \mathsf{PSE.Search})$, a secure proximity searchable encryption scheme for the Hamming distance, such that for any PPT adversary $\mathcal{A}_{\mathsf{PSE}}$ for $\mathsf{Exp}_{IND}^{\mathsf{PSE}}$, there exists a PPT adversary $\mathcal{A}_{\mathsf{IPE}}$ for $\mathsf{Exp}_{IND}^{\mathsf{IPE}}$, such that for any security parameter $\lambda \in \mathbb{N}$,*

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{PSE}}}^{\mathsf{Exp}_{IND}^{\mathsf{PSE}}} = \mathsf{Adv}_{\mathcal{A}_{\mathsf{IPE}}}^{\mathsf{Exp}_{IND}^{\mathsf{IPE}}}$$

Proof of Theorem 2. The correctness of the scheme follows from the correctness of the underlying IPE scheme. Assume there exists $x_i \in X$, $i \in [1, \ell]$, such that $f_{y,t}(x_i) = 1$. That is $\mathcal{D}(y, x_i) \leq t$ with $\mathcal{D}(y, x_i)$ the Hamming distance between vectors y and x_i. Then there exists a unique $\mathsf{tk}_j \in Q_{y,t}$ such that $b_j \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_j, \mathsf{ct}_i)$ and $b = 1$ with overwhelming probability by the correctness of the IPE scheme. Now assume that for some $x_i \in X$, $i \in [1, \ell]$, we have $f_{y,t}(x_i) = 0$. Then for all $\mathsf{tk}_j \in Q_{y,t}$, $b_j \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_j, \mathsf{ct}_i)$ and $b_j = 1$ with negligible probability. Then considering the worst case where either $\mathcal{D}(y, x_\ell) = t$ or for all $x_i \in X$, $f_{y,t}(x_i) = 0$, we have:

$$\begin{aligned}
&\Pr\big[\mathsf{PSE.Search}(\mathsf{pp}, Q_{y,t}, I_X) = J_{X,y,t}\big] \\
&\geq 1 - \ell(t+1) \times \Pr\left[\begin{smallmatrix}\mathsf{IPE.Decrypt}(\mathsf{pp},\mathsf{tk}_j,\mathsf{ct}_i) \\ \neq (\mathcal{D}(x_i,y) \stackrel{?}{=} d_j)\end{smallmatrix}\right] \\
&\geq 1 - \ell(t+1) \times \mathsf{negl}(\lambda).
\end{aligned}$$

We now prove the security of the construction. Let $\mathcal{A}_{\mathsf{PSE}}$ be a PPT adversary for the experiment $\mathsf{Exp}^{\mathsf{PSE}}_{IND}$ and $\mathcal{C}_{\mathsf{IPE}}$ be an challenger for $\mathsf{Exp}^{\mathsf{IPE}}_{IND}$. Then we can build a PPT adversary $\mathcal{A}_{\mathsf{IPE}}$ for the experiment $\mathsf{Exp}^{\mathsf{IPE}}_{IND}$ which works as follows:

1. $\mathcal{A}_{\mathsf{IPE}}$ receives pp from $\mathcal{C}_{\mathsf{IPE}}$ and forwards it to $\mathcal{A}_{\mathsf{PSE}}$.
2. $\mathcal{A}_{\mathsf{IPE}}$ receives two m-query histories $\mathsf{History}^{(0)}, \mathsf{History}^{(1)}$ from $\mathcal{A}_{\mathsf{PSE}}$ where $\mathsf{History}^{(\beta)} = (X^{(\beta)}, F^{(\beta)})$ for $\beta \in \{0, 1\}$.
3. For each ${x_i}^{(\beta)} \in X^{(\beta)}$, $i \in [1, \ell]$, $\mathcal{A}_{\mathsf{IPE}}$ encodes it as ${x_i}^{(\beta)*} \in \{\text{-}1, 1\}^n$ and creates the encryption query $S_i = ({x_i}^{(0)*} || \text{ -1}, \ {x_i}^{(1)*} || \text{ -1})$.
4. $\mathcal{A}_{\mathsf{IPE}}$ sets $S = S_1, \cdots, S_\ell$.
5. For each $f_j^{(\beta)} \in F^{(\beta)}$, $j \in [1, m]$:
   (a) $\mathcal{A}_{\mathsf{IPE}}$ extracts a vector ${y_j}^{(\beta)} \in \mathbb{Z}_q^n$ and the distance threshold $t \in \mathbb{N}$.
   (b) $\mathcal{A}_{\mathsf{IPE}}$ encodes ${y_j}^{(\beta)}$ as ${y_j}^{(\beta)*} \in \{\text{-}1, 1\}^n$ and creates $D_j^{(0)} = (d_0, \cdots, d_t)$ such that $d_k = n - 2k$ with $0 \leq k \leq t$.
   (c) $\mathcal{A}_{\mathsf{IPE}}$ creates $D_j^{(0)*}$ by reordering the elements in $D_j^{(0)}$ such that for all $k \in [0, t]$ and $d_k^{(0)} \in D_j^{(0)*}, d_k^{(1)} \in D_j^{(1)}$ we have
   $$\big(\langle {x_i}^{(0)}, {y_j}^{(0)} \rangle \stackrel{?}{=} d_k^{(0)}\big) = \big(\langle {x_i}^{(1)}, {y_j}^{(1)} \rangle \stackrel{?}{=} d_k^{(1)}\big).$$
   ($\mathcal{A}_{\mathsf{IPE}}$ can always find a permutation to make this last condition by the admissibility requirement.)
   (d) $\mathcal{A}_{\mathsf{IPE}}$ samples a random permutation $\psi_j : [0, t] \to [0, t]$.
   (e) For $0 \leq k \leq t$, $\mathcal{A}_{\mathsf{IPE}}$ creates ${y_j}^{(\beta)*} || \ d_k^{(\beta)}$ with $\beta \in \{0, 1\}$, $d_k^{(0)} \in D_j^{(0)*}$ and $d_k^{(1)} \in D_j^{(1)}$. Then $\mathcal{A}_{\mathsf{IPE}}$ computes
   $$R_j^{(\beta)} = \psi_j\Big(\ {y_j}^{(\beta)*} || \ d_0^{(\beta)}, \cdots, {y_j}^{(\beta)*} || \ d_t^{(\beta)} \ \Big)$$
   and sets $R_j = (R_j^{(0)}, R_j^{(1)})$.
   (f) $\mathcal{A}_{\mathsf{IPE}}$ sets $R = R_1, \cdots, R_m$.
6. $\mathcal{A}_{\mathsf{IPE}}$ sends the token generation queries R and encryption queries S to $\mathcal{C}_{\mathsf{IPE}}$ and receives back a set of tokens $T^{(\beta)} = \mathsf{tk}_{1,0}^{(\beta)}, \cdots, \mathsf{tk}_{m,t}^{(\beta)}$ and a set of encrypted keywords $C^{(\beta)} = \mathsf{ct}_1^{(\beta)}, \cdots, \mathsf{ct}_\ell^{(\beta)}$ such that
   $$\begin{aligned}
   \mathsf{tk}_{j,k}^{(\beta)} &\leftarrow \mathsf{IPE.TokGen}(\mathsf{sk}, \ {y_j}^{(\beta)*} || \ d_k^{(\beta)}) \\
   \mathsf{ct}_i^{(\beta)} &\leftarrow \mathsf{IPE.Encrypt}(\mathsf{sk}, \ {x_i}^{(\beta)*} || \text{ -1})
   \end{aligned}$$
   for $i \in [1, \ell]$, $j \in [1, m]$, $k \in [0, t]$ and $\beta \in \{0, 1\}$. $\mathcal{A}_{\mathsf{IPE}}$ forwards $T^{(\beta)}$ and $C^{(\beta)}$ to $\mathcal{A}_{\mathsf{PSE}}$, respectively as the encrypted index $I^{(\beta)}$ and the list of queries $Q^{(\beta)}$.
7. $\mathcal{A}_{\mathsf{IPE}}$ receives $\beta' \in \{0, 1\}$ from $\mathcal{A}_{\mathsf{PSE}}$ and returns it.

$\underline{\mathsf{PSE.Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pp})}$:
Run and output $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{IPE.Setup}(1^\lambda)$.

$\underline{\mathsf{PSE.Trapdoor}(\mathsf{sk}, f_{y,t}) \to Q_{y,t}}$:

1. For $i = 0$ to t, compute $d_j = n - 2j$,
2. Set $D = (d_0, ..., d_t)$,
3. Sample random permutation $\pi : [0, t] \to [0, t]$,
4. Compute $D^* = \pi(D) = \{d_1^*, \cdots, d_t^*\}$,
5. Encode y as $y^* \in \{-1, 1\}^n$,
6. For $1 \leq j \leq t$ call $\mathsf{tk}_j \leftarrow \mathsf{IPE.TokGen}(\mathsf{sk}, y^* || \ d_j^*)$.,
7. Output $Q_{y,t} = (\mathsf{tk}_1, \cdots, \mathsf{tk}_t)$.

$\underline{\mathsf{PSE.BuildIndex}(\mathsf{sk}, X) \to I_X}$:

1. For each keyword $x_i \in X$, $i \in \{1, \cdots, \ell\}$, encode $x_i^* \in \{-1, 1\}^n$, compute $\mathsf{ct}_i \leftarrow \mathsf{IPE.Encrypt}(\mathsf{sk}, x_i^* || \text{ -1})$.
2. Outputs $I_X = (\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell)$.

$\underline{\mathsf{PSE.Search}(\mathsf{pp}, Q_{y,t}, I_X) \to J_{X,y,t}}$:

1. Initialize $J_{X,y,t} = \emptyset$.
2. For each $\mathsf{ct}_i \in I_X$ and for each $\mathsf{tk}_j \in Q_{y,t}$, call $b_j \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_j, \mathsf{ct}_i)$. If $b_j = 1$, add i to $J_{X,y,t}$, continue to ct_{i+1}.
3. Outputs $J_{X,y,t}$.

Figure 6: Construction of proximity search from $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$.

	Underlying IPE scheme				
	MRProjC	RProj	[BCSW19]	[KT14]	[SSW09]
group order	Prime	Prime	Prime	Prime	Composite
Setup	$\sigma((n+1)/\sigma)^3$	$(n+1)^3$	$(n+1)^3$	$(6n+6)^3$	$4n+8$
BuildIndex	$\ell(n+\sigma+1)$	$\ell(n+1)$	$\ell(12n+21)$	$6\ell(n+1)$	$\ell(32n+36)$
Trapdoor	$(t+1)(n+\sigma+1)$	$(t+1)(n+1)$	$(t+1)(12n+21)$	$6(t+1)(n+1)$	$(t+1)(24n+40)$
Search	$\ell(t+1)(n+\sigma+1)$	$\ell(t+1)(n+1)$	$\ell(t+1)(6n+12)$	$6\ell(t+1)(n+1)$	$\ell(t+1)(4n+8)$
$\lvert\mathsf{sk}\rvert$	$2(n+1)^2/\sigma + 4n + 2\sigma + 6$	$2(n+1)^2 + 2$	$24n + 42$	$60(n+1)^2$	$4n+8$
$\lvert\mathcal{I}\rvert$	$\ell(n+\sigma+1)$	$\ell(n+1)$	$\ell(6n+12)$	$6\ell(n+1)$	$\ell(2n+4)$
$\lvert\mathsf{tk}_{y,t}\rvert$	$(t+1)(n+\sigma+1)$	$(t+1)(n+1)$	$(t+1)(6n+12)$	$6(t+1)(n+1)$	$(t+1)(2n+4)$

Table 4: PSE scheme efficiency for keywords of size n depending on underlying $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ scheme. Upper part of the table shows number of group or pairing operations per function. Lower part of the table shows number of group elements per component. The scheme of Shen, Shi, and Waters [SSW09] uses a composite order group whose order is the product of four large primes. The number n is the length of the biometric template, σ is the number of bases in the multi random projection scheme, t is the desired distance tolerance, and ℓ is the total number of records in the database.

Since the number of token generation queries, $m \times t$, sent by $\mathcal{A}_{\mathsf{IPE}}$ remains polynomial in the security parameter, the advantage of $\mathcal{A}_{\mathsf{PSE}}$ is

$$\mathsf{Adv}^{\mathsf{Exp}^{\mathsf{PSE}}_{IND}}_{\mathcal{A}_{\mathsf{PSE}}} = \mathsf{Adv}^{\mathsf{Exp}^{\mathsf{IPE}}_{IND}}_{\mathcal{A}_{\mathsf{IPE}}}$$

This completes the proof of Theorem 2. □

Table 4 presents the resulting efficiency of distance hiding PSE schemes based on different $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ constructions. This table corresponds to $t + 1$ tokens with all operations on dimension $n + 1$.

6 Implementation

This section presents an implementation and an evaluation of the PSE scheme proposed in this paper.

6.1 Implementation

We implemented the MRProj construction described in section 4 and a PSE (see section 5) scheme using it in Python3. These implementations can be found in a Github repository [ACD+21]. Our IPE implementations uses

			Time							Sizes		
			MRProj				MRProjC		RProjC	MRProj		RProjC
n	σ	t	Setup	BuildIndex	Trapdoor	Search	Trapdoor	Search	Setup	|EncDB|	|sk|	|sk|
128	3	38	75	1.5	.36	234	.01	31	4×10^3	5.9×10^6	5.6×10^5	1.6×10^6
192	5	57	47	2.2	.8	495	.01	46	1.3×10^4	8.9×10^6	7.7×10^5	3.6×10^6
256	7	76	57	2.9	1.4	850	.02	62	3.2×10^4	1.2×10^7	9.8×10^5	6.4×10^6
384	10	115	94	4.4	3.1	1870	.03	92	1.1×10^5	1.8×10^7	1.5×10^6	1.4×10^7
512	13	153	153	5.7	5.7	3282	.04	140	2.6×10^5	2.4×10^7	2.1×10^6	2.6×10^7
768	19	230	269	8.6	13.4	7210	.06	185	8.6×10^5	3.6×10^7	3.2×10^6	5.7×10^7
1024	25	307	268	10.8	22.4	12600	.08	241	2.0×10^6	4.7×10^7	4.3×10^6	1.0×10^8

Table 5: Operations timing (in seconds) and sizes (in bytes) for different vector sizes. n represents the vector length, σ the number of bases used, and $t = .30$ the distance tolerance. Setup and BuildIndex procedures for MRProj and MRProjC schemes are the same procedures, MRProjC uses vectors whose length is 1 fewer. We only report these algorithms for MRProj. Timing and storage for the MRProjC Setup is interpolated. Measured $n = 10$ to 240 in steps of 10. For timing, cubic fit with coefficients $y = .003x^3 - .578x^2 + 36x - 557$ with $R^2 = .996$. For storage, quadratic fit with coefficients $y = 96x^2 + 192x + 573$ with $R^2 = 1$.

the Charm [AGM+13] and FLINT [Har10] libraries for the pairing group operations and finite field arithmetic in $\mathbb{Z}_q$. For comparison purposes, we used the pairing group over the asymmetric curve MNT159, the same as in Kim et al.'s FHIPE implementation [Lew16].

The search, encryption and token generation algorithms were parallelized. Benchmarking tests for each algorithm were implemented and the number of random projections, the distance threshold and the input vector sizes for these tests can vary. This allowed us to compare efficiency for different parameters and pinpoint values that yield a practical and accurate scheme. With a number of random projections equal to 1, we obtain Setup timings and secret key size for RProjC. Setting the distance threshold to 0 allows us to get timings for MRProj. To be as realistic as possible, we used iris readings from the ND 0405 as input vectors to the benchmarking tests.

6.2 Evaluation

We evaluate our implementations on a Linux server with an AMD Ryzen 9 3950X 16-Core processor and 64GB of RAM. Remember that the preferred input vector size for correctness is 1024 (as stated in Section 3).

Timing We evaluate the timing efficiency of our PSE construction with and without the multi random projection technique. Table 5 reports the timings for all four algorithms of the PSE scheme. MRProj corresponds to the PSE construction presented in this paper. RProjC corresponds to Kim et al.'s FHIPE construction, MRProjC corresponds to the same scheme but with the multi random projection technique applied . In the last column of the timing section of the table, we report the timing of the Setup algorithm without this multi random projection construction. During our tests, we noticed a jump in Setup timings when going from sub-vectors of 40 to 60 group elements, we thus chose σ values that yield sub-vectors lengths of approximately 40. We make three main observations.

