# Proximity Searchable Encryption for Biometrics

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

September 25, 2020

**Abstract**

Biometric databases collect entire countries worth of citizens' sensitive information with few cryptographic protections. The critical required functionality is proximity search, the ability to search for all records *close* to a queried value, that is within a bounded distance. Biometrics usually operate in high dimensional space where an exponential number (in the dimension) of values are close.

This work builds searchable encryption that supports proximity queries for the Hamming metric. The Hamming metric is frequently used for the iris biometric. Searchable encryption schemes have *leakage*, which is information revealed to the database server such as identifiers of records returned which is known as *access pattern leakage*.

Prior work on proximity searchable encryption falls into two classes: 1) Li et al. (INFOCOM 2010) and Boldyreva and Chenette (FSE 2014) support only a polynomial number of close values, 2) Kim et al. (SCN 2018) leak the distance between the query and all stored records. The first class is not feasible due to the exponential number of close values. The second class allows the server to compute geometry of the space, enabling attacks akin to those on nearest neighbor schemes (Kornaropoulos et al. IEEE S&P 2019, 2020).

We build proximity search out of a new variant of inner product encryption called multi-point inner product encryption (MPIPE). MPIPE is built from function-hiding, secret-key, inner product predicate encryption (Shen, Shi, and Waters, TCC 2009). Our construction leaks access pattern and when two database records are the same distance from the queried point.

In most applications of searchable encryption the data distribution is not known a priori, making it prudent to consider leakage in a variety of settings. However, biometrics' statistics are well studied and static. Frequently in biometric search at most one record is returned. In this setting, access pattern leakage and the additional leakage of distance equality is unlikely to be harmful.

We also introduce a technique for reducing key size of a class of inner product encryption schemes based on dual pairing vector spaces. Our technique splits these vector spaces into multiple, smaller components, yielding keys that are a linear number of group elements instead of quadratic. We instantiate this technique on the scheme of Okamoto and Takashima (Eurocrypt, 2012) and show security under the same assumption (decisional linear).

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

## 1 Introduction

The Aadhaar system in India links a citizen's biometrics with a unique 12 digit number with over 1 billion numbers issued [Dau14]. Argentina and Kenya also hold biometric databases of their population [Fou]. The Tribune newspaper purchased full access to Aardhar database for $13 [Kha18]. Access to these databases can prove to be extremely damaging to citizens. Biometrics, by their very definition, cannot be updated if an adversary learns them. As we can see access is not tightly controlled for many existing databases. These databases need cryptographic protections targeted at supporting searching over biometric data. We focus on constructing a specific searchable encryption scheme for biometric databases.

*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

In searchable encryption, a client $C$ wishes to outsource data to a semi-honest server $S$. $S$ is provided with an index $\mathcal{I}$ (and encrypted records). The goal of searchable encryption is for $S$ to be able to return the appropriate records while minimizing $S$'s knowledge, called *leakage*, about the stored data or queries. Searchable encryption techniques exist for keyword equality [SWP00], set operations over keyword equality [CJJ+13, PKV+14], and selection, projection and cross product [KM18]. See Bösch et al. [BHJP14] and Fuller et al. [FVY+17] for reviews of the area.

There are two primary general approaches to searchable encryption. *Property preserving encryption* retains compatibility with existing database indexing mechanisms by retaining a property of data, such as equality [BFOR08], but destroying other structure. The second is called *structured encryption* which creates a new indexing mechanism that requires server side changes. Searchable encryption has a complicated history with constructions and attacks based on the leakage starting with the work of Cash et al. [CGPR15]. Property preserving encryption is subject to much stronger attacks, in the case of order preserving encryption, Grubbs et al. fully reconstructed databases up to a symmetry [GSB+17]. We focus on structured encryption.

For biometric databases, we wish to support searches for values that are *close* to a queried value. Suppose that each record $r_i$ consists of only some biometric value. The goal is to be able to create an index $\mathcal{I}$ and tokens $\mathsf{tk}_{r^*}$ such that the server $S$ given $(\mathcal{I}, \mathsf{tk}_{r^*})$ can compute the set $W_{r^*} = \{r_i | \mathcal{D}(r_i, r^*) \leq t\}$ where $\mathcal{D}$ is a distance metric and $t$ is some defined distance parameter.[1] We call this *proximity search* and focus on building searchable encryption for proximity search.