1. Setup and BuildIndex have comparable performance for MRProj and MRProjC (the only difference is adding 1 to underlying dimension). However, Trapdoor is substantially slower for MRProj since it prepares $t + 1$ tokens, but performance remains reasonable.

2. Distance hiding has a large impact on the Search algorithm. MRProjC Search takes 4 minutes, MRProj Search takes 3.5 hours. Both approaches scans the whole database which is problematic for large datasets. We discuss possible solutions in Section 8.

3. Finally, this table shows that Setup without multi random projection is completely impractical for large input vector sizes. In particular, for vectors of size 1024, Setup takes more than eleven days. In comparison, Setup using multi random projection takes less than five minutes for input vectors of size 1024. Our multi random projection construction thus allows to use a large enough input vector size to maintain a high correctness while increasing the efficiency of the setup algorithm. This is explained by the fact that the Setup algorithm's running time is dominated by the matrix inversion. It is then more efficient to perform multiple inversions of small matrices than a single inversion of a bigger one.

Storage We evaluate the impact of the multi random projection PSE construction on storage efficiency. As can be seen on table 5, the impact is low for small input vectors, however, it makes a big difference for larger ones. Indeed, when the size of the Barbosa key (key generated without the multi random projection technique) grows quadratically with the vector size, the size of the key generated with the multi random projection technique grows with $(n/\sigma)^2 * \sigma \approx n^2/\sigma$. For vectors of size 1024, we consider $\sigma = 25$ and the secret key generated with the multi random projection technique is 23.2 times smaller than the single basis key, confirming the asymptotic analysis.

7 Further Related Work

In this section we review further related work on proximity search. We defer discussion of leakage abuse attacks to Appendix A. Li et al. [LWW+10], Wang et al. [WMT+13] and Boldyreva and Chenette [BC14] reduced proximity search to keyword equality search. These works propose two complimentary approaches:

1. When adding a record x_i to a database, also insert all close values as keywords, that is $\{x_j \mid \mathcal{D}(x_i, x_j) \leq t\}$ are added as keywords associated to x_i.
2. The second approach requires a searchable encryption scheme that supports disjunctive search. It inserts just x_i, but when searching for y it searches for the disjunction $\vee_{x_i|\mathcal{D}(x_i,y)\leq t}\ x_i$.

Either approach can be instantiated using a searchable encryption scheme that supports disjunction over keyword equality (inheriting any leakage). However, for biometrics, the number of keywords $\vee_{x_i|\mathcal{D}(x_i,y)\leq t}\{x_i\}$ usually grows exponentially in t. In existing disjunctive schemes, the size of the query grows with the size of the disjunction [FVY+17], making this approach only viable for constant values of t.

Kuzu et al.'s [KIK12] solution relies on *locality sensitive hashes* [IM98]. A locality sensitive hash ensures that close values have a higher probability to produce collisions than values that are far apart. Thus, a scheme can be built from any scheme supporting disjunctive keyword equality, inheriting any leakage. The server learns the number of matching locality sensitive hashes for each record (which is expected to be more than 0). The number of matching locality sensitive hashes is a proxy for the distance between the query value and the records. More matching locality sensitive hashes implies smaller distance. This allows the server to establish the approximate distance between each record and the query.

Zhou and Ren [ZR18] propose a variant of inner product encryption that reveals if the distance is less than t only. However, their security is based on $\mathbf{A}x_i$ and $y\mathbf{B}$ hiding x_i and y for secret square $\mathbf{A}$ and $\mathbf{B}$. Security is heuristic with no underlying assumption or proof of information theoretic security.

8 Conclusion

Iris biometric feature extractors produce feature vectors similar in the binary Hamming metric. Inner product encryption was proposed to build encrypted search for the binary Hamming metric. In this work we explored a domain specific solution for secure searchable encryption for iris biometric databases.

We observed in the statistics of the iris biometric data that large vectors are required for both correctness and minimizing leakage. With large vectors, we see that the distance between readings of the same class can be separated from the distance distribution from the readings of other classes (see Figure 3). This means that with a fixed distance threshold, we can ensure that more readings of the same class are approved while readings from other classes are denied (with high probability).

In prior work, Setup was not feasible for large vector lengths due to the cost of inverting large matrices. In the most relevant prior work [KLM+18], they skip this step in benchmarking due to the high cost. Our interpolation results show that for $n = 1024$ would take roughly 23 days. This is estimated on a parallel implementation in C. The length $n = 1024$ is the length of prior iris feature extractors. We do not consider this time acceptable.

In the RProjC scheme of Kim et al. [KLM+18], additionally the distance is leaked between queries and all points in the database. Based on prior work on trilaterilation, with a constant number of queries observed in n, the server can build complete distance information between the stored data points. If the adversary knows auxiliary information about the database, the encryption may not protect the data at all.

In this work we offer solutions to these two problems. We show a *multi random projection* approach that allows for breaking large vectors into small vectors. This allows us to use smaller matrices greatly reducing the computational

time required to invert the matrices. Doing two $n/2$ inversions takes $1/4$ the time of one size n inversion. Careful optimization improves Setup time by four orders of magnitude while only increasing search time by 3%.

We show how to use *predicate* inner product encryption to build a scheme that hides the distance between the query and the stored records. By using a predicate scheme instead of one that gives the value of the inner product, the server only learns if the two vectors are a fixed distance from one another. This greatly reduces the information that is leaked through remotely executing this operation. The server only learns information about data that are close the queried point and learns nothing about data that are outside the distance threshold. We show this scheme leaks only access pattern and distance equality leakage.

The improvement in accuracy for higher n also yields an improvement of leakage profile for our MRProj scheme. When two or more classes are returned from a single query, this leaks that the returned items are within distance $2t$ (through access pattern) and whether they are the same distance from the query (distance equality leakage). Decreasing the statistical overlap between classes minimizes the probability of both leakages which translates to a more private system for sensitive biometric data.

The transformation comes at a cost of making search slower and no longer appropriate for moderately sized databases. We believe that this transformation is required in order to maintain the integrity of sensitive biometric information. Thus, our main open problem is whether or not this significant slow down to search is avoidable. For databases at larger scales, doing a linear search of the entire database for each query is unacceptable. With our distance hiding transformation we have to do a linear scan for each subtoken (that checks a specific distance) and so we see a significant (but linear) slowdown over a single linear database scan. Of particular interest are approaches that use indices that natively support k nearest neighbors but are not vulnerable to recent attacks (such as [KPT19, KE19]) and interactive solutions where the client can guide the search.

Acknowledgements The authors wish to thank Daniel Wichs for important preliminary discussion. The work of L.D. is supported by the Harriott Fellowship. C.C. is supported by NSF Award #1849904 and a fellowship from Synchrony Inc. The work of B.F. is supported by NSF Award #1849904 and ONR Grant N00014-19-1-2327. The work of A.H. is supported by NSF Award #1750795.

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. 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+21] Sohaib Ahmad, Chloe Cachet, Luke Demarest, Benjamin Fuller, and Ariel Hamlin. An implementation of proximity searchable encryption (PSE). `https://github.com/chloecachet/pse`, 2021.

[AEG+06] James Aspnes, Tolga Eren, David Kiyoshi Goldenberg, A Stephen Morse, Walter Whiteley, Yang Richard Yang, Brian DO Anderson, and Peter N Belhumeur. A theory of network localization. *IEEE Transactions on Mobile Computing*, 5(12):1663–1678, 2006.

[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 *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–10. IEEE, 2020.

[AGM+13] Joseph Akinyele, Christina Garman, Ian Miers, Matthew Pagano, Michael Rushanan, Matthew Green, and Aviel Rubin. Charm: A framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3, 06 2013.

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

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

[BCSW19] Manuel Barbosa, Dario Catalano, Azam Soleimanian, and Bogdan Warinschi. Efficient function-hiding functional encryption: From inner-products to orthogonality. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, pages 127–148, Cham, 2019. Springer International Publishing.

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

[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.

[BPCR04] Ruud M Bolle, Sharath Pankanti, Jonathan H Connell, and Nalini K Ratha. Iris individuality: A partial iris model. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 927–930. IEEE, 2004.

[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 *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 668–679, 2015.

[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.

[DRD+20] Pawel Drozdowski, Christian Rathgeb, Antitza Dantcheva, Naser Damer, and Christoph Busch. Demographic bias in biometrics: A survey on an emerging challenge. *IEEE Transactions on Technology and Society*, 1(2):89–103, 2020.

[EA11] Cem Evrendilek and Huseyin Akcan. On the complexity of trilateration with noisy range measurements. *IEEE Communications Letters*, 15(10):1097–1099, 2011.

[FMC+20] Francesca Falzon, Evangelia Anna Markatou, David Cash, Adam Rivkin, Jesse Stern, and Roberto Tamassia. Full database reconstruction in two dimensions. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 443–460, 2020.

[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.

[GKL+20] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Lucy Li, Rachit Agarwal, and Thomas Ristenpart. Pancake: Frequency smoothing for encrypted data stores. In *29th USENIX Security Symposium*, pages 2451–2468, 2020.

[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.

[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.

[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.

[Har10] William B. Hart. Fast library for number theory: An introduction. In Komei Fukuda, Joris van der Hoeven, Michael Joswig, and Nobuki Takayama, editors, *Mathematical Software – ICMS 2010*, pages 88–91, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[HBF08] Karen P Hollingsworth, Kevin W Bowyer, and Patrick J Flynn. The best bits in an iris code. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):964–973, 2008.

[HBF10] Karen Hollingsworth, Kevin W Bowyer, and Patrick J Flynn. Similarity of iris texture between identical twins. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 22–29. IEEE, 2010.

[HWKL18] Yi Huang, Adams Kong Wai-Kin, and Kwok-Yan Lam. From the perspective of cnn to adversarial iris images. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (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 *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[KE19] Evgenios M Kornaropoulos and Petros Efstathopoulos. The case of adversarial inputs for secure similarity approximation protocols. In *2019 IEEE European Symposium on Security and Privacy (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.

[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.

[KLM+16] Sam Kim, Kevin Lewi, Avradip Mandal, Hart William Montgomery, Arnab Roy, and David J Wu. Function-hiding inner product encryption is practical. *IACR Cryptology ePrint Archive*, 2016:440, 2016.

[KLM+18] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J Wu. Function-hiding inner product encryption is practical. In *International Conference on Security and Cryptography for Networks*, pages 544–562. Springer, 2018.

[KMO18] Seny Kamara, Tarik Moataz, and Olya Ohrimenko. Structured encryption and leakage suppression. pages 339–370, 2018.

[KP10] Ajay Kumar and Arun Passi. Comparison and combination of iris matchers for reliable personal authentication. *Pattern recognition*, 43(3):1016–1026, 2010.

[KPT19] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Data recovery on encrypted databases with k-nearest neighbor query leakage. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1033–1050. IEEE, 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 *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1223–1240. IEEE, 2020.

[KT14] Yutaka Kawai and Katsuyuki Takashima. Predicate- and attribute-hiding inner product encryption in a public key setting. In Zhenfu Cao and Fangguo Zhang, editors, *Pairing-Based Cryptography – Pairing 2013*, pages 113–130, Cham, 2014. Springer International Publishing.

[Lai20] Lucas Laird. Metric dimension of hamming graphs and applications to computational biology. *arXiv preprint arXiv:2007.01337*, 2020.

[Lew16] Kevin Lewi. FHIPE github. `https://github.com/kevinlewi/fhipe`, 2016.

[LMP18] M. Lacharité, B. Minaud, and K. G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 297–314, 2018.

[LTBL20] Lucas Laird, Richard C Tillquist, Stephen Becker, and Manuel E Lladser. Resolvability of hamming graphs. *SIAM Journal on Discrete Mathematics*, 34(4):2063–2081, 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.

[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.

[ODGS16] Nadia Othman, Bernadette Dorizzi, and Sonia Garcia-Salicetti. Osiris: An open source iris recognition software. *Pattern Recognition Letters*, 82:124–131, 2016.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Advances in Cryptology – CRYPTO 2010*, pages 191–208, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *Advances in Cryptology – EUROCRYPT 2012*, pages 591–608, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[OT15] Tatsuaki Okamoto and Katsuyuki Takashima. Dual pairing vector spaces and their applications. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 98(1):3–15, 2015.

[PBDT05] Nissanka Bodhi Priyantha, Hari Balakrishnan, Erik D Demaine, and Seth Teller. Mobile-assisted localization in wireless sensor networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1, pages 172–183. IEEE, 2005.

[PBF+08] P Jonathon Phillips, Kevin W Bowyer, Patrick J Flynn, Xiaomei Liu, and W Todd Scruggs. The iris challenge evaluation 2005. In *2008 IEEE Second International Conference on Biometrics: Theory, Applications and Systems*, pages 1–8. IEEE, 2008.

[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.

[Red] Redacted. Citation redacted for blinding purposes. Citation redacted for blinding purposes.

[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.

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

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *Theory of Cryptography*, pages 457–473, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

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

[TFL19] Richard C Tillquist, Rafael M Frongillo, and Manuel E Lladser. Metric dimension. *arXiv preprint arXiv:1910.04103*, 2019.

[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.

[ZD08] Sheikh Ziauddin and Matthew N Dailey. Iris recognition performance enhancement using weighted majority voting. In *2008 15th IEEE International Conference on Image Processing*, pages 277–280. IEEE, 2008.

[ZKP16] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *25th USENIX Security Symposium*, pages 707–720, 2016.

[ZR18] Kai Zhou and Jian Ren. Passbio: Privacy-preserving user-centric biometric authentication. *IEEE Transactions on Information Forensics and Security*, 13(12):3050–3063, 2018.

A Leakage Abuse Attacks

Searchable encryption achieves acceptable performance by *leaking* information to the server. See Kamara, Moataz, and Ohrimenko for an overview of leakage types in structured encryption [KMO18]. The key to attacks is combining leakage with auxiliary data, such as the frequency of values stored in the data set. Together these sources can prove catastrophic – allowing the attacker to run attacks to recover either the queries being made or the data stored in the database. We consider attacks that rely on injecting files or queries [ZKP16] to be out of scope. Common, attackable, relevant leakage profiles are:

1. *Response length leakage* [KKNO16, GLMP18] Often known as *volumetric leakage*, the attacker is given access to only the number of records returned for each query. Based on this information, attacks cross-correlate with auxiliary information about the dataset, and identify high frequency items in both the encrypted database and the auxiliary dataset.

2. *Query equality leakage* [WLD+17] the attacker is able to glean which queries are querying the same value, but not necessarily the value itself. Attacks on this profile rely on having information about the query distribution, and much like the response length leakage attacks, match with that auxiliary information based on frequency.

3. *Access pattern leakage* [IKK12, CGPR15] here the attacker is given knowledge if the same dataset element is returned for different queries. This allows the attacker to build a *co-occurrence* matrix, mapping what records are returned for pairs of queries. Based on the frequencies of the co-occurrence matrix for the encrypted dataset, and the co-occurrence matrix for the auxiliary dataset, the attack can identify records.

Recent attacks have targeted the geometry present in range search [GSB+17,LMP18,GLMP18,KPT20,FMC+20]. Building on the co-occurrence matrix (available with access patten leakage) consider the case when records a, b, c are returned by a first query and c, d are returned by a second query. One can immediately infer that the comparison relation between a and d is the same as the comparison relation between b and e. As more constraints of this type are collected one can collect an ordering of all records (up to reflection).

In two (or three) dimensional Euclidean space, trilateration has been practiced for hundreds of years: one is assumed to know the location of $x_1, ..., x_k$ and the pairwise distances $\mathcal{D}(x_i, y)$ and is trying to find the location of y. Determining the location of y requires k to be one larger than the dimension. The problem is more difficult but well studied for approximate distances [EA11]. Similar ideas can be applied in discrete metrics with each learned distance reducing the set of possible y. In the Hamming metric of dimension n, $k = \Theta(n)$ suffices [TFL19, LTBL20, Lai20].

B Multi Random Projection applied to the OT12 IPE scheme [OT12, Section 4]

To show the generality of our multi random projection technique we apply it to a second IPE scheme of Okamoto and Takashima [OT12, Section 4]. We note that this scheme is a public key scheme that is adaptively attribute-hiding against chosen plaintext attacks under the (decisional linear) DLIN assumption. This corresponds to three changes to Definition 2:

1. The adversary no longer specifies pairs of functions, only a single value,
2. The adversary can adaptively query for values f_j receiving back tk_j,
3. There is only a single challenge plaintext $x^{(0)}, x^{(1)}$ because the adversary can encrypt values on either own.

Since this scheme is public key and is not function hiding it cannot be directly used to instantiate PSE. We use it as a second example of the applicability of the transform.

B.1 Additional notation and definitions

Let $\mathbb{F}_q$ denote a finite field of order q and $GL(n, \mathbb{F}_q)$ be the general linear group of degree n over $\mathbb{F}_q$. Let the vectors $\vec{e}_i$ be defined as $\vec{e}_i = (0^{i-1}, 1, 0^{n-i})$ for $1 \leq i \leq n$. Let $\mathbb{V}$ be a vector space, to differentiate its elements from other values we will use bold letters. Let $\boldsymbol{b}_i \in \mathbb{V}$, $1 \leq i \leq n$, then we denote the subspace generated by these vectors as $\mathsf{span}(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n) \subseteq \mathbb{V}$. Consider the bases $\mathbb{B} = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)$ and $\mathbb{B}^* = (\boldsymbol{b}_1^*, \cdots, \boldsymbol{b}_n^*)$, and the vectors $\vec{x}$ and $\vec{v}$ then $(\vec{x})_{\mathbb{B}} = \sum_{i=1}^n x_i \boldsymbol{b}_i$ and $(\vec{v})_{\mathbb{B}^*} = \sum_{i=1}^n v_i \boldsymbol{b}_i^*$. Note that we will consider bases over both $\mathbb{F}_q$ and $\mathbb{G}_q$.

Definition 8 (Symmetric Bilinear Group)**.** *Suppose $\mathbb{G}, \mathbb{G}_T$ are an additive and multiplicative groups (respectively) of prime order q with generators $g \in \mathbb{G}$, and $g_T \in \mathbb{G}_T$ respectively. The group $\mathbb{G}$ uses additive notation, and the group $\mathbb{G}_T$ uses multiplicative notation. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate (i.e. $e(g, g) \neq 1$) bilinear pairing operation such that for all $x, y \in \mathbb{Z}_q$, $e(x(g), y(g)) = e(g, g)^{xy}$. Assume the group operations in $\mathbb{G}, \mathbb{G}_T$ and the pairing operation e are efficiently computable, then $(\mathbb{G}, \mathbb{G}_T, g, e)$ defines a symmetric bilinear group. Let $\mathcal{G}_{bpg}$ be an algorithm that takes input 1^λ and outputs a description of bilinear pairing groups $(q, \mathbb{G}, \mathbb{G}_T, g, e)$ with security parameter λ.*

We use the symmetric version of dual pairing vector spaces [OT15] where the pairing is based on symmetric bilinear groups defined in Definition 8.

Definition 9 (Dual Pairing Vector Spaces)**.** *Let $(q, \mathbb{G}, \mathbb{G}_T, g, e_{bg})$ be the symmetric bilinear pairing groups, then* Dual Pairing Vector Spaces *(DVPS) is a tuple of prime q, N-dimensional vector space $\mathbb{V} = \overbrace{\mathbb{G} \times \ldots \times \mathbb{G}}^{N}$ over $\mathbb{F}_q$, cyclic group $\mathbb{G}_T$ of order q, canonical basis $\mathbb{A}$ defined as:*

$$\mathbb{A} := (\vec{a}_1, \ldots, \vec{a}_n), \quad \vec{a}_i := \left(0^{i-1}, g, 0^{N-i}\right)$$

and pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The pairing e is defined with respect to e_{bg} from the symmetric bilinear pairing group $e(\vec{x}, \vec{y}) = \prod_{i=1}^N e_{bg}(g_i, h_i) \in \mathbb{G}_T$ where $\vec{x} = (g_1, \ldots, g_N) \in \mathbb{V}$ and $\vec{y} = (h_1, \ldots, h_N) \in \mathbb{V}$. This pairing is nondegenerate bilinear, i.e. $e(s\vec{x}, t\vec{y}) = e(\vec{x}, \vec{y})^{st}$ and if $e(\vec{x}, \vec{y}) = 1$ for all $\vec{y} \in \mathbb{V}$ then $\vec{x} = 0^N$. For all i and j, $e(\vec{a}_i, \vec{a}_j) = e(G, G)^{\delta_{i,j}}$ where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise, and $e(g, g) \neq 1 \in \mathbb{G}_T$.

DPVS also has a linear transformation ("canonical maps") $\phi_{i,j}$ on $\mathbb{V}$ such that $\phi_{i,j}(\vec{a}_j) = \vec{a}_i$ and $\phi_{i,j}(\vec{a}_k) = 0$ if $k \neq j$. We define $\phi_{i,j}(\vec{x}) := \left(0^{i-1}, g_j, 0^{N-i}\right)$ where $\vec{x} = (g_1, \ldots, g_N)$. We then define the dual-pairing vector space generator *as $\mathcal{G}_{\mathsf{dpvs}}$ which takes input 1^λ ($\lambda \in \mathbb{N}$) and $N \in \mathbb{N}$:*

1. *Runs $(q, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}_{bpg}\left(1^\lambda\right)$,*

2. *Compute* $\mathbb{A}, \mathbb{V}$,

3. *Returning* $(q, \mathbb{G}, \mathbb{G}_T, g, e, \mathbb{V}, \mathbb{A})$.

Lemma 1. *Let* $(q, \mathbb{G}, \mathbb{G}_T, g, e, \mathbb{V}, \mathbb{A}) \leftarrow \mathcal{G}_{\mathsf{dpvs}}$ *be a (DPVS) generator as described above. We can efficiently sample a random linear transformation* W *by sampling random coefficients* $\{r_{i,j}\}_{i,j=1,\cdots,n} \xleftarrow{\$} GL(n, \mathbb{F}_q)$ *and setting*

$$W := \sum_{i,j=1}^{n,n} r_{i,j}\phi_{i,j}.$$

The matrix $R := (r_{i,j})$ *and* $R^* := ((r_{i,j})^{-1})^T$ *then defines the adjoint action on* $\mathbb{V}$ *and we can define* $(W^{-1})^T$ *as*

$$(W^{-1})^T := \sum_{i,j=1}^{N,N} r^*_{i,j}\phi_{i,j}$$

such that for any $x, y \in \mathbb{V}$*, we have*

$$e(W(x), (W^{-1})^T(y)) = e(x, y).$$

Assumption 1 (Decisional Linear Assumption)**.** *Let* $\lambda \in N$ *and* $\beta \in \{0, 1\}$*. We define a generator for the* Decisional Linear Assumption *(DLIN) problem,* $\mathcal{G}^{\mathsf{DLIN}}_\beta$*, which on input* 1^λ*:*

1. *Samples* $\mathsf{param}_{\mathbb{G}} = (q, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathbb{G}_{bpg}(1^\lambda)$.

2. *Samples* $\kappa, \delta, \xi, \sigma \xleftarrow{\$} \mathbb{F}_q$.

3. *Sets* $Y^{(0)} = (\delta + \sigma)g$ *and* $Y^{(1)} \xleftarrow{\$} \mathbb{G}$.

4. *Returns* $(\mathsf{param}_{\mathbb{G}}, g, \xi g, \kappa g, \delta\xi g, \sigma\kappa g, Y^{(\beta)})$.

The DLIN problem then consists in guessing β *given* $(\mathsf{param}_{\mathbb{G}}, g, \xi g, \kappa g, \delta\xi g, \sigma\kappa g, Y^{(\beta)}) \leftarrow \mathcal{G}^{\mathsf{DLIN}}_\beta(1^\lambda)$*. The* decisional linear assumption *is that for any PPT distinguisher* $\mathcal{D}$ *for the DLIN problem the advantage is:*

$$\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}}(\lambda) = \Big|\ \Pr[\mathcal{D}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}^{\mathsf{DLIN}}_0(1^\lambda)] - \Pr[\mathcal{D}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}^{\mathsf{DLIN}}_1(1^\lambda)]\ \Big| = \mathsf{negl}(\lambda)$$

B.2 Construction

This construction is an adaptation of Okamoto and Takashima's IPE scheme [OT12, Section 4] (setting $\alpha = 1$ in Figure 7 yields the original scheme). As in the original construction, we first need to describe a random dual orthonormal bases generator, $\mathcal{G}^{\mathsf{IPE}^*}_{\mathsf{ob}}$, which will be called in the main construction's Setup algorithm to generate the master keys. This is different from the previous generator as it generates α sets of bases.

Construction 2 (Dual Orthonormal Bases Generator)**.** *Let* $\mathcal{G}_{\mathsf{dpvs}}$ *be a symmetric dual-pairing vector space generator as described in Definition 9. Let* $\lambda, N, \alpha \in \mathbb{N}$*, where* λ *is the security parameter,* N *is the dimension of the vector space and* α *is the number of dual orthonormal bases pairs to generate. Then on inputs* 1^λ*,* N *and* α*, the orthonormal bases generator* $\mathcal{G}^{\mathsf{IPE}^*}_{\mathsf{ob}}$ *works as follows:*

1. *Sample* $(q, \mathbb{G}, \mathbb{G}_T, g, e, \mathbb{V}, \mathbb{A}) \leftarrow \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, N)$.

2. *Sample a non-zero element of the field,* $\psi \xleftarrow{\$} \mathbb{F}^\times_q$.

3. *Set* $g_T = e(G, G)^\psi$ *and* $\mathsf{param}_{\mathbb{V}} = (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e, g_T)$.

4. *For each basis index* $1 \le \ell \le \alpha$*:*

   (a) *Sample a random map, as described in Lemma 1,* $X_\ell = (\chi_{\ell,i,j}) \xleftarrow{\$} GL(N, \mathbb{F}_q)$ *and set* $(\vartheta_{\ell,i,j}) = \psi \cdot (X^T_\ell)^{-1}$*, where* $1 \le i, j \le N$.

   (b) *For* $1 \le i \le N$*, set* $\boldsymbol{b}_{\ell,i} = \sum_{j=1}^N \chi_{\ell,i,j} \cdot \boldsymbol{a}_j$ *and* $\boldsymbol{b}^*_{\ell,i} = \sum_{j=1}^N \vartheta_{\ell,i,j} \cdot \boldsymbol{a}_j$*, where* $(\boldsymbol{a}_1, \cdots, \boldsymbol{a}_N) = \mathbb{A}$.

$\underline{\mathsf{Setup}(1^\lambda, n, \alpha)}$:

1. Sample $(\mathsf{param}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^*) \leftarrow \mathcal{G}_{\mathsf{ob}}^{\mathsf{IPE}}(1^\lambda, N)$,
2. For $1 \le \ell \le \alpha$, set $\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,0}, \cdots, \boldsymbol{b}_{\ell,n/\alpha}, \boldsymbol{b}_{\ell,N-1)})$ and $\hat{\mathbb{B}}^*_\ell = (\boldsymbol{b}^*_{\ell,0}, \cdots, \boldsymbol{b}^*_{\ell,n/\alpha}, \boldsymbol{b}^*_{\ell,3n/\alpha+1}, \cdots, \boldsymbol{b}^*_{\ell,N-1)})$.
3. $\mathsf{pk} = (1^\lambda, \mathsf{param}_{\mathbb{V}}, \{\hat{\mathbb{B}}_\ell\}_{\ell=1,\cdots,\alpha})$ and $\mathsf{sk} = \{\hat{\mathbb{B}}^*_\ell\}_{\ell=1,\cdots,\alpha}$.

$\underline{\mathsf{TokGen}(\mathsf{pk}, \mathsf{sk}, \vec{v})}$:

1. Sample $\sigma \leftarrow \mathbb{F}_q$
2. Divide $\vec{v}$ in α smaller vectors of length n/α, such that $\vec{v} = (\vec{v}_1, \cdots, \vec{v}_\alpha)$.
3. For $1 \le \ell \le \alpha$, sample $\vec{\eta}_\ell \xleftarrow{\$} \mathbb{F}_q^{n/\alpha}$ and set $\boldsymbol{k}_\ell := (\overbrace{1}^{1}, \overbrace{\sigma\vec{v}_\ell}^{n/\alpha}, \overbrace{0, \cdots, 0}^{2n/\alpha}, \overbrace{\vec{\eta}_\ell}^{n/\alpha}, \overbrace{0}^{1})_{\mathbb{B}^*_\ell}$
4. $\mathsf{tk}_{\vec{v}} := (\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\alpha)$

$\underline{\mathsf{Encrypt}(\mathsf{pk}, m, \vec{x})}$:

1. Sample $\omega \leftarrow \mathbb{F}_q$
2. Divide $\vec{x}$ in α smaller vectors of length n/α, such that $\vec{x} = (\vec{x}_1, \cdots, \vec{x}_\alpha)$.
3. For $1 \le \ell \le \alpha$, sample $\zeta_\ell, \varphi_\ell \xleftarrow{\$} \mathbb{F}_q$,
4. Set

$$\boldsymbol{c}_\ell = (\overbrace{\zeta_\ell}^{1}, \overbrace{\omega\vec{x}_\ell}^{n/\alpha}, \overbrace{0, \cdots, 0}^{3n/\alpha}, \overbrace{\varphi_\ell}^{1})_{\mathbb{B}_\ell}$$
$$\boldsymbol{c}_0 = m \cdot g_{\mathsf{T}}^{\left(\sum_{\ell=1}^{\alpha} \zeta_\ell\right)}$$

5. Return $\mathsf{ct}_{\vec{x}} := (c_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_\alpha)$

$\underline{\mathsf{Decrypt}(\mathsf{pk}, \mathsf{ct}_{\vec{x}}, \mathsf{sk}_{\vec{v}}) :}$
Return $m' = \prod_{\ell=1}^{\alpha} e(\boldsymbol{c}_\ell, \boldsymbol{k}_\ell) \;/\; c_0$

Figure 7: Description of modified IPE algorithms.

(c) Set $\mathbb{B}_\ell = (\boldsymbol{b}_{\ell,1}, \cdots, \boldsymbol{b}_{\ell,N})$ *and* $\mathbb{B}^*_\ell = (\boldsymbol{b}^*_{\ell,1}, \cdots, \boldsymbol{b}^*_{\ell,N})$.

5. Return $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \mathbb{B}^*_\ell\}_{\ell=1,\cdots,\alpha})$.

In this construction $\vec{x}$ will always denote the attribute, and $\vec{v}$ will denote the predicate. As in the original scheme, we assume that the first element of $\vec{x}$ is nonzero. Furthermore, note above we've used inner product encryption with no associated plaintext, here we include the value m which can be decrypted if the inner product is 0 and is hidden otherwise.

Construction 3. *Let* $\lambda \in \mathbb{N}$ *be the security parameter and* $n, \alpha \in \mathbb{N}$ *such that* $n/\alpha \in \mathbb{N}$ *and define* $N = 4n/\alpha + 2$. *Let* $\vec{x}, \vec{v} \in \mathbb{F}_q^n \setminus \{\vec{0}\}$ *and such that the first element of* $\vec{x}$ *is nonzero. Define the algorithms as in Figure 7.*

Correctness If the inner product of our attribute vector and our predicate vector is zero (in each basis), $\langle \vec{x}, \vec{v} \rangle = \sum_{\ell=1}^{\alpha} \langle \vec{x}_\ell, \vec{v}_\ell \rangle = 0$, then by the properties of our group structures we cancel terms,

$$\prod_{\ell=1}^{\alpha} e(\boldsymbol{c}_\ell, \boldsymbol{k}_\ell) = g_T^{\left(\sum_{\ell=1}^{\alpha} \zeta_\ell + \omega\sigma\langle \vec{x}_\ell, \vec{v}_\ell \rangle\right)} = g_T^{\left(\sum_{\ell=1}^{\alpha} \zeta_\ell\right)},$$

and finally conclude $m' = m$, therefore our construction is correct when the inner product is zero.

Key Reduction The key reduction is summarized in Table 6. In the Okamoto and Takashima scheme the DPVSs are over vectors of dimension $4n + 2$ with the public key being $n + 2$ basis vectors and the secret key being $2n + 1$. Ciphertexts and tokens are a single vector. By splitting into α bases we introduce an α overhead on each object while reducing the dimension to $4n/\alpha + 2$ and also reducing the number of basis vectors released in the public and secret key to $2n/\alpha + 1$ and $n/\alpha + 2$ respectively.

Security The proposed IPE scheme achieves the same security as the original construction [OT12, Theorem 1].