**Prior approaches to proximity search** 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. The first when adding a record $r_i$ to a database, also inserts all close values as keywords, that is $\{r_j | \mathcal{D}(r_i, r_j) \leq t\}$ are added as keywords associated to $r_i$. The second approach requires a searchable encryption scheme that supports disjunctive search. It inserts just $r_i$, but when searching for $r^*$ searches for the disjunction $\vee_{r_j | \mathcal{D}(r_j, r^*) \leq t} \; r_j$. 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 $\vee_{r_i | \mathcal{D}(r_i, r^*) \leq t} \{r_i\}$ 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 $t$. Looking ahead, our approach can be viewed as a disjunction of distances, the size of this disjunction at most $n$.

Kim et al. [KLM+18] use function hiding inner product encryption ($\mathsf{IPE}$) [BJK15] to store records on the server. Inner product encryption creates a system of ciphertexts and tokens that when combined allow computation of an inner product. More precisely, one encrypts values $r_i$ into $c_{r_i}$ which are stored on the server, then for a value $r^*$ one creates a token $\mathsf{tk}_{r^*}$. Given a ciphertext $c_{r_i}$ and token $\mathsf{tk}_{r^*}$, the server can compute $\langle r_i, r^* \rangle$. Function hiding means that it is hard for the server to learn the value $r^*$.

Kim et al. observe for binary vectors with the Hamming metric (the number of positions that differ) that if records are encoded as vectors in $\{-1, 1\}$ then $\mathcal{D}(r_i, r^*) = (n - \langle r_i, r^* \rangle)/2$ for $|r_i| = n$. Thus, revealing inner product allows computation of distance. By design, the server can infer the distance of every record with respect to the query. This allows the server to establish the geometry of the space by correlating distances across queries. Recent work has shown how to attack nearest neighbor systems using similar information [KPT19, MT19, KPT20].

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. For some value $r_i$, the client samples $k$ locality sensitive hashes and computes $h_1(r_i), \cdots, h_k(r_i)$. The client then inserts $h_1(r_i), \cdots, h_k(r_i)$ as keywords for the record $r_i$ in the database. When querying for value $r^*$ the client computes the hashes of $r^*$, $h_1(r^*), \cdots, h_k(r^*)$. The server then returns every record for which at least one hash matches. Thus, a scheme can be built from any scheme supporting disjunctive keyword equality, inheriting any leakage. Since $S$ 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. With some error, this allows the server to establish the distance between each record and the query, again enabling the server to establish geometry.

[1] *Only* points within the distance $t$ are returned, *not* the closest point or $k$-closest points as in nearest neighbor systems [RKV95].

The existence of leakage means that to understand a searchable encryption scheme requires understanding the cryptographic guarantees, attacker posture, and data. This whole system perspective is at odds with the goal of building general purpose database systems. However, the statistical properties of biometric data are well understood and stable over time. Thus, they represent a prime target for developing searchable encryption using this whole system approach.

As we discuss in Section 3, passive leakage abuse attacks rely on correlating records that are jointly returned by some $\mathsf{tk}_r$ over a sequence of queries. In good biometric systems the goal is to return at most 1 record. Thus, we establish the goal of building a proximity searchable encryption system that leaks no information under the condition that at most one record is returned with each query. As discussed in Section 3, this is feasible for the iris biometric.

**Our Contribution** The primary contributions of this work are introducing:

1. A proximity searchable encryption scheme built on function-hiding, secret-key, inner-product predicate encryption (first constructed by Shen et al. [SSW09]), that we call MPIPE for multi-point inner product encryption.[2] As there are many variants of inner product encryption or IPE, we use $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ to denote the function-hiding, secret-key, predicate variant (similarly $\mathsf{MPIPE}_{\mathsf{fh,sk,pred}}$).[3] For a biometric of dimension $n$, our ciphertexts are a single IPE ciphertext on dimension $n+1$, our tokens consist of $(t+1)$ IPE tokens on dimension $n+1$.