Component	Number of Group Elements
Secret Key	$8n^2/\alpha + 8n + 2\alpha$
Public Key	$4n^2/\alpha + 10n + 4\alpha$
Ciphertext	$4n + 2\alpha$
Token	$4n + 2\alpha$

Table 6: Sizes in Group Elements of Each Component of Revised Scheme. The value α is how many separate bases are used. Considering $\alpha = 1$ gives sizes for the original scheme of Okamoto and Takashima. Setting $\alpha = \Omega(n)$ makes all components a linear number of group elements.

Theorem 3. *The IPE construction in Figure 7 with $\alpha = 1$ is adaptively attribute-hiding against chosen plaintext attacks under the DLIN assumption, such that for any PPT adversary $\mathcal{A}$ there exists PPT distinguishers $\mathcal{D}_{0\text{-}1}, \mathcal{D}_{1\text{-}1}, \mathcal{D}_{0\text{-}2\text{-}h}, \mathcal{D}_{1\text{-}2\text{-}h\text{-}1}, \mathcal{D}_{1\text{-}2\text{-}h\text{-}2}$ such that for any security parameter $\lambda \in \mathbb{N}$*

$$\mathsf{Adv}^{\mathsf{IPE}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{0\text{-}1}}(\lambda) + \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{1\text{-}1}}(\lambda) + \sum_{h=1}^{\nu} \left(\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{0\text{-}2\text{-}h}}(\lambda) + \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{1\text{-}2\text{-}h\text{-}1}}(\lambda) + \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{1\text{-}2\text{-}h\text{-}2}}(\lambda) \right) + \frac{28\nu + 11}{q}$$

where $\nu \in \mathbb{N}$ is the maximum number of key queries $\mathcal{A}$ can make.[9]

This proof (like the proofs we build from) involve a system of games where each game changes a single element of a vector and is shown to be indistinguishable from the last game. These indistinguishability statements are made from a system of problems that stem from the decision linear assumption. We modify the original problems of Okamoto and Takashima [OT12] to include multiple bases of the DPVS. We can maintain security while spreading material across bases, because the public portions are incomplete and the bases are sampled independently, making it difficult to create meaningful relationships between bases. Using the same structure for our system of games and problems (but now including security with multiple bases) we show that our scheme matches the security of Okamoto and Takashima [OT12].

Proof of Theorem 3. For this theorem's proof we refer the reader to Okamoto and Takashima's proof of Theorem 1 [OT12, Section 4.3.1]. Notice that in this version Games $0', 1, 2\text{-}h\text{-}1, \cdots, 2\text{-}h\text{-}4, 3$ are replaced by Games $0^*, 1^*, 2\text{-}h\text{-}1^*, \cdots, 2\text{-}h\text{-}4^*, 3^*$ and the dimension of the hidden subspaces is $2n/\alpha$ instead of $2n$. □

Lemma 2. *For any PPT adversary $\mathcal{A}$ there exists PPT distinguishers $\mathcal{D}_1, \mathcal{D}_{2\text{-}h\text{-}1}, \mathcal{D}_{2\text{-}h\text{-}2}$ such that for any security parameter $\lambda \in \mathbb{N}$ in Game 0*,*

$$\Pr[\mathcal{A} \text{ wins} \mid t = 1] - \frac{1}{2} \leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_1}(\lambda) + \sum_{h=1}^{\nu} \left(\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{2\text{-}h\text{-}1}}(\lambda) + \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{2\text{-}h\text{-}2}}(\lambda) \right) + \frac{22\nu + 6}{q}$$

where $\nu \in \mathbb{N}$ is the maximum number of key queries $\mathcal{A}$ can make.[10]

Proof of Lemma 2. For a detailed high level overview of the proof, we refer the reader to Okamoto and Takashima's work [OT12, Section 4.3.2]. The games and the problems described in their proofs had to be updated to fit our new construction, but as in the original work, the goal is to show that indistinguishably of the games reduces to the DLIN assumption through a hierarchy of Problems. In the rest of this proof, we will describe the updated version of the needed games and problems. The tree of the reductions, from the games to the DLIN assumption, can be found in Figure 8.

We define the following $4\nu + 3$ updated games. In each game we will only describe the component that changed compared to the previous game (either the keys or the ciphertexts). The boxed parts in keys and ciphertexts indicate parts that have changed compared to the previous game.

Game 0* : This game is the same as the game described in the original proof [OT12, Definition 5] except that before the setup phase the bit $t \xleftarrow{\$} \{0, 1\}$ is sampled and the game is aborted when $t \neq s$, where $s = 1$ when $m^{(0)} = m^{(1)}$ and $s = 0$ otherwise. For this proof we only consider the case where $t = 1$ thus $m^{(0)} = m^{(1)}$ and c_0 is

[9] In the original paper the constant was $(29\nu + 17)/q$ instead of $(28\nu + 11)/q$ but the proof still holds despite this small difference.
[10] In the original paper the constant was $(23\nu + 12)/q$ instead of $(22\nu + 6)/q$ but the proof still holds despite this small difference.

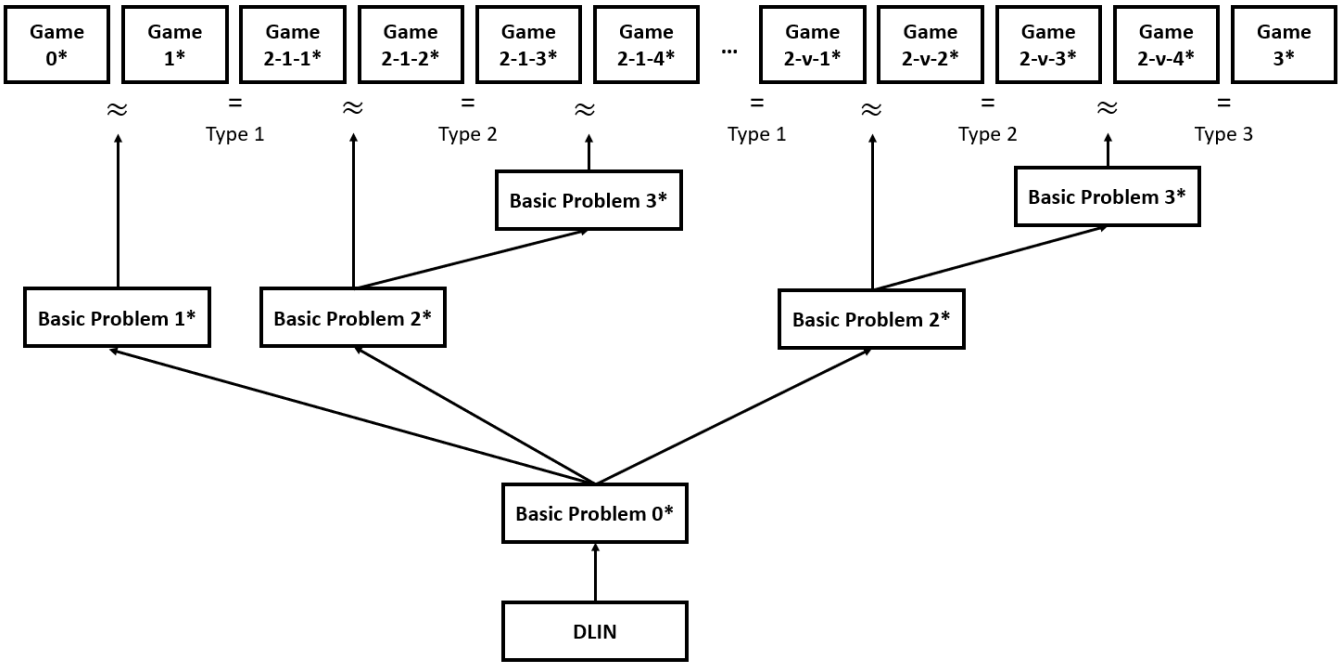


Figure 8: Structure of reductions.

independent from β. The keys and ciphertexts are built as in our construction. The answer to a key query for some vector $\vec{v} = (\vec{v}_1, \cdots, \vec{v}_\alpha)$ is

$$\boldsymbol{k}_\ell = (1, \sigma\vec{v}_\ell, 0^{n/\alpha}, 0^{n/\alpha}, \vec{\eta}_\ell, 0)_{\mathbb{B}_\ell^*}$$

where $1 \leq \ell \leq \alpha$, $\sigma \xleftarrow{\$} \mathbb{F}_q$ and $\vec{\eta}_\ell \xleftarrow{\$} \mathbb{F}_q^{n/\alpha}$. The challenge ciphertexts for attribute $\vec{x}^{(\beta)} = (\vec{x}_1^{(\beta)}, \cdots, \vec{x}_\alpha^{(\beta)})$ and message $m^{(\beta)}$ is

$$\boldsymbol{c}_\ell = (\zeta_\ell, \omega\vec{x}_\ell^{(\beta)}, 0^{n/\alpha}, 0^{n/\alpha}, 0^{n/\alpha}, \varphi_\ell)_{\mathbb{B}_\ell}$$

and

$$c_0 = m^{(\beta)} g_{\mathsf{T}}^{\left(\sum_{\ell=1}^{\alpha} \zeta_\ell\right)}$$

where $1 \leq \ell \leq \alpha$, $\beta \xleftarrow{\$} \{0,1\}$ and $\omega, \zeta_\ell, \varphi_\ell \xleftarrow{\$} \mathbb{F}_q$.

Game 1* : This game is the same as Game 0* except that the challenge ciphertexts are now

$$\boldsymbol{c}_\ell = (\zeta_\ell, \omega\vec{x}_\ell^{(\beta)}, \boxed{zx_{\ell,1}^{(\beta)}}, \boxed{0^{(n/\alpha)-1}}, 0^{n/\alpha}, 0^{n/\alpha}, \varphi_\ell)_{\mathbb{B}_\ell}$$

where $x_{\ell,1}^{(\beta)} \neq 0$ is the first coordinate of $\vec{x}_\ell^{(\beta)}$, $z \xleftarrow{\$} \mathbb{F}_q$ and all other values are generated as in Game 0*.

Game 2-h-1* : For $1 \leq h \leq \nu$, each game is the same as Game 2-(h-1)-4* (here Game 2-0-4* is Game 1*), except that the challenge ciphertexts are now

$$\boldsymbol{c}_\ell = (\zeta_\ell, \omega\vec{x}_\ell^{(\beta)}, \boxed{\omega'\vec{x}_\ell^{(\beta)}}, \boxed{\omega_0''\vec{x}_\ell^{(0)} + \omega_1''\vec{x}_\ell^{(1)}}, 0^{n/\alpha}, \varphi_\ell)_{\mathbb{B}_\ell}$$

where $\omega', \omega_0'', \omega_1'' \xleftarrow{\$} \mathbb{F}_q$ and all other values are generated as Game 2-(h-1)-4*.

Game 2-h-2* : For $1 \leq h \leq \nu$, each game is the same as Game 2-h-1*, except that the h^{th} key query for $\vec{v}$ is now

$$\boldsymbol{k}_\ell = (1, \sigma\vec{v}_\ell, \boxed{\sigma'\vec{v}_\ell}, 0^{n/\alpha}, \vec{\eta}_\ell, 0)_{\mathbb{B}^*_\ell}$$

where $\sigma' \xleftarrow{\$} \mathbb{F}_q$ and all other values are generated as in Game 2-h-1*.

Game 2-h-3* : For $1 \leq h \leq \nu$, each game is the same as Game 2-h-2*, except that the challenge ciphertexts are now

$$\boldsymbol{c}_\ell = (\zeta_\ell, \omega\vec{x}_\ell^{(\beta)}, \boxed{\omega'_0\vec{x}_\ell^{(0)} + \omega'_1\vec{x}_\ell^{(1)}}, \omega''_0\vec{x}_\ell^{(0)} + \omega''_1\vec{x}_\ell^{(1)}, 0^{n/\alpha}, \varphi_\ell)_{\mathbb{B}_\ell}$$

where $\omega'_0, \omega'_1 \xleftarrow{\$} \mathbb{F}_q$ and all other values are generated as Game 2-h-2*.

Game 2-h-4* : For $1 \leq h \leq \nu$, each game is the same as Game 2-h-3*, except that the h^{th} key query for $\vec{v}$ is now

$$\boldsymbol{k}_\ell = (1, \sigma\vec{v}_\ell, \boxed{0^{n/\alpha}, \sigma''\vec{v}_\ell}, \vec{\eta}_\ell, 0)_{\mathbb{B}^*_\ell}$$

where $\sigma'' \xleftarrow{\$} \mathbb{F}_q$ and all other values are generated as in Game 2-h-3*.

Game 3* : The game is the same as Game 2-ν-2*, except that the challenge ciphertexts are now

$$\boldsymbol{c}_\ell = \Big(\zeta_\ell, \boxed{\omega_0\vec{x}_\ell^{(0)} + \omega_1\vec{x}_\ell^{(1)}}, \omega'_0\vec{x}_\ell^{(0)} + \omega'_1\vec{x}_\ell^{(1)}, \omega''_0\vec{x}_\ell^{(0)} + \omega''_1\vec{x}_\ell^{(1)}, 0^{n/\alpha}, \varphi_\ell\Big)_{\mathbb{B}_\ell}$$

where $\omega_0, \omega_1 \xleftarrow{\$} \mathbb{F}_q$ and all other values are generated as Game 2-h-2*. Notice that with this modification, $\boldsymbol{c}_\ell$ becomes independent from the bit $\beta \xleftarrow{\$} \{0,1\}$.

Let $t = 1$, we define the advantage of a PPT machine $\mathcal{A}$ in Game g* as $\mathsf{Adv}^{(g*)}_{\mathcal{A}}(\lambda)$, where $g = 0, 1, 2\text{-}h\text{-}1, \cdots, 2\text{-}h\text{-}4, 3$. In the following proofs, we will calculate the difference of advantages for each pair of neighboring games. As in the original proof [OT12, Section 4.3.2] we then obtain

$$\begin{aligned}
\big|\, \mathsf{Adv}^{(0*)}_{\mathcal{A}}(\lambda) \,\big| &\leq \big|\, \mathsf{Adv}^{(0*)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(1*)}_{\mathcal{A}}(\lambda) \,\big| + \big|\, \mathsf{Adv}^{(2\text{-}\nu\text{-}4*)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(3*)}_{\mathcal{A}}(\lambda) \,\big| + \mathsf{Adv}^{(3*)}_{\mathcal{A}}(\lambda) \\
&\quad + \sum_{h=1}^{\nu} \Big(\big|\, \mathsf{Adv}^{(2\text{-}h\text{-}4*)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2\text{-}h\text{-}1*)}_{\mathcal{A}}(\lambda) \,\big| + \sum_{i=2}^{4} \big|\, \mathcal{A}^{(2\text{-}h\text{-}(i-1)*)}(\lambda) - \mathcal{A}^{(2\text{-}h\text{-}i*)}(\lambda) \,\big| \Big) \\
&\leq \mathsf{Adv}^{\mathsf{bp1}^*}_{\mathcal{D}_1}(\lambda) + \sum_{h=1}^{\nu} \Big(\mathsf{Adv}^{\mathsf{bp2}^*}_{\mathcal{D}_{2\text{-}h\text{-}1}}(\lambda) + \mathsf{Adv}^{\mathsf{bp3}^*}_{\mathcal{D}_{2\text{-}h\text{-}1}}(\lambda) \Big) + \frac{10\nu + 1}{q} \\
&\leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_1}(\lambda) + \sum_{h=1}^{\nu} \Big(\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{2\text{-}h\text{-}1}}(\lambda) + \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}_{2\text{-}h\text{-}2}}(\lambda) \Big) + \frac{22\nu + 6}{q}
\end{aligned}$$

In the above, bounds on $\mathsf{Adv}^{\mathsf{bp1}^*}_{\mathcal{D}_1}(\lambda), \mathsf{Adv}^{\mathsf{bp2}^*}_{\mathcal{D}_{2\text{-}h\text{-}1}}(\lambda)$ and $\mathsf{Adv}^{\mathsf{bp3}^*}_{\mathcal{D}_{2\text{-}h\text{-}1}}(\lambda)$ are described in Lemmas 4, 5 and 6 respectively. This hybrid proof relies on both computational and information theoretical problems. The computational problems are the following:

Basic problem 0* embeds a DLIN instance in the smallest and simplest dual pairing vector space possible. The resulting orthonormal bases are 3x3 matrices and are built using the random elements ξ and κ from the DLIN instance. The game is then to distinguish between a vector in which the middle element is zero and a vector in which the middle element is random.

Basic problem 1* consists in distinguishing between two challenge ciphertexts. One where the third slot contains zeros, as in the actual construction, and the second where the third slot contains a randomized copy of the second slot (i.e. the vector x).