   Our scheme has access pattern leakage. In addition, our scheme leaks if two records are the same distance from the queried record. Since biometric data is usually far apart, a system can be tuned to rarely return multiple records, establishing a tradeoff between true accept rate and leakage. Section 3 discusses this tradeoff when 1) queries are assumed to have the same distribution as the biometric and 2) when queries are assumed to be arbitrarily distributed.
2. A transform to reduce key sizes of inner product encryption schemes based on dual pairing vector spaces while retaining the same functionality. This transformation allows for a reduction of key sizes in IPE from quadratic in the dimension to linear without effecting the asymptotic size of ciphertexts or tokens. Our transform builds on recent work in multi-input IPE [AGRW17, DOT18] but is from IPE to IPE, only borrowing ideas. A similar technique can also be used to build an unbounded IPE scheme [TT20].

**Proximity Searchable Encryption Scheme from IPE** In Section 4, we introduce *Multi-point inner product encryption* (MPIPE) that allows for testing if the Hamming distance is in a set without leaking the distance. This scheme yields a searchable encryption scheme that supports proximity testing (within distance $t$). We first define $\mathsf{MPIPE}_{\mathsf{fh,sk,pred}}$ and show it can be built from $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$. We then show that $\mathsf{MPIPE}_{\mathsf{fh,sk,pred}}$ suffices to construct a proximity searchable encryption scheme. Before describing our MPIPE primitive we introduce notation for $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$. $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ allows one to generate ciphertexts $c_{\vec{x}}$ for a plaintext value $\vec{x}$ and tokens $\mathsf{tk}_{\vec{y}}$ for a value $\vec{y}$. An evaluator can compute $\mathsf{Decrypt}(c_{\vec{x}}, \mathsf{tk}_{\vec{y}})$ which outputs 1 if $\langle c_{\vec{x}}, \mathsf{tk}_{\vec{y}} \rangle = 0$ and 0 otherwise. The evaluator should learn nothing about values $\vec{x}$ and $\vec{y}$ other than whether their inner product is 0. We build $\mathsf{MPIPE}_{\mathsf{fh,sk,pred}}$ in two steps:

1. $\mathsf{IPE}_{\mathsf{fh,sk,pred}}$ implies the ability to test if the inner product is equal to some other value (without revealing this value). This can be achieved by adding the value as the $n+1^{th}$ element to the first vector and adding a -1 element to the second vector. That is,

$$\langle \vec{x} \;||\; \text{-1}, \vec{y} \;||\; d \rangle = 0 \Leftrightarrow \langle \vec{x}, \vec{y} \rangle = d.$$

2. We then build $\mathsf{MPIPE}_{\mathsf{fh,sk,pred}}$ by testing multiple points by creating $t$ different tokens. Our method minimizes ciphertext size, generating just a single ciphertext with a -1 appended. It then generates $t$ tokens corresponding to distances $d_1, ..., d_t$:

$$\mathsf{tk}'_{y,\{d_1,...,d_t\}} = \pi\left(\mathsf{tk}_{y||d_1}, \mathsf{tk}_{y||d_2}, ...., \mathsf{tk}_{y||d_t}\right).$$

[2] We use public key and secret key and token in place of master public key, master secret key, and function key respectively.

[3] The predicate property restricts information learned by the server, only allowing them to learn when the inner product is 0, rather than learning the inner product.

In the above $\pi$ is a random permutation that hides which distance matched. This permutation is freshly sampled with the creation of each $\mathsf{tk}'$. Decryption proceeds by calling the $\mathsf{IPE}$ decryption for all $t$ tokens returning 1 if and only if exactly one $\mathsf{IPE}$ decryption returns 1. In this construction the adversary can learn if two values, $x_1$ and $x_2$, match the same point in the set $\{d_1, ..., d_t\}$ That is, the adversary learns the predicate $\mathcal{D}(x_1, y) \stackrel{?}{=} \mathcal{D}(x_2, y)$, otherwise called *distance equality leakage*. One can switch the role of ciphertexts and tokens if one expects the volume of queries to be much larger than the volume of the database.

This leads us to our first informal theorem (formally Theorem 1).

**Theorem** (IPE to MPIPE)**.** *Given a secure construction of $\mathit{IPE}_{\mathsf{fh},\mathsf{sk},\mathit{pred}}$ there is a secure construction of $\mathit{MPIPE}_{\mathsf{fh},\mathsf{sk},\mathit{pred}}$ subject to leaking the equality pattern of matched distances.*

Using Kim et al.'s observation [KLM+18] on the equivalence between inner product and binary Hamming distance implies that $\mathsf{MPIPE}_{\mathsf{fh},\mathsf{sk},\mathsf{pred}}$ yields a proximity searchable encryption (see Theorem 3):

**Theorem** (IPE yields proximity searchable encryption)**.** *Given a secure construction of $\mathit{IPE}_{\mathsf{fh},\mathsf{sk},\mathit{pred}}$ there is a proximity searchable encryption scheme that leaks only 1) number of returned records and identifiers (access pattern) and 2) whether two records are the same distance from queried value.*

**MPIPE Variant** While $\mathsf{MPIPE}$ hides the *value* of the distance matched, it still leaks when two ciphertexts match the same distance for a token. We mention a variant of $\mathsf{MPIPE}_{\mathsf{fh},\mathsf{sk},\mathsf{pred}}$ which eliminates leakage when distances match on a *single* token and is directly built from $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk},\mathsf{pred}}$. We believe it is not trivial to extend it beyond a single token and leave it open for future work.

This variant is less efficient than the above, it further extends the dimension required of the underlying $\mathsf{IPE}$. Let $\pi_{\vec{x}}, \pi_{\vec{y}}$ be random permutations. This variant concatenates the values of the range to the first vector generating an $\mathsf{IPE}$ token for $\vec{y}' = (\vec{y}||\pi_{\vec{y}}(d_1||d_2||...||d_t))$. It then generates multiple $\mathsf{IPE}$ ciphertexts as follows:

$$\mathsf{ct}_{\vec{x}'} = \pi_{\vec{x}} \left( \mathsf{ct}_{\vec{x}||-1||0||....||0}, \mathsf{ct}_{\vec{x}||0||-1||....||0}, ...., \mathsf{ct}_{\vec{x}||0||0||...||-1} \right).$$

Similarly to the previous construction, during decryption, $\mathsf{MPIPE}$ calls the $\mathsf{IPE}$ decryption algorithm on the token and each ciphertext and returns 1 only when some $\mathsf{IPE}$ decryption returns 1. For biometrics of dimension $n$, calls are made to a dimension $(n + t + 1)$ $\mathsf{IPE}_{\mathsf{fh},\mathsf{sk},\mathsf{pred}}$ scheme with $\mathsf{MPIPE}$ ciphertext consisting of $(t + 1)$ $\mathsf{IPE}$ ciphertexts.

**Reducing Key Size in IPE** Our second contribution is a technique for reducing key size in $\mathsf{IPE}$ variants. To introduce the idea, we first need to introduce an object used in many $\mathsf{IPE}$ systems called dual pairing vector space (DPVS). DPVSs are used in bilinear groups to build $\mathsf{IPE}$. A bilinear group is a pair of groups $(\mathbb{G}, \mathbb{G}_T)$ equipped with a group operation and an additional pairing operation that allows for a single *multiplication* (we assume all groups are of prime order of sufficient size throughout). That is, there is a pairing $e(xg, yg) = e(g, g)^{xy}$ for generator $g \in \mathbb{G}$. (Following the notation of Okamoto and Takashima we use additive notation for the source group and multiplicative in the target group [OT12].) In a DPVS system, there are two bases $\mathbb{B}$ and $\mathbb{B}^*$. These bases have the property that for $\vec{b}_i \in \mathbb{B}$ and $\vec{b}^*_j \in \mathbb{B}^*$ then

$$\langle \vec{b}_i, \vec{b}^*_j \rangle = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases}.$$

The high-level idea of a scheme is to sample $\mathbb{B}$ and $\mathbb{B}^*$ randomly (subject to the basis restrictions above), and then project $\vec{x}$ and $\vec{y}$ into the bases $\mathbb{B}$ and $\mathbb{B}^*$ and encode them in $\mathbb{G}$. When computing the pairing between one computes $\langle x\mathbb{B}, \mathbb{B}^* y \rangle = \langle x, y \rangle$. For a basis $\mathbb{B}$ we use the notation $(\vec{x})_{\mathbb{B}}$ to indicate that $\vec{x}$ is first appropriately mapped into the basis and then encoded into $\mathbb{G}$. To demonstrate our technique, we start from the scheme of Okamoto and Takashima [OT12, Section 4] which is a public key inner product scheme $\mathsf{IPE}_{\mathsf{pk}}$.[4]

[4]Okamoto and Takashima also introduce a technique for reducing key size, we believe our technique is conceptually simpler and complementary to their technique.