Basic problem 2* consists in distinguishing between two challenge keys. One where the third slot contains zeros, as in the actual construction, and the second where the third slot contains a randomized copy of the second slot (i.e. the vector v).

Basic problem 3* consists in distinguishing between two challenge keys. One where the randomized vector is in the third slot and the other where it is in the fourth slot. The second slot being all zeros in both cases.

The information theoretical problems are the following:

Type 1 is a linear transformation inside a hidden subspace of a ciphertext. Lemma 7 [OT12] states that the advantage of a PPT adversary $\mathcal{A}$ in a Type 1 distinguishing game is

$$\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(h\text{-}1)\text{-}4)*}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}h\text{-}1)*}(\lambda)\right| \leq \frac{2}{q}.$$

Type 2 is a linear transformation inside a hidden subspace of a ciphertext where the corresponding token is preserved. Lemma 9 [OT12] states that the advantage of a PPT adversary $\mathcal{A}$ in a Type 2 distinguishing game is

$$\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}h\text{-}2)*}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}h\text{-}3)*}(\lambda)\right| \leq \frac{8}{q}.$$

Type 3 is a linear transformation across both hidden and partially public subspaces. Lemma 11 [OT12] states that the advantage of a PPT adversary $\mathcal{A}$ in a Type 3 distinguishing game is

$$\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\nu\text{-}4)*}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(3)*}(\lambda)\right| \leq \frac{1}{q}.$$

We now give a detailed description of the needed computational problems and their respective proofs.

B.3 Basic Problem 0*

This is a modified version of **Basic Problem 0** [OT10, Definition 18]. Let $\lambda, \alpha \in \mathbb{N}$ and $\beta \in \{0,1\}$. We define a Basic Problem 0* generator, $\mathcal{G}_\beta^{\mathsf{bp0}^*}$, which on inputs 1^λ and α:

1. Samples $\kappa, \xi, \rho, \tau \xleftarrow{\$} \mathbb{F}_q^\times$ and $\delta, \sigma, \omega \xleftarrow{\$} \mathbb{F}_q$.
2. Samples $(q, \mathbb{G}, \mathbb{G}_T, g, e, \mathbb{V}, \mathbb{A}) \leftarrow \mathcal{G}_{\mathsf{dpvs}}$ and sets $\mathsf{pp} = (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e, G_T)$ where $G_T = e(g,g)^{\kappa\xi}$.
3. For $1 \leq \ell \leq \alpha$:
   (a) Samples a random transformation, as described in Lemma 1, $X_\ell = (\chi_{\ell,1}, \chi_{\ell,2}, \chi_{\ell,3}) \xleftarrow{\$} GL(3, \mathbb{F}_q)$ and sets $(\nu_{\ell,1}, \nu_{\ell,2}, \nu_{\ell,3}) = ((X_\ell)^T)^{-1}$.
   (b) Computes $\boldsymbol{b}_{\ell,i} = \kappa \sum_{j=1}^3 \chi_{\ell,i,j} \boldsymbol{a}_j$ and sets $\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,1}, \boldsymbol{b}_{\ell,3})$.
   (c) Computes $\boldsymbol{b}^*_{\ell,i} = \xi \sum_{j=1}^3 \nu_{\ell,i,j} \boldsymbol{a}_j$ and sets $\mathbb{B}^*_\ell = (\boldsymbol{b}^*_{\ell,1}, \boldsymbol{b}^*_{\ell,2}, \boldsymbol{b}^*_{\ell,3})$.
   (d) Set $\boldsymbol{f}_\ell = (\omega, \tau, 0)_{\mathbb{B}_\ell}$.
   (e) Sets $\boldsymbol{y}_\ell^{(0)} = (\delta, 0, \sigma)_{\mathbb{B}^*_\ell}$ and $\boldsymbol{y}_\ell^{(1)} = (\delta, \rho, \sigma)_{\mathbb{B}^*_\ell}$.
4. Returns $(\mathsf{pp}, \{\hat{\mathbb{B}}_\ell, \mathbb{B}^*_\ell, \boldsymbol{y}_\ell^{(\beta)}, \boldsymbol{f}_\ell\}_{\ell=1,\cdots,\alpha}, \kappa g, \xi g, \delta\xi g)$.

Basic Problem 0* consists in guessing β given

$$(\mathsf{pp}, \{\hat{\mathbb{B}}_\ell, \mathbb{B}^*_\ell, \boldsymbol{y}_\ell^{(\beta)}, \boldsymbol{f}_\ell\}_{\ell=1,\cdots,\alpha}, \kappa g, \xi g, \delta\xi g) \leftarrow \mathcal{G}_\beta^{\mathsf{bp0}^*}(1^\lambda, \alpha).$$

We define the advantage of a PPT machine $\mathcal{A}_{\mathsf{bp0}^*}$ for Basic Problem 0* as

$$\mathsf{Adv}^{\mathsf{bp0}^*}_{\mathcal{A}_{\mathsf{bp0}^*}}(\lambda) = \left| \Pr[\mathcal{A}_{\mathsf{bp0}^*}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_0^{\mathsf{bp0}^*}(1^\lambda, \alpha)] - \Pr[\mathcal{A}_{\mathsf{bp0}^*}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_1^{\mathsf{bp0}^*}(1^\lambda, \alpha)] \right|$$

Lemma 3. *For any PPT adversary $\mathcal{A}_{bp0^*}$ for Basic Problem 0*, there exists a PPT distinguisher $\mathcal{D}$ for the DLIN problem such that for any security parameter $\lambda \in \mathbb{N}$,*

$$\mathsf{Adv}^{bp0^*}_{\mathcal{A}_{bp0^*}}(\lambda) \leq \mathsf{Adv}^{DLIN}_{\mathcal{D}}(\lambda) + \frac{5}{q}.$$

Proof. Let $\mathcal{A}_{\mathsf{bp0}^*}$ be an adversary for Basic Problem 0*. We can then build $\mathcal{D}$, a distinguisher for the DLIN assumption, as follows:

1. $\mathcal{D}$ receives a DLIN instance $(\mathsf{param}_{\mathbb{G}}, g,\ \xi g, \kappa g, \delta\xi g, \sigma\kappa g, Y^{(\beta)})$, where $\mathsf{param}_{\mathbb{G}} = (q, \mathbb{G}, \mathbb{G}_T, g, e)$ and $Y^{(\beta)}$ is either $Y^{(0)} = (\delta + \sigma)g$ or $Y^{(1)} = \psi g \xleftarrow{\$} \mathbb{G}$.

2. $\mathcal{D}$ samples $(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e) \xleftarrow{\$} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, 3, \mathsf{param}_{\mathbb{G}})$.

3. $\mathcal{D}$ computes $g_T = e(\kappa g, \xi g) = e(g, g)^{\kappa\xi}$ and sets $\mathsf{pp} = (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e, g_T)$.

4. $\mathcal{D}$ considers[11] the following basis vectors

   $$\boldsymbol{u}_1 = (\kappa, 0, 0)_{\mathbb{A}},\ \boldsymbol{u}_2 = (\text{-}\kappa, \text{-}\xi, \kappa\xi)_{\mathbb{A}},\ \boldsymbol{u}_3 = (0, \xi, 0)_{\mathbb{A}}$$

   such that $\mathbb{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3)$ is a basis of $\mathbb{V}$. Notice that from the given DLIN instance, $\mathcal{D}$ can efficiently compute $\boldsymbol{u}_1, \boldsymbol{u}_3$.

5. Similarly $\mathcal{D}$ considers

   $$\boldsymbol{u}^*_1 = (\xi, 0, 1)_{\mathbb{A}},\ \boldsymbol{u}^*_2 = (0, 0, 1)_{\mathbb{A}},\ \boldsymbol{u}^*_3 = (0, \kappa, 1)_{\mathbb{A}}$$

   such that $\mathbb{U}^* = (\boldsymbol{u}^*_1, \boldsymbol{u}^*_2, \boldsymbol{u}^*_3)$ is a basis of $\mathbb{V}$. Notice that from the given DLIN instance, $\mathcal{D}$ can efficiently compute $\boldsymbol{u}^*_1, \boldsymbol{u}^*_2, \boldsymbol{u}^*_3$.

6. $\mathcal{D}$ samples $\eta, \varphi \xleftarrow{\$} \mathbb{F}_q$ such that $\eta \neq 0$ and sets

   $$\boldsymbol{v} = (\varphi g, \text{-}\eta g, \eta\kappa g) = (\varphi, \text{-}\eta, \eta\kappa)_{\mathbb{A}}$$

   and

   $$\boldsymbol{w}^{(\beta)} = (\delta\xi g, \sigma\kappa g, Y^{(\beta)})$$

7. $\mathcal{D}$ generates α random linear transformations $W_1, \cdots, W_\alpha$ on $\mathbb{V}$, as shown in Lemma 1.

8. For $1 \leq \ell \leq \alpha$:

   (a) $\mathcal{D}$ calculates

   $$\boldsymbol{b}_{\ell,i} = W_\ell(\boldsymbol{u}_i) \text{ for } i = 1, 3,$$
   $$\boldsymbol{b}^*_{\ell,i} = (W_\ell^{-1})^T(\boldsymbol{u}^*_i) \text{ for } i = 1, 2, 3$$

   and sets $\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,1}, \boldsymbol{b}_{\ell,3})$ and $\mathbb{B}^*_\ell = (\boldsymbol{b}^*_{\ell,1}, \boldsymbol{b}^*_{\ell,2}, \boldsymbol{b}^*_{\ell,3})$

   (b) $\mathcal{D}$ sets $\boldsymbol{f}_\ell = W_\ell(\boldsymbol{v})$ and $\boldsymbol{y}^{(\beta)}_\ell = (W_\ell^{-1})^T(\boldsymbol{w}^{(\beta)})$.

9. $\mathcal{D}$ sends $(\mathsf{pp}, \{\hat{\mathbb{B}}_\ell, \mathbb{B}^*_\ell, \boldsymbol{y}^{(\beta)}_\ell, \boldsymbol{f}_\ell\}_{\ell=1,\cdots,\alpha}, \kappa g, \xi g, \delta\xi g)$ to $\mathcal{A}_{\mathsf{bp0}^*}$ and returns whatever $\mathcal{A}_{\mathsf{bp0}^*}$ sends back.

[11] In the next two steps $\mathcal{D}$ considers basis vectors of the matrices Π, Π^*,

$$\Pi = \begin{pmatrix} \kappa & & \\ \text{-}\kappa & \text{-}\xi & \kappa\xi \\ & \xi & 1 \end{pmatrix}\ \Pi^* = \begin{pmatrix} \xi & & 1 \\ & & 1 \\ & \kappa & 1 \end{pmatrix}$$

and observe that $\Pi(\Pi^*)^T = \kappa\xi I_3$. $\mathcal{D}$ cannot efficiently compute Π.

For the moment assume that η and κ are all now zero, we will later account for the probability that each could be 0 Define $\tau \stackrel{def}{=} \xi^{-1}\eta$, since $\eta \neq 0$ it holds that $\tau \neq 0$. Similarly, define $\omega \stackrel{def}{=} \tau + \kappa^{-1}\varphi$, we have

$$\begin{aligned}\boldsymbol{f}_\ell = W_\ell(\boldsymbol{v}) &= W_\ell\big((\varphi, \text{-}\eta, \eta\kappa)_{\mathbb{A}}\big) = W_\ell\big(((\omega-\tau)\kappa, \text{-}\tau\xi, \tau\kappa\xi)_{\mathbb{A}}\big) \\ &= W_\ell\big(\omega\boldsymbol{u}_1 + \tau\boldsymbol{u}_2\big) = W_\ell\big((\omega, \tau, 0)_{\mathbb{U}}\big) = (\omega, \tau, 0)_{\mathbb{B}_\ell}\end{aligned}$$

When $\beta = 0$ and $Y^{(0)} = (\delta + \sigma)g$ we have

$$\begin{aligned}\boldsymbol{y}_\ell^{(0)} &= (W_\ell^{-1})^T\big(\delta\xi g, \sigma\kappa g, (\delta+\sigma)g\big) \\ &= (W_\ell^{-1})^T\big((\delta\xi, \sigma\kappa, \delta+\sigma)_{\mathbb{A}}\big) \\ &= (W_\ell^{-1})^T\big(\delta\boldsymbol{u}_1^* + \sigma\boldsymbol{u}_3^*\big) \\ &= (W_\ell^{-1})^T\big((\delta, 0, \sigma)_{\mathbb{U}^*}\big) \\ &= (\delta, 0, \sigma)_{\mathbb{B}_\ell^*}\end{aligned}$$

When $\beta = 1$ and $Y^{(1)} = \psi g$ where $\psi \stackrel{\$}{\leftarrow} \mathbb{F}_q$, if we define $\rho = \psi - \delta - \sigma$, we have

$$\begin{aligned}\boldsymbol{y}_\ell^{(1)} &= (W_\ell^{-1})^T\big(\delta\xi g, \sigma\kappa g, \psi g\big) \\ &= (W_\ell^{-1})^T\big(\delta\xi g, \sigma\kappa g, (\rho+\delta+\sigma)g\big) \\ &= (W_\ell^{-1})^T\big((\delta\xi, \sigma\kappa, \rho+\delta+\sigma)_{\mathbb{A}}\big) \\ &= (W_\ell^{-1})^T\big(\delta\boldsymbol{u}_1^* + \rho\boldsymbol{u}_2^* + \sigma\boldsymbol{u}_3^*\big) \\ &= (W_\ell^{-1})^T\big((\delta, \rho, \sigma)_{\mathbb{U}^*}\big) \\ &= (\delta, \rho, \sigma)_{\mathbb{B}_\ell^*}\end{aligned}$$

Since the k linear maps W_ℓ are sampled uniformly and independently, the distribution of the bases $\mathbb{B}_\ell$ and $\mathbb{B}_\ell^*$ is the same as if they had been generated using $\mathcal{G}_\beta^{\mathsf{bp0}^*}$. Then for the distributions of $\boldsymbol{f}_\ell, \boldsymbol{y}_\ell^{(\beta)}$ to match the ones of the inputs expected by $\mathcal{A}$, we need $\kappa, \rho, \xi \neq 0$. This is true except with probability $2/q$ when $\beta = 0$, and with probability $3/q$ when $\beta = 1$. We then have:

$$\mathsf{Adv}^{\mathsf{bp0}^*}_{\mathcal{A}_{\mathsf{bp0}^*}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}}(\lambda) + \frac{5}{q}.$$

□

B.4 Basic Problem 1*

This is a modified version of **Problem 1** [OT12, Definition 8]. Let $\lambda, \alpha, n \in \mathbb{N}$, $\beta \in \{0,1\}$, and set $N = 4n/\alpha + 2$. We define a Basic Problem 1* generator, $\mathcal{G}_\beta^{\mathsf{bp1}^*}$, which on inputs 1^λ, α and n:

1. Samples $\omega, z \stackrel{\$}{\leftarrow} \mathbb{F}_q$.
2. Samples $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \mathbb{B}_\ell^*\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}_{\mathsf{ob}}^{\mathsf{IPE}^*}(1^\lambda, N)$.
3. For $1 \leq \ell \leq \alpha$:
   (a) Sets $\hat{\mathbb{B}}_\ell^* = (\boldsymbol{b}_{\ell,0}^*, \cdots, \boldsymbol{b}_{\ell,n/\alpha}^*, \boldsymbol{b}_{\ell,3n/\alpha+1}^* \cdots, \boldsymbol{b}_{\ell,N-1}^*)$.
   (b) Samples $\gamma_\ell \stackrel{\$}{\leftarrow} \mathbb{F}_q$.
   (c) Sets $\boldsymbol{g}_{\ell,1}^{(0)} = (0, \omega\vec{e}_1, 0^{n/\alpha}, 0^{n/\alpha}, 0^{n/\alpha}, \gamma_\ell)_{\mathbb{B}_\ell}$ and $\boldsymbol{g}_{\ell,1}^{(1)} = (0, \omega\vec{e}_1, z\vec{e}_1, 0^{n/\alpha}, 0^{n/\alpha}, \gamma_\ell)_{\mathbb{B}_\ell}$.
   (d) For $2 \leq i \leq n/\alpha$, sets $\boldsymbol{g}_{\ell,i} = \omega\boldsymbol{b}_{\ell,i}$.
4. Return
$$(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \hat{\mathbb{B}}_\ell^*, \boldsymbol{g}_{\ell,1}^{(\beta)}, \{\boldsymbol{g}_{\ell,i}\}_{i=2,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha}).$$