In this scheme vectors are of size $4n + 2$ as follows:

$$c_{\vec{x}'} = (\zeta, \omega \cdot \vec{x}, 0^{2n}, 0^n, \varphi)_{\mathbb{B}},$$
$$\mathsf{tk}_{\vec{y}'} = (1, \sigma \cdot \vec{y}, 0^{2n}, \eta_1, ..., \eta_n, 0)_{\mathbb{B}^*}.$$

In addition, the ciphertext includes an encoding of $\zeta$ in the target group, $c_0 = g_T^{\zeta}$, so one can compute:

$$g_T^{(\langle \vec{x}', \vec{y}' \rangle - \zeta)} = g_T^{\omega\sigma\langle x, y \rangle}$$

and check if this value is the identity in the target group. In the above, all greek letters $(\zeta, \varphi, \sigma, \omega, \eta_i)$ are random field elements. The secret key in the above is the two bases $\mathbb{B}, \mathbb{B}^*$ which are a quadratic number of field elements.

Our core idea is that one needs not to have a DPVS basis pair for the entire vector, instead one can compute $\alpha$ pairs of DPVS bases that allow for partial computation of the inner product as long this intermediate value is blinded until all factors have been included. Since the size of bases grow quadratically with the number of dimensions, by creating $\alpha$ copies the size of each basis reduces by a factor of $\alpha^2$, resulting in an overall reduction by a factor of $\alpha$. Suppose that $\{\mathbb{B}_i, \mathbb{B}_i^*\}_{i=1}^{\alpha}$ are $\alpha$-pairs of DPVS bases (consisting of $4n/\alpha + 2$ vectors of dimension of $4n/\alpha + 2$), our ciphertexts now consist of $\alpha + 1$ values and tokens consist of $\alpha$ values as follows:

$$c_{\vec{x}} = \left\{ \left\{ (\overbrace{\zeta_i}^{1}, \overbrace{\omega\vec{x}_i}^{n/\alpha}, \overbrace{0, \ldots, 0}^{n/\alpha}, \overbrace{0, \ldots, 0}^{n/\alpha}, \overbrace{0, \ldots, 0}^{n/\alpha}, \overbrace{\varphi_i}^{1})_{\mathbb{B}_i} \right\}_{i=1}^{\alpha}, \right.$$
$$\left. g_T^{(\sum_{i=1}^{\alpha} \zeta_i)} \right\},$$
$$\mathsf{tk}_{\vec{y}} = \left\{ (\overbrace{1}^{1}, \overbrace{\sigma\vec{y}_i}^{n/\alpha}, \overbrace{0, \ldots, 0}^{n/\alpha}, \overbrace{0, \ldots, 0}^{n/\alpha}, \overbrace{\vec{\eta}_i}^{n/\alpha}, \overbrace{0}^{1})_{\mathbb{B}_i^*} \right\}_{i=1}^{\alpha}.$$

Note, $\vec{x}, \vec{y}$ are split into $\alpha$ parts each of size $n/\alpha$ and $\vec{x}_i$ refers to the $i$th component. Each pair $\mathbb{B}_i, \mathbb{B}_i^*$ is a DPVS basis pair. We show security of this transformation in the public key setting based on the decisional linear assumption as in Okamoto and Takashima [OT12, Section 4]. We believe this technique can be applied across a number of inner product encryption variants [TAO16].

**Organization** The rest of this work is organized as follows, Section 2 describes mathematical and cryptographic preliminaries, Section 3 describes prior leakage attacks and how to configure a biometric database so these attacks are mitigated, Section 4 introduces our proximity search, Section 5 describes our transform for reducing key size.

# 2 Preliminaries

Let $\lambda$ 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 $\lambda$, respectively. 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 $q$ and $\mathbb{F}_q$ denotes a finite field of order $q$. We use $\mathbb{F}_q^{\times}$ as a shorter notation for $\mathbb{F}_q \setminus \{0\}$. Let $GL(n, \mathbb{F}_q)$ denote the general linear group of degree $n$ over $\mathbb{F}_q$. Let $\vec{x}$ denote a vector over $\mathbb{F}_q$ such that $\vec{x} = (x_1, \cdots, x_n) \in \mathbb{F}_q^n$, the dimension of vectors should be apparent from context. Let the vectors $\vec{e}_i$ be defined as $\vec{e}_i = (0^{i-1}, 1, 0^{n-i})$ for $1 \leq i \leq n$. Consider vectors $\vec{x} = (x_1, \cdots, x_n)$ and $\vec{v} = (v_1, \cdots, v_n)$, 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.

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 basis over both $\mathbb{F}_q$ and $\mathbb{G}_q$.

Hamming distance is defined as the distance between the bit vectors $x$ and $y$: $d(x,y) = |\{i \,|\, x_i \neq y_i\}|$. We note that if a vector is encoded as follows:

$$x_{\pm 1} \stackrel{def}{=} \begin{cases} x_{\pm 1,i} = 1 & \text{if } x_i = 1 \\ x_{\pm 1,i} = -1 & \text{if } x_i = 0. \end{cases}$$

Then it is true that $\langle x_{\pm 1}, y_{\pm 1} \rangle = n - 2d(x,y)$.

## 2.1 Secret Key Inner Product Predicate Encryption

Secret-key predicate encryption with function privacy supporting inner products queries was first proposed by Shen et al. [SSW09]. This primitive allows 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. First we need to define predicate encryption.

**Definition 1** (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:*

- $\mathsf{PE.Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pp})$*: Takes as input the security parameter $\lambda$, outputting a secret key $\mathsf{sk}$ and some public parameters $\mathsf{pp}$.*
- $\mathsf{PE.Encrypt}(\mathsf{sk}, x) \to \mathsf{ct}_x$*: Takes as input the secret key $\mathsf{sk}$, a plaintext $x \in \mathcal{M}$, outputting a ciphertext $\mathsf{ct}_x$.*
- $\mathsf{PE.TokGen}(\mathsf{sk}, f) \to \mathsf{tk}_f$*: Takes as input the secret key $\mathsf{sk}$, a predicate $f \in \mathcal{F}$, outputting a token $\mathsf{tk}_f$.*
- $\mathsf{PE.Decrypt}(\mathsf{pp}, \mathsf{tk}_f, \mathsf{ct}_x) \to b$*: Takes as input the public parameters $\mathsf{pp}$, a token $\mathsf{tk}_f$, a ciphertext $\mathsf{ct}_x$ and outputs a bit $b \in \{0,1\}$.*

*We require the scheme to have the following properties:*

- ***Correctness:*** *For any $x \in \mathcal{M}$ let $\mathsf{ct}_x$ denote the random variable resulting from $\mathsf{PE.Encrypt}(\mathsf{sk}, x)$. Similarly, for any $f \in \mathcal{F}$ let $\mathsf{tk}_f$ denote the random variable resulting from $\mathsf{PE.TokGen}(\mathsf{sk}, f)$. $\mathsf{PE}$ is correct if $\mathsf{PE.Decrypt}(\mathsf{pp}, \mathsf{tk}_f, \mathsf{ct}_x) = b$ where, with overwhelming probability, $b = 1$ if $f(x) = 1$ and $b = 0$ if $f(x) = 0$.*
- ***Security of admissible queries:*** *Any PPT adversary $\mathcal{A}$ has only $\mathsf{negl}(\lambda)$ advantage in the following game with challenger $\mathcal{C}$ . Let $r = \mathtt{poly}(\lambda)$ and $s = \mathtt{poly}(\lambda)$. Token and encryption queries must meet the following admissibility requirements:*

$$\forall j \in [1,r],\ \forall i \in [1,s],\ \mathsf{PE.Decrypt}(\mathsf{pp}, \mathsf{tk}_j^{(0)}, \mathsf{ct}_i^{(0)}) = \mathsf{PE.Decrypt}(\mathsf{pp}, \mathsf{tk}_j^{(1)}, \mathsf{ct}_i^{(1)})$$