Then Basic Problem 1* consists in guessing β given

$$(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \hat{\mathbb{B}}^*_\ell, \boldsymbol{g}^{(\beta)}_{\ell,1}, \{\boldsymbol{g}_{\ell,i}\}_{i=2,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}^{\mathsf{bp1}^*}_\beta(1^\lambda, n, \alpha).$$

We define the advantage of a PPT machine $\mathcal{A}_{\mathsf{bp1}^*}$ for Basic Problem 1* as

$$\mathsf{Adv}^{\mathsf{bp1}^*}_{\mathcal{A}_{\mathsf{bp1}^*}}(\lambda) = \left| \Pr[\mathcal{A}_{\mathsf{bp1}^*}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}^{\mathsf{bp1}^*}_0(1^\lambda, n, \alpha)] - \Pr[\mathcal{A}_{\mathsf{bp1}^*}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}^{\mathsf{bp1}^*}_1(1^\lambda, n, \alpha)] \right|$$

Lemma 4. *For any PPT adversary $\mathcal{A}_{bp1^*}$ for Basic Problem 1*, there exists a PPT distinguisher $\mathcal{D}$ for the DLIN problem such that for any security parameter $\lambda \in \mathbb{N}$,*

$$Adv^{bp1^*}_{\mathcal{A}_{bp1^*}}(\lambda) \leq Adv^{bp0^*}_{\mathcal{A}_{bp0^*}}(\lambda) \leq Adv^{DLIN}_{\mathcal{D}}(\lambda) + \frac{5}{q}.$$

Proof of Lemma 4. Let $\mathcal{A}_{\mathsf{bp1}^*}$ be an arbitrary adversary for Basic Problem 1*. Then we can build $\mathcal{A}_{\mathsf{bp0}^*}$, an adversary for Basic Problem 0* as follows:

1. Receive a Basic Problem 0* instance
$$(\mathsf{pp}, \{\hat{\mathbb{B}}_\ell, \mathbb{B}^*_\ell, \boldsymbol{y}^{(\beta)}_\ell, \boldsymbol{f}_\ell\}_{\ell=1,\cdots,\alpha}, \kappa g, \xi g, \delta\xi g) \leftarrow \mathcal{G}^{\mathsf{bp0}^*}_\beta(1^\lambda, \alpha).$$
2. Extract g_T and $\mathsf{param}_{\mathbb{G}}(q, \mathbb{G}, \mathbb{G}_T, g, e)$ from pp and run $(q, \mathbb{G}, \mathbb{G}_T, g, e, \mathbb{V}, \mathbb{A}) \leftarrow \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, N, \mathsf{param}_{\mathbb{G}})$. Sets $\mathsf{param}_{\mathbb{V}} = (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e, g_T)$.
3. For $1 \leq \ell \leq \alpha$:
   (a) Sample a random linear transformation W_ℓ on $\mathbb{V}$, $W_\ell = (w_{\ell,1}, \cdots, w_{\ell,N}) \xleftarrow{\$} GL(N, \mathbb{F}_q)$.
   (b) Compute $\boldsymbol{g}^{(\beta)}_{\ell,1} = W_\ell(0, \boldsymbol{y}^{(\beta)}, 0^{N-4})$. (Recall that $\boldsymbol{y}^{(\beta)} \in \mathbb{G}^3$.)
   (c) For $2 \leq i \leq n$, compute $\boldsymbol{g}_{\ell,i} = W_\ell(0^i, \delta\xi g, 0^{N-i-1})$.
   (d) Compute:
$$\begin{aligned}
\boldsymbol{d}_{\ell,1} &= W_\ell(0, \boldsymbol{b}^*_{\ell,1}, 0^{N-4}),\\
\boldsymbol{d}_{\ell,n/\alpha+1} &= W_\ell(0, \boldsymbol{b}^*_{\ell,2}, 0^{N-4}),\\
\boldsymbol{d}_{\ell,N} &= W_\ell(0, \boldsymbol{b}^*_{\ell,3}, 0^{N-4}),\\
\{\boldsymbol{d}_{\ell,i} &= W_\ell(0^{i+1}, \xi g, 0^{N-i-2})\}_{i=0,2\leq i\leq n/\alpha}\\
\{\boldsymbol{d}_{\ell,i} &= W_\ell(0^i, \xi g, 0^{N-i-1})\}_{n/\alpha+2\leq i\leq N-1}.
\end{aligned}$$
   (e) Consider the following vectors ($\boldsymbol{d}^*_{\ell,n/\alpha+1}$ is not efficiently computable)
$$\begin{aligned}
\boldsymbol{d}^*_{\ell,1} &= (W^{-1}_\ell)^T(0, \boldsymbol{b}_{\ell,1}, 0^{N-4}),\\
\boldsymbol{d}^*_{\ell,n/\alpha+1} &= (W^{-1}_\ell)^T(0, \boldsymbol{b}_{\ell,2}, 0^{N-4}),\\
\boldsymbol{d}^*_{\ell,N} &= (W^{-1}_\ell)^T(0, \boldsymbol{b}_{\ell,3}, 0^{N-4}),\\
\{\boldsymbol{d}^*_{\ell,i} &= (W^{-1}_\ell)^T(0^{i+1}, \kappa g, 0^{N-i-2})\}_{i=0,2\leq i\leq n/\alpha},\\
\{\boldsymbol{d}^*_{\ell,i} &= (W^{-1}_\ell)^T(0^i, \kappa g, 0^{N-i-1})\}_{n/\alpha+2\leq i\leq N-1}.
\end{aligned}$$
   (f) $\mathcal{A}_{\mathsf{bp0}^*}$ sets $\mathbb{D}_\ell = (\boldsymbol{d}_{\ell,0}, \cdots, \boldsymbol{d}_{\ell,N})$ and $\hat{\mathbb{D}}^*_\ell = (\boldsymbol{d}^*_{\ell,1}, \cdots, \boldsymbol{d}^*_{\ell,n/\alpha}, \boldsymbol{d}^*_{\ell,3n/\alpha+1}, \cdots, \boldsymbol{d}^*_{\ell,N})$.
4. Send $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{D}_\ell, \hat{\mathbb{D}}^*_\ell, \boldsymbol{g}^{(\beta)}_{\ell,1}, \{\boldsymbol{g}_{\ell,i}\}_{i=1,\cdots,n}\}_{\ell=1,\cdots,\alpha})$ to $\mathcal{A}_{\mathsf{bp1}^*}$ and output the response bit.

From $\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,1}, \boldsymbol{b}_{\ell,3})$ and ξg, $\mathcal{A}_{\mathsf{bp0}^*}$ is only able to compute $\boldsymbol{d}^*_{\ell,i}$ for $i = 0, \cdots, n/\alpha, n/\alpha+2, \cdots, N$. From $\mathbb{B}^* = (\boldsymbol{b}^*_{\ell,1}, \boldsymbol{b}^*_{\ell,2}, \boldsymbol{b}^*_{\ell,3})$ and κg, $\mathcal{A}_{\mathsf{bp0}^*}$ is able to compute $\boldsymbol{d}_{\ell,i}$ for $i = 0, \cdots, N$. Then for $1 \leq \ell \leq \alpha$, $\mathbb{D}_\ell$ and $\mathbb{D}^*_\ell$ are dual orthonormal bases. Then when we define

$$\omega \stackrel{def}{=} \delta, \gamma \stackrel{def}{=} \sigma, z \stackrel{def}{=} \rho,$$

we have

$$\begin{aligned}
\boldsymbol{g}^{(0)}_{\ell,1} &= (0, \omega\vec{e}_1, 0^{n/\alpha}, 0^{n/\alpha}, \gamma)_{\mathbb{D}_\ell} \\
\boldsymbol{g}^{(1)}_{\ell,1} &= (0, \omega\vec{e}_1, z\vec{e}_1, 0^{n/\alpha}, \gamma)_{\mathbb{D}_\ell}
\end{aligned}$$

and for $2 \leq i \leq n$,$\boldsymbol{g}_{\ell,i} = \omega\boldsymbol{d}_{\ell,i}$. We then have $\mathsf{Adv}^{\mathsf{bp1}^*}_{\mathcal{A}_{\mathsf{bp1}^*}}(\lambda) \leq \mathsf{Adv}^{\mathsf{bp0}^*}_{\mathcal{A}_{\mathsf{bp0}^*}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}}(\lambda) + 5/q$.

Linear Algebra In the below we show that the linear system is properly prepared. Without loss of generality consider $\alpha = 1$. Then from BP0*, we have:

$$\begin{aligned}
\boldsymbol{u}^*_1 &= (\xi, 0, 1)_{\mathbb{A}} = (\xi g, 0, g) \\
\boldsymbol{u}^*_2 &= (0, 0, 1)_{\mathbb{A}} = (0, 0, g) \\
\boldsymbol{u}^*_3 &= (0, \kappa, 1)_{\mathbb{A}} = (0, \kappa g, g)
\end{aligned}$$

The matrix $(X^{-1})^T$ (from Basic Problem 0*) is a random linear transformation (i.e. a random 3×3 matrix):

$$(X^{-1})^T = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3}. \end{pmatrix}$$

As a result for $\mathbb{B}^* = (\boldsymbol{b}^*_1, \boldsymbol{b}^*_2, \boldsymbol{b}^*_3)$:

$$\begin{aligned}
\boldsymbol{b}^*_1 &= (X^{-1})^T(\boldsymbol{u}^*_1) = (X^{-1})^T(\xi g, 0, g) \\
&= \Big((x_{1,1}\xi + x_{1,3})g, (x_{2,1}\xi + x_{2,3})g, (x_{3,1}\xi + x_{3,3})g\Big) \\
\boldsymbol{b}^*_2 &= (X^{-1})^T(\boldsymbol{u}^*_2) \\
&= (X^{-1})^T(0, 0, g) \\
&= \Big(x_{1,3}g, x_{2,3}g, x_{3,3}g\Big) \\
\boldsymbol{b}^*_3 &= (X^{-1})^T(\boldsymbol{u}^*_3) \\
&= (X^{-1})^T(0, \kappa g, g) \\
&= \Big((x_{1,2}\kappa + x_{1,3})g, (x_{2,2}\kappa + x_{2,3})g, (x_{3,2}\kappa + x_{3,3})g\Big)
\end{aligned}$$

From BP1* we have the random linear transformation (i.e. random $N \times N$ matrix) W:

$$W = \begin{pmatrix} w_{1,1} & \cdots & w_{1,N} \\ \vdots & \ddots & \vdots \\ w_{N,1} & \cdots & w_{N,N} \end{pmatrix}$$

and we obtain $\mathbb{D} = (\boldsymbol{d}_0, \cdots, \boldsymbol{d}_{N-1})$ as follows:

$$\begin{aligned}
\boldsymbol{d}_j &= W(0^{j+1}, \xi g, 0^{N-j-2}) = \Big(w_{1,j+2}\xi g, \cdots, w_{N,j+2}\xi g\Big), \text{ for } j \in \{0, 2, 3, ..., n/\alpha\}, \\
\boldsymbol{d}_1 &= W(0, \boldsymbol{b}_1^*, 0^{N-4}) = W\Big(0, (x_{1,1}\xi + x_{1,3})g, (x_{2,1}\xi + x_{2,3})g, (x_{3,1}\xi + x_{3,3})g, 0^{N-4}\Big) \\
&= \Big((w_{i,2}x_{1,1} + w_{i,3}x_{2,1} + w_{i,4}x_{3,1})\xi g + (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})g\Big)_{i=1,\cdots,N} \\
\boldsymbol{d}_{n/\alpha+1} &= W(0, \boldsymbol{b}_2^*, 0^{N-4}) = W\Big(0, x_{1,3}g, x_{2,3}g, x_{3,3}g, 0^{N-4}\Big) = \Big((w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})g\Big)_{i=1,\cdots,N} \\
\boldsymbol{d}_j &= W(0^j, \boldsymbol{b}_2^*, 0^{N-j-1}) = \Big(w_{1,j+1}\xi g, \cdots, w_{N,j+1}\xi g\Big), \\
&\quad \text{for } j \in \{n/\alpha + 2, \cdots, N-1\} \\
\boldsymbol{d}_{N-1} &= W(0, \boldsymbol{b}_3^*, 0^{N-4}) = W\Big(0, (x_{1,2}\kappa + x_{1,3})g, (x_{2,2}\kappa + x_{2,3})g, (x_{3,2}\kappa + x_{3,3})g, 0^{N-4}\Big) \\
&= \Big((w_{i,2}x_{1,2} + w_{i,3}x_{2,2} + w_{i,4}x_{3,2})\kappa g + (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})g\Big)_{i=1,\cdots,N}
\end{aligned}$$

Similarly, from BP0* we have:

$$\begin{aligned}
\boldsymbol{y}^{(0)} &= (\delta, 0, \sigma)_{\mathbb{B}^*} = \Big((x_{i,1}\xi + x_{i,3})\delta g + (x_{i,2}\kappa + x_{i,3})\sigma g\Big)_{i=1,2,3}, \\
\boldsymbol{y}^{(1)} &= (\delta, \rho, \sigma)_{\mathbb{B}^*} = \Big((x_{i,1}\xi + x_{i,3})\delta g + \rho\, x_{i,3}g + (x_{i,2}\kappa + x_{i,3})\sigma g\Big)_{i=1,2,3}
\end{aligned}$$

From BP1* we have:

$$\begin{aligned}
\boldsymbol{g}_1^{(0)} &= W(0, \boldsymbol{y}^{(0)}, 0^{N-4}) \\
&= W\Big(0, (x_{1,1}\xi + x_{1,3})\delta G + (x_{1,2}\kappa + x_{1,3})\sigma g, (x_{2,1}\xi + x_{2,3})\delta g + (x_{2,2}\kappa + x_{2,3})\sigma g, (x_{3,1}\xi + x_{3,3})\delta g + (x_{3,2}\kappa + x_{3,3})\sigma g, 0^{N-4}\Big) \\
&= \Big((w_{i,2}x_{1,1} + w_{i,3}x_{2,1} + w_{i,4}x_{3,1})\delta\xi g + (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})\delta g \\
&+ (w_{i,2}x_{1,2} + w_{i,3}x_{2,2} + w_{i,4}x_{3,2})\sigma\kappa g + (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})\sigma g\Big)_{i=1,\cdots,N}
\end{aligned}$$

and

$$\begin{aligned}
\boldsymbol{g}_1^{(1)} &= W(0, \boldsymbol{y}^{(1)}, 0^{N-4}) \\
&= W\Big(0, (x_{1,1}\xi + x_{1,3})\delta g + (x_{1,2}\kappa + x_{1,3})\sigma g, (x_{2,1}\xi + x_{2,3})\delta g + (x_{2,2}\kappa + x_{2,3})\sigma g, \\
&(x_{3,1}\xi + x_{3,3})\delta g + (x_{3,2}\kappa + x_{3,3})\sigma g, 0^{N-4}\Big) \\
&= \Big((w_{i,2}x_{1,1} + w_{i,3}x_{2,1} + w_{i,4}x_{3,1})\delta\xi g + (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})\delta g \\
&+ (w_{i,2}x_{1,2} + w_{i,3}x_{2,2} + w_{i,4}x_{3,2})\sigma\kappa g + (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})\sigma g\Big)_{i=1,\cdots,N}
\end{aligned}$$

Notice that for $\omega \stackrel{def}{=} \delta$, $z \stackrel{def}{=} \rho$ and $\gamma \stackrel{def}{=} \sigma$:

$$\begin{aligned}
(0, \omega\vec{e}_1, 0^{n/\alpha}, 0^n, \gamma)_{\mathbb{D}} &= (0, \delta, 0^{n/\alpha-1}, 0^{n/\alpha}, 0^n, \sigma)_{\mathbb{D}} \\
&= \delta \boldsymbol{d}_2 + \sigma \boldsymbol{d}_N \\
&= \Big((w_{i,2}x_{1,1} + w_{i,3}x_{2,1} + w_{i,4}x_{3,1})\delta\xi g \\
&+ (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})\delta g \\
&+ (w_{i,2}x_{1,2} + w_{i,3}x_{2,2} + w_{i,4}x_{3,2})\sigma\kappa g \\
&+ (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})\sigma g \Big)_{i=1,\cdots,N} \\
&= \boldsymbol{g}_1^{(0)} \\
(0, \omega\vec{e}_1, z\vec{e}_1, 0^n, \gamma)_{\mathbb{D}} &= (0, \delta, 0^{n/\alpha-1}, \rho, 0^{n/\alpha-1}, 0^{n/\alpha}, \sigma)_{\mathbb{D}} \\
&= \delta \boldsymbol{d}_2 + \rho \boldsymbol{d}_{n/\alpha+1} + \sigma \boldsymbol{d}_N \\
&= \Big((w_{i,2}x_{1,1} + w_{i,3}x_{2,1} + w_{i,4}x_{3,1})\delta\xi g \\
&+ (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})\delta g \\
&+ (w_{i,2}x_{1,2} + w_{i,3}x_{2,2} + w_{i,4}x_{3,2})\sigma\kappa g \\
&+ (w_{i,2}x_{1,3} + w_{i,3}x_{2,3} + w_{i,4}x_{3,3})\sigma g \Big)_{i=1,\cdots,N} \\
&= \boldsymbol{g}_1^{(1)}
\end{aligned}$$

This completes the proof of Lemma 4. □

B.5 Basic Problem 2*

This is a modified version of **Problem 2** [OT12, Definition 9]. Let $\lambda, \alpha, n \in \mathbb{N}$ and $\beta \in \{0,1\}$ and set $N = 4n/\alpha + 2$. We define a Basic Problem 2* generator, $\mathcal{G}_\beta^{\mathsf{bp2}^*}(1^\lambda, \alpha, n)$:

1. Sample $\delta, \delta_0, \tau, \omega, \sigma \xleftarrow{\$} \mathbb{F}_q$.
2. Sample $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \mathbb{B}_\ell^*\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}_{\mathsf{ob}}^{\mathsf{IPE}^*}(1^\lambda, N)$.
3. For $1 \le \ell \le \alpha$ set
$$\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,0}, \cdots, \boldsymbol{b}_{\ell,n/\alpha}, \boldsymbol{b}_{\ell,3n/\alpha+1} \cdots, \boldsymbol{b}_{\ell,N}).$$
4. For $1 \le \ell \le \alpha$, for $1 \le i \le n/\alpha$:
   (a) Set $\boldsymbol{h}_{\ell,i}^{(0)} = (0, \delta\vec{e}_i, 0^{n/\alpha}, 0^{n/\alpha}, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell}$ and $\boldsymbol{h}_{\ell,i}^{(1)} = (0, \delta\vec{e}_i, \tau\vec{e}_i, 0^{n/\alpha}, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell}$.
   (b) Set $\boldsymbol{g}_{\ell,i} = (0, \omega\vec{e}_i, \sigma\vec{e}_i, 0^{n/\alpha}, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell}$.
5. Return
$$(\mathsf{param}_{\mathbb{V}}, \{\hat{\mathbb{B}}_\ell, \mathbb{B}_\ell^*, \{\boldsymbol{h}_{\ell,i}^{(\beta)}, \boldsymbol{g}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha}).$$

Basic Problem 2* is to guess β given $(\mathsf{param}_{\mathbb{V}}, \{\hat{\mathbb{B}}_\ell, \mathbb{B}_\ell^*, \{h_{\ell,i}^{(\beta)}, g_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}_\beta^{\mathsf{bp2}^*}(1^\lambda, n, \alpha)$. We define the advantage of a PPT machine $\mathcal{A}_{\mathsf{bp2}^*}$ for Basic Problem 2* as

$$\mathsf{Adv}^{\mathsf{bp2}^*}_{\mathcal{A}_{\mathsf{bp2}^*}}(\lambda) = \Big| \Pr[\mathcal{A}_{\mathsf{bp2}^*}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_0^{\mathsf{bp2}^*}(1^\lambda, n, \alpha)] - \Pr[\mathcal{A}_{\mathsf{bp2}^*}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_1^{\mathsf{bp2}^*}(1^\lambda, n, \alpha)] \Big|$$

Lemma 5. *Let $\lambda \in \mathbb{N}$ be a security parameter. For any PPT adversary $\mathcal{A}_{bp2^*}$ for Basic Problem 2*, there exists a PPT adversary $\mathcal{A}_{bp0^*}$ for Basic Problem 0* and a PPT distinguisher $\mathcal{D}$ for the DLIN problem such that,*

$$Adv^{bp2^*}_{\mathcal{A}_{bp2^*}}(\lambda) = Adv^{bp0^*}_{\mathcal{A}_{bp0^*}}(\lambda) \le Adv^{DLIN}_{\mathcal{D}}(\lambda) + \frac{5}{q}.$$

Proof of Lemma 5. Let $\mathcal{A}_{\mathsf{bp2}^*}$ be an arbitrary adversary for Basic Problem 2*. Then we can build $\mathcal{A}_{\mathsf{bp0}^*}$, an adversary for Basic Problem 0* as follows:

1. Receive a Basic Problem 0* instance
$$(\mathsf{pp}, \{\hat{\mathbb{B}}_\ell, \mathbb{B}^*_\ell, \boldsymbol{y}^{(\beta)}_\ell, \boldsymbol{f}_\ell\}_{\ell=1,\cdots,\alpha}, \kappa g, \xi g, \delta\xi g) \leftarrow \mathcal{G}^{\mathsf{bp0}^*}_\beta(1^\lambda, \alpha).$$
2. Extract g_T and $\mathsf{param}_{\mathbb{G}}(q, \mathbb{G}, \mathbb{G}_T, G, e)$ from pp, run $(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e) \leftarrow \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, N, \mathsf{param}_{\mathbb{G}})$. Set $\mathsf{param}_{\mathbb{V}} = (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e, g_T)$.
3. For $1 \le \ell \le \alpha$:
   (a) Sample a random linear transformation $W_\ell = (w_{\ell,1}, \cdots, w_{\ell,N}) \xleftarrow{\$} GL(N, \mathbb{F}_q)$.
   (b) For $1 \le i \le n/\alpha$, compute
   $$\boldsymbol{g}_{\ell,i} = W_\ell(0, 0^{3(i-1)}, \boldsymbol{f}_\ell, 0^{3(n-i)}, 0).$$
   (c) For $1 \le i \le n/\alpha$, compute
   $$\boldsymbol{h}^{(\beta)}_{\ell,i} = (W_\ell^{-1})^T(0, 0^{3(i-1)}, \boldsymbol{y}^{(\beta)}_\ell, 0^{3(N-i)}, 0).$$
   (d) Compute $\boldsymbol{d}_{\ell,0} = W_\ell(\kappa g, 0^{N-1})$ and $\boldsymbol{d}_{\ell,N} = W_\ell(0^{N-1}, \kappa g)$.
   (e) For $1 \le i \le n/\alpha$ and $1 \le j \le 3$, compute
   $$\boldsymbol{d}_{\ell,n(j-1)+i} = W_\ell(0, 0^{3(i-1)}, \boldsymbol{b}_{\ell,j}, 0^{3(n-i)}, 0).$$
   (f) Compute $\boldsymbol{d}^*_{\ell,0} = (W_\ell^{-1})^T(\xi g, 0^{N-1})$ and $\boldsymbol{d}^*_{\ell,N} = (W_\ell^{-1})^T(0^{N-1}, \xi g)$.
   (g) For $1 \le i \le n/\alpha$ and $1 \le j \le 3$, compute
   $$\boldsymbol{d}^*_{\ell,n(j-1)+i} = (W_\ell^{-1})^T(0, 0^{3(i-1)}, \boldsymbol{b}^*_{\ell,j}, 0^{3(n-i)}, 0).$$
   (h) Sets $\mathbb{D}^*_\ell = (\boldsymbol{d}^*_{\ell,0}, \cdots, \boldsymbol{d}^*_{\ell,N})$ and $\hat{\mathbb{D}}_\ell = (\boldsymbol{d}_{\ell,0}, \cdots, \boldsymbol{d}_{\ell,n/\alpha}, \boldsymbol{d}_{\ell,2n/\alpha+1}, \cdots, \boldsymbol{d}_{\ell,N})$.
4. Send
   $$(\mathsf{param}_{\mathbb{V}}, \{\mathbb{D}^*_\ell, \hat{\mathbb{D}}_\ell, \{\boldsymbol{h}^{(\beta)}_{\ell,i}, \boldsymbol{g}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$$
   to $\mathcal{A}_{\mathsf{bp2}^*}$.
5. Return β' from $\mathcal{A}_{\mathsf{bp2}^*}$.

From $\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,1}, \boldsymbol{b}_{\ell,3})$ and ξg, $\mathcal{A}_{\mathsf{bp0}^*}$ is able to compute $\boldsymbol{d}_{\ell,j}$ for $j = 0, \cdots, n/\alpha, 2n/\alpha + 1, \cdots, N$. Similarly, from $\mathbb{B}^* = (\boldsymbol{b}^*_{\ell,1}, \boldsymbol{b}^*_{\ell,2}, \boldsymbol{b}^*_{\ell,3})$ and κg, $\mathcal{A}_{\mathsf{bp0}^*}$ can compute $\boldsymbol{d}_{\ell,j}$ for $j = 0, \cdots, N$. Then for $1 \le \ell \le \alpha$, $\mathbb{D}_\ell$ and $\mathbb{D}^*_\ell$ are dual orthonormal bases. Then we have for $1 \le i \le n/\alpha$:

$$\begin{aligned}
\boldsymbol{h}^{(0)}_{\ell,i} &= (0, \delta\vec{e}_i, 0^{n/\alpha}, 0^{n/\alpha}, \sigma\vec{e}_i, 0)^*_{\mathbb{D}_\ell} \\
\boldsymbol{h}^{(1)}_{\ell,i} &= (0, \delta\vec{e}_i, \rho\vec{e}_i, 0^{n/\alpha}, \sigma\vec{e}_i, 0)^*_{\mathbb{D}_\ell} \\
\boldsymbol{g}_{\ell,i} &= (0, \omega\vec{e}_i, \tau\vec{e}_i, 0^{n/\alpha}, 0^{n/\alpha}, 0)_{\mathbb{D}_\ell}.
\end{aligned}$$

We then have
$$\mathsf{Adv}^{\mathsf{bp2}^*}_{\mathcal{A}_{\mathsf{bp2}^*}}(\lambda) = \mathsf{Adv}^{\mathsf{bp0}^*}_{\mathcal{A}_{\mathsf{bp0}^*}}(\lambda) \le \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}}(\lambda) + \frac{5}{q}.$$
This completes the proof of Lemma 5 □

B.6 Basic Problem 3*

This is a modified version of **Problem 3** [OT12, Definition 10]. Let $\lambda, \alpha, n \in \mathbb{N}$ and $\beta \in \{0,1\}$, and set $N = 4n/\alpha+2$. We define a Basic Problem 3* generator, $\mathcal{G}_\beta^{\mathsf{bp3}^*}$, which on inputs 1^λ, α and n:

1. Samples $\tau, \delta_0, \omega', \omega'', \kappa', \kappa'' \xleftarrow{\$} \mathbb{F}_q$.
2. Samples $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \mathbb{B}_\ell^*\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}_{\mathsf{ob}}^{\mathsf{IPE}^*}(1^\lambda, N, \alpha)$.
3. For $1 \le \ell \le \alpha$:
   (a) Set $\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,0}, \cdots, \boldsymbol{b}_{\ell,n/\alpha}, \boldsymbol{b}_{\ell,3n/\alpha+1} \cdots, \boldsymbol{b}_{\ell,N-1})$.
   (b) Sets $\hat{\mathbb{B}}_\ell^* = (\boldsymbol{b}_{\ell,0}, \cdots, \boldsymbol{b}_{\ell,n/\alpha}, \boldsymbol{b}_{\ell,2n/\alpha+1} \cdots, \boldsymbol{b}_{\ell,N-1})$.
4. For $1 \le \ell \le \alpha$, for $1 \le i \le n/\alpha$:
   (a) Sets $\boldsymbol{h}_{\ell,i}^{(0)} = (0, 0^{n/\alpha}, \tau\vec{e}_i, 0^{n/\alpha}, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell^*}$ and $\boldsymbol{h}_{\ell,i}^{(1)} = (0, 0^{n/\alpha}, 0^{n/\alpha}, \tau\vec{e}_i, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell^*}$.
   (b) Sets $\boldsymbol{g}_{\ell,i} = (0, 0^{n/\alpha}, \omega'\vec{e}_i, \omega''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell}$.
   (c) Sets $\boldsymbol{f}_{\ell,i} = (0, 0^{n/\alpha}, \kappa'\vec{e}_i, \kappa''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell}$.
5. Return $(\mathsf{param}_{\mathbb{V}}, \{\hat{\mathbb{B}}_\ell, \hat{\mathbb{B}}_\ell^*, \{\boldsymbol{h}_{\ell,i}^{(\beta)}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$.

Basic Problem 3* consists in guessing β given

$$(\mathsf{param}_{\mathbb{V}}, \{\hat{\mathbb{B}}_\ell, \hat{\mathbb{B}}_\ell^*, \{\boldsymbol{h}_{\ell,i}^{(\beta)}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}_\beta^{\mathsf{bp3}^*}(1^\lambda, n, \alpha).$$

We define the advantage of a PPT machine $\mathcal{A}_{\mathsf{bp3}^*}$ for Basic Problem 3* as

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{bp3}^*}}^{\mathsf{bp3}^*}(\lambda) = \left| \Pr[\mathcal{A}_{\mathsf{bp3}^*}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_0^{\mathsf{bp3}^*}(1^\lambda, n, \alpha)] - \Pr[\mathcal{A}_{\mathsf{bp3}^*}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_1^{\mathsf{bp3}^*}(1^\lambda, n, \alpha)] \right|$$

Lemma 6. *For any PPT adversary $\mathcal{A}_{bp3^*}$ for Basic Problem 3*, there exists a PPT distinguisher $\mathcal{D}$ for the DLIN problem such that for any security parameter $\lambda \in \mathbb{N}$,*

$$Adv_{\mathcal{A}_{bp3^*}}^{bp3^*}(\lambda) \le Adv_{\mathcal{A}_{bp2^*}}^{bp2^*}(\lambda) + \frac{2}{q} \le Adv_{\mathcal{D}}^{DLIN}(\lambda) + \frac{7}{q}.$$

Proof of Lemma 6. Basic Problem 3* can be decomposed into two experiments, Experiment 3-1 and 3-2 (Definitions 10 and 11 respectively). We will show that these two games are close and then use the triangle inequality. We now define these experiments.

Definition 10 (Experiment 3-1)**.** *Let $\eta \in \{0,1\}$. We define the Experiment* 3-1 *generator $\mathcal{G}_\eta^{\exp 3\text{-}1}(1^\lambda, n, \alpha)$:*

1. *Samples $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \mathbb{B}_\ell^*\}_{1\le\ell\le\alpha}) \leftarrow \mathcal{G}_{ob}^{IPE^*}(1^\lambda, N, \alpha)$.*
2. *For $1 \le \ell \le \alpha$, sets $\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,0}, \cdots, \boldsymbol{b}_{\ell,n/\alpha}, \boldsymbol{b}_{\ell,3n/\alpha+1}, \cdots, \boldsymbol{b}_{\ell,N})$ and $\hat{\mathbb{B}}_\ell^* = (\boldsymbol{b}_{\ell,0}^*, \cdots, \boldsymbol{b}_{\ell,n/\alpha}^*, \boldsymbol{b}_{\ell,2n/\alpha+1}^*, \cdots, \boldsymbol{b}_{\ell,N}^*)$.*
3. *Samples $\tau, \tau', \delta_0, \omega', \omega'', \kappa', \kappa'' \xleftarrow{\$} \mathbb{F}_q$.*
4. *For $1 \le \ell \le \alpha$, for $1 \le i \le n/\alpha$ set:*

$$\begin{aligned}
\boldsymbol{h}_{\ell,i}^{(0)} &= (0, 0^{n/\alpha}, \tau\vec{e}_i, 0^{n/\alpha}, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell^*},\\
\boldsymbol{h}_{\ell,i}^{(1)} &= (0, 0^{n/\alpha}, \tau\vec{e}_i, \tau'\vec{e}_i, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell^*},\\
\boldsymbol{g}_{\ell,i} &= (0, 0^{n/\alpha}, \omega'\vec{e}_i, \omega''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell},\\
\boldsymbol{f}_{\ell,i} &= (0, 0^{n/\alpha}, \kappa'\vec{e}_i, \kappa''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell}.
\end{aligned}$$

5. *Return* $(\mathsf{param}_{\mathbb{V}}, \{\hat{\mathbb{B}}_\ell, \hat{\mathbb{B}}^*_\ell, \{\boldsymbol{h}^{(\eta)}_{\ell,i}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$.

Experiment 3-1 *consists in guessing* $\eta \in \{0,1\}$ *given*

$$(\mathsf{param}_{\mathbb{V}}, \{\hat{\mathbb{B}}_\ell, \hat{\mathbb{B}}^*_\ell, \{\boldsymbol{h}^{(\eta)}_{\ell,i}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}_\eta^{\exp 3\text{-}1}(1^\lambda, n, \alpha).$$

We define the advantage of a PPT machine $\mathcal{D}$ *for Experiment 3-1 as*

$$\mathsf{Adv}^{\exp 3-1}_{\mathcal{D}}(\lambda) = \left| \Pr[\mathcal{D}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_0^{\exp 3-1}(1^\lambda, n, \alpha)] - \Pr[\mathcal{D}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_1^{\exp 3-1}(1^\lambda, n, \alpha)] \right|$$

Definition 11 (Experiment 3-2)**.** *Let* $\eta \in \{1,2\}$*. We define the Experiment 3-2 generator* $\mathcal{G}_\eta^{\exp 3\text{-}2}(1^\lambda, n, \alpha)$*:*

1. *Samples* $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \mathbb{B}^*_\ell\}_{1\le\ell\le\alpha}) \leftarrow \mathcal{G}^{\mathsf{IPE}^*}_{ob}(1^\lambda, N, \alpha)$.
2. *For* $1 \le \ell \le \alpha$*, sets*

$$\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,0}, \cdots, \boldsymbol{b}_{\ell,n/\alpha}, \boldsymbol{b}_{\ell,3n/\alpha+1}, \cdots, \boldsymbol{b}_{\ell,N})$$
$$\hat{\mathbb{B}}^* = (\boldsymbol{b}^*_{\ell,0}, \cdots, \boldsymbol{b}^*_{\ell,n/\alpha}, \boldsymbol{b}^*_{\ell,2n/\alpha+1}, \cdots, \boldsymbol{b}^*_{\ell,N}).$$

3. *Samples* $\tau, \tau', \delta_0, \omega', \omega'', \kappa', \kappa'' \xleftarrow{\$} \mathbb{F}_q$.
4. *For* $1 \le \ell \le \alpha$*, for* $1 \le i \le n/\alpha$ *set:*

$$\begin{aligned}
\boldsymbol{h}^{(1)}_{\ell,i} &= (0, 0^{n/\alpha}, \tau\vec{e}_i, \tau'\vec{e}_i, \delta_0\vec{e}_i, 0)_{\mathbb{B}^*_\ell},\\
\boldsymbol{h}^{(2)}_{\ell,i} &= (0, 0^{n/\alpha}, 0^{n/\alpha}, \tau'\vec{e}_i, \delta_0\vec{e}_i, 0)_{\mathbb{B}^*_\ell},\\
\boldsymbol{g}_{\ell,i} &= (0, 0^{n/\alpha}, \omega'\vec{e}_i, \omega''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell)},\\
\boldsymbol{f}_{\ell,i} &= (0, 0^{n/\alpha}, \kappa'\vec{e}_i, \kappa''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell}.
\end{aligned}$$

5. *Return* $(\mathsf{param}_{\mathbb{V}},$
$\{\hat{\mathbb{B}}_\ell, \hat{\mathbb{B}}^*_\ell, \{\boldsymbol{h}^{(\eta)}_{\ell,i}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$.

Experiment 3-2 *consists in guessing* $\eta \in \{1,2\}$ *given*

$$(\mathsf{param}_{\mathbb{V}}, \{\hat{\mathbb{B}}_\ell, \hat{\mathbb{B}}^*_\ell, \{\boldsymbol{h}^{(\eta)}_{\ell,i}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}_\eta^{\exp 3\text{-}1}(1^\lambda, n, \alpha).$$

We define the advantage of a PPT machine $\mathcal{D}$ *for Experiment* 3-2 *as*

$$\mathsf{Adv}^{\exp 3-2}_{\mathcal{D}}(\lambda) = \left| \Pr[\mathcal{D}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_1^{\exp 3-2}(1^\lambda, n, \alpha)] - \Pr[\mathcal{D}(1^\lambda, X) = 1 \mid X \leftarrow \mathcal{G}_2^{\exp 3-2}(1^\lambda, n, \alpha)] \right|$$

Lemma 7. *For any PPT distinguisher* $\mathcal{D}$ *and for any security parameter* $\lambda \in \mathbb{N}$*,*

$$\mathsf{Adv}^{\exp 3-1}_{\mathcal{D}}(\lambda) \le \frac{1}{q}$$

Proof. Sample $\theta \xleftarrow{\$} \mathbb{F}_q$. Then for $1 \le i \le n/\alpha$ set

$$\begin{aligned}
\boldsymbol{d}_{\ell,2n/\alpha+i} &= \boldsymbol{b}_{\ell,2n/\alpha+i} - \theta\boldsymbol{b}_{\ell,n/\alpha+i},\\
\boldsymbol{d}^*_{n/\alpha+i} &= \boldsymbol{b}^*_{\ell,n/\alpha+i} - \theta\boldsymbol{b}^*_{\ell,2n/\alpha+i}.
\end{aligned}$$

For $1 \le \ell \le \alpha$, define

$$\begin{aligned}
\mathbb{D}_\ell &= (\boldsymbol{b}_{\ell,0}, \cdots, \boldsymbol{b}_{\ell,2n/\alpha}, \boldsymbol{d}_{\ell,2n/\alpha+1}, \cdots, \boldsymbol{d}_{\ell,3n/\alpha}, \boldsymbol{b}_{\ell,3n/\alpha+1}, \cdots, \boldsymbol{b}_{\ell,N-1}),\\
\mathbb{D}^*_\ell &= (\boldsymbol{b}^*_{\ell,0}, \cdots, \boldsymbol{b}^*_{\ell,n/\alpha}, \boldsymbol{d}^*_{\ell,n/\alpha+1}, \cdots, \boldsymbol{d}^*_{\ell,2n/\alpha}, \boldsymbol{b}^*_{\ell,2n/\alpha+1}, \cdots, \boldsymbol{b}^*_{\ell,N-1})
\end{aligned}$$

which form dual orthonormal bases. Then we have

$$\begin{aligned}
\boldsymbol{h}_{\ell,i}^{(0)} &= (0, 0^{n/\alpha}, \tau\vec{e}_i, 0^{n/\alpha}, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell^*} \\
&= (0, 0^{n/\alpha}, \tau\vec{e}_i, \tau'\vec{e}_i, \delta_0\vec{e}_i, 0)_{\mathbb{D}_\ell^*} \\
\boldsymbol{g}_{\ell,i} &= (0, 0^{n/\alpha}, \omega'\vec{e}_i, \omega''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell} \\
&= (0, 0^{n/\alpha}, \tilde{\omega}'\vec{e}_i, \omega''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{D}_\ell} \\
\boldsymbol{f}_{\ell,i} &= (0, 0^{n/\alpha}, \kappa'\vec{e}_i, \kappa''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell} \\
&= (0, 0^{n/\alpha}, \tilde{\kappa}'\vec{e}_i, \kappa''\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{D}_\ell}
\end{aligned}$$

In the above, $\tau' = \text{-}\theta\tau$, $\tilde{\omega}' = \omega' + \theta\omega''$ and $\tilde{\kappa}' = \kappa' + \theta\kappa''$. Notice that since θ, ω' and κ' are sampled independently and uniformly, then τ', $\tilde{\omega}'$ and $\tilde{\kappa}'$ are independently and uniformly distributed except when $\tau = 0$, which happens with probability $1/q$. As a result, the distributions when $\eta = 0$ and when $\eta = 1$ are equivalent, except with probability $1/q$. □

Lemma 8. *For any PPT distinguisher $\mathcal{D}$ for Experiment 3-2, there is a PPT adversary $\mathcal{A}_{\mathsf{bp2}^*}$ for Basic Problem 2* such that for any security parameter $\lambda \in \mathbb{N}$,*

$$\mathsf{Adv}_{\mathcal{D}}^{\exp 3\text{-}2}(\lambda) \leq \mathsf{Adv}_{\mathcal{A}_{\mathsf{bp2}^*}}^{\mathsf{bp2}^*}(\lambda) + \frac{1}{q}$$

Proof. Suppose we have a PPT distinguisher $\mathcal{D}$ for Experiment 3-2, then we can build a PPT adversary $\mathcal{A}_{\mathsf{bp2}^*}$ for Basic Problem 2*. On receiving a Basic Problem 2* instance $(\mathsf{param}_V, \{\hat{\mathbb{B}}_\ell, \mathbb{B}_\ell^*, \{\boldsymbol{h}_{\ell,i}^{(\beta)}, \boldsymbol{g}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$, $\mathcal{A}_{\mathsf{bp2}^*}$ sets, for $1 \leq \ell \leq \alpha$,

$$\begin{aligned}
\mathbb{D}_\ell &= (\boldsymbol{b}_{\ell,0}, \boldsymbol{b}_{\ell,2n/\alpha+1}, \cdots, \boldsymbol{b}_{\ell,3n/\alpha}, \boldsymbol{b}_{\ell,n/\alpha+1}, \cdots, \boldsymbol{b}_{\ell,2n/\alpha}, \boldsymbol{b}_{\ell,1}, \cdots, \boldsymbol{b}_{\ell,n/\alpha}, \boldsymbol{b}_{\ell,3n/\alpha+1}, \cdots, \boldsymbol{b}_{\ell,N-1}) \\
\hat{\mathbb{D}}_\ell &= (\boldsymbol{b}_{\ell,0}, \boldsymbol{b}_{\ell,2n/\alpha+1}, \cdots, \boldsymbol{b}_{\ell,3n/\alpha}, \boldsymbol{b}_{\ell,3n/\alpha+1}, \cdots, \boldsymbol{b}_{\ell,N-1})
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{D}_\ell^* &= (\boldsymbol{b}_{\ell,0}^*, \boldsymbol{b}_{\ell,2n/\alpha+1}^*, \cdots, \boldsymbol{b}_{\ell,3n/\alpha}^*, \boldsymbol{b}_{\ell,n/\alpha+1}^*, \cdots, \boldsymbol{b}_{\ell,2n/\alpha}^*, \boldsymbol{b}_{\ell,1}^*, \cdots, \boldsymbol{b}_{\ell,n/\alpha}^*, \boldsymbol{b}_{\ell,3n/\alpha+1}^*, \cdots, \boldsymbol{b}_{\ell,N-1}^*) \\
\hat{\mathbb{D}}_\ell^* &= (\boldsymbol{b}_{\ell,0}^*, \boldsymbol{b}_{\ell,2n/\alpha+1}^*, \cdots, \boldsymbol{b}_{\ell,3n/\alpha}^*, \boldsymbol{b}_{\ell,3n/\alpha+1}^*, \cdots, \boldsymbol{b}_{\ell,N-1}^*)
\end{aligned}$$

Then $\mathcal{A}_{\mathsf{bp2}^*}$ samples $\eta_1, \eta_2 \xleftarrow{\$} \mathbb{F}_q$ and sets

$$\boldsymbol{f}_{\ell,i} = \eta_1 \boldsymbol{b}_{\ell,i} + \eta_2 \vec{e}_i, \text{ for } 1 \leq i \leq n/\alpha$$

$\mathcal{A}_{\mathsf{bp2}^*}$ sends

$$(\mathsf{param}_{\mathbb{V}}, \{\hat{D}_\ell, \hat{D}_\ell^*, \{\boldsymbol{h}_{\ell,i}^{(\beta)}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$$

to $\mathcal{D}$ and receives back $\beta' \in \{0,1\}$. $\mathcal{A}_{\mathsf{bp2}^*}$ outputs β'. Thus,

$$\begin{aligned}
\boldsymbol{h}_{\ell,i}^{(0)} &= (0, \delta\vec{e}_i, 0^{n/\alpha}, 0^{n/\alpha}, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell^*} \\
&= (0, 0^{n/\alpha}, 0^{n/\alpha}, \delta\vec{e}_i, \delta_0\vec{e}_i, 0)_{\mathbb{D}_\ell^*} \\
\boldsymbol{h}_{\ell,i}^{(1)} &= (0, \delta\vec{e}_i, \tau\vec{e}_i, 0^{n/\alpha}, \delta_0\vec{e}_i, 0)_{\mathbb{B}_\ell^*} \\
&= (0, 0^{n/\alpha}, \tau\vec{e}_i, \delta\vec{e}_i, \delta_0\vec{e}_i, 0)_{\mathbb{D}_\ell^*} \\
\boldsymbol{g}_{\ell,i} &= (0, \omega\vec{e}_i, \sigma\vec{e}_i, 0^{n/\alpha}, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell} \\
&= (0, 0^{n/\alpha}, \sigma\vec{e}_i, \omega\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{D}_\ell} \\
\boldsymbol{f}_{\ell,i} &= (0, (\eta_1 + \eta_2\omega)\vec{e}_i, \eta_2\sigma\vec{e}_i, 0^{n/\alpha}, 0^{n/\alpha}, 0)_{\mathbb{B}_\ell} \\
&= (0, 0^{n/\alpha}, \eta_2\sigma\vec{e}_i, (\eta_1 + \eta_2\omega)\vec{e}_i, 0^{n/\alpha}, 0)_{\mathbb{D}_\ell}.
\end{aligned}$$

Since $\delta, \tau, \omega, \sigma, \eta_1$ and η_2 are independently and uniformly sampled, then $\delta, \tau, \omega, \sigma, \eta_1 + \eta_2\omega$ and $\eta_2\sigma$ are independently and uniformly distributed in $\mathbb{F}_q$ except when $\sigma = 0$, which happens with probability $1/q$. As a result, the distributions of $(\mathsf{param}_{\mathbb{V}}, \{\hat{D}_\ell, \hat{D}^*_\ell, \{\boldsymbol{h}^{(\beta)}_{\ell,i}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$ and of the output of $\mathcal{G}^{\exp\,3\text{-}2}_\beta$ are equivalent except with probability $1/q$. □

Then from Lemmas 7, 8 and 5, for any PPT adversary $\mathcal{A}_{\mathsf{bp3}^*}$ there exists PPT adversaries, $\mathcal{A}_{\mathsf{bp2}^*}$ and $\mathcal{A}_{\mathsf{DLIN}^*}$, such that for any security parameter $\lambda \in N$ we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{bp3}^*}_{\mathcal{A}_{\mathsf{bp3}^*}}(\lambda) &\leq \left| \Pr[\mathcal{A}_{\mathsf{bp3}^*}(1^\lambda, \mathcal{G}^{\exp\,3\text{-}1}_0(1^\lambda, n, \alpha)) = 1] - \Pr[\mathcal{A}_{\mathsf{bp3}^*}(1^\lambda, \mathcal{G}^{\exp\,3\text{-}2}_2(1^\lambda, n, \alpha)) = 1] \right| \\
&\leq \left| \Pr[\mathcal{A}_{\mathsf{bp3}^*}(1^\lambda, \mathcal{G}^{\exp\,3\text{-}1}_0(1^\lambda, n, \alpha)) = 1] - \Pr[\mathcal{A}_{\mathsf{bp3}^*}(1^\lambda, \mathcal{G}^{\exp\,3\text{-}1}_1(1^\lambda, n, \alpha)) = 1] \right| \\
&+ \left| \Pr[\mathcal{A}_{\mathsf{bp3}^*}(1^\lambda, \mathcal{G}^{\exp\,3\text{-}2}_1(1^\lambda, n, \alpha)) = 1] - \Pr[\mathcal{A}_{\mathsf{bp3}^*}(1^\lambda, \mathcal{G}^{\exp\,3\text{-}2}_2(1^\lambda, n, \alpha)) = 1] \right| \\
&\leq \mathsf{Adv}^{\exp\,3\text{-}1}_{\mathcal{A}_{\mathsf{bp3}^*}}(\lambda) + \mathsf{Adv}^{\exp\,3\text{-}2}_{\mathcal{A}_{\mathsf{bp3}^*}}(\lambda) \leq \mathsf{Adv}^{\mathsf{bp2}^*}_{\mathcal{A}_{\mathsf{bp2}^*}(\lambda)} + \frac{2}{q} \leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{A}_{\mathsf{DLIN}}(\lambda)} + \frac{7}{q}
\end{aligned}$$

This completes the proof of Lemma 6. □

This completes the proof of Lemma 2. □