  *$\mathsf{Exp}_{IND}^{\mathsf{PE}}$ is defined as:*

  1. *$\mathcal{C}$ draws $\beta \xleftarrow{\$} \{0,1\}$, computes $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{PE.Setup}(1^\lambda)$ and sends $\mathsf{pp}$ to $\mathcal{A}$.*
  2. *For $1 \leq i \leq s$, $\mathcal{A}$ chooses two plaintexts $x_i^{(0)}, x_i^{(1)} \in \mathcal{M}$. For $1 \leq j \leq r$, $\mathcal{A}$ chooses two predicates $f_j^{(0)}, f_j^{(1)} \in \mathcal{F}$. We denote $R = \left(x_1^{(0)}, x_1^{(1)}\right), \cdots, \left(x_s^{(0)}, x_s^{(1)}\right)$ and $S = \left(f_1^{(0)}, f_1^{(1)}\right), \cdots, \left(f_r^{(0)}, f_r^{(1)}\right)$.*
  3. *$\mathcal{A}$ sends the list of token generation queries $R$ and the list of encryption queries $S$ to $\mathcal{C}$. $\mathcal{A}$ immediately loses the game if $R$ and $S$ are not admissible, otherwise it receives back a list of tokens $T^{(\beta)} = \mathsf{tk}_1^{(\beta)}, \cdots, \mathsf{tk}_r^{(\beta)}$ and a list of ciphertexts $C^{(\beta)} = \mathsf{ct}_1^{(\beta)}, \cdots, \mathsf{ct}_s^{(\beta)}$ such that $ct_i^{(\beta)} \leftarrow \mathsf{PE.Encrypt}\left(\mathsf{sk}, x_i^{(\beta)}\right)$ and $tk_j^{(\beta)} \leftarrow \mathsf{PE.TokGen}\left(\mathsf{sk}, f_j^{(\beta)}\right)$, with $i \in \{1, \cdots, r\}$ and $j \in \{1, \cdots, s\}$.*

*4. $\mathcal{A}$ returns $\beta' \in \{0,1\}$ and her* advantage *in the game is defined to be*

$$\mathsf{Adv}^{Exp^{PE}_{IND}}_{\mathcal{A}}(\lambda) = \left| \Pr[\mathcal{A}(1^\lambda, T^{(0)}, C^{(0)}) = 1] - \Pr[\mathcal{A}(1^\lambda, T^{(1)}, C^{(1)}) = 1] \right|$$

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 $\mathcal{F} = \{f_y | y \in \mathbb{Z}_q^n\}$ 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.

# 3 Leakage Attacks and Biometric Data

Searchable encryption achieves acceptable performance by allowing the attacker to glean information about the queries being asked and records returned. Formally called *leakage*, the attacker gathers information while observing queries and responses, such as the number of records returned as part of a query. 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. Fuller et al. [FVY+17] noted this type of attack has difficulty over uniform distributions.

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 attern 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. See Definition 4.

Almost all searchable encryption schemes leak the access pattern and these attacks have higher efficacy than those relying on response length or query equality. When considering the leakage profile of a biometric searchable encryption scheme, we have to consider the leakage profile the scheme would have in general, but more importantly, what leakage profile *it actually has based on the data distribution.* Leakage *explicitly* depends on the distribution of records and queries.

## 3.1 Biometrics Statistical Properties

Biometrics represent a unique target for encrypted databases as their statistical properties are stable and well-understood. In this section, we provide a brief overview of how irises are captured and the statistical properties of the iris. The majority of this discussion applies to other biometrics such as fingerprints and face as well.

Daugman [Dau05, Dau09] introduced the seminal iris processing pipeline. This pipeline assumes a near infrared camera. Note that iris images in near infrared are believed to be independent from the visible light pattern and that the iris is epigenetic, irises of identical twins are believed to be independent [Dau09, HBF10]. Traditional iris recognition consists of three primary phases:

**Segmentation** This 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** This 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. It is designed to deal with pupil dilation and changes in the angle of image collection.

**Feature Extraction** In the final stage the fixed size rectangular array is used for feature extraction. In Daugman's original work this consisted of convolving small areas of the rectangle with a fixed 2D wavelet. The resulting sign of the convolution was used to populate a binary vector. (The convolution actually produces an imaginary number, frequently the sign of either the real or imaginary component is used as the resulting set of features.) Each bit of the resulting feature vector can be seen as characterizing a small region of the iris. Different bits correspond to different rectangular regions (which in turn correspond to small polar regions.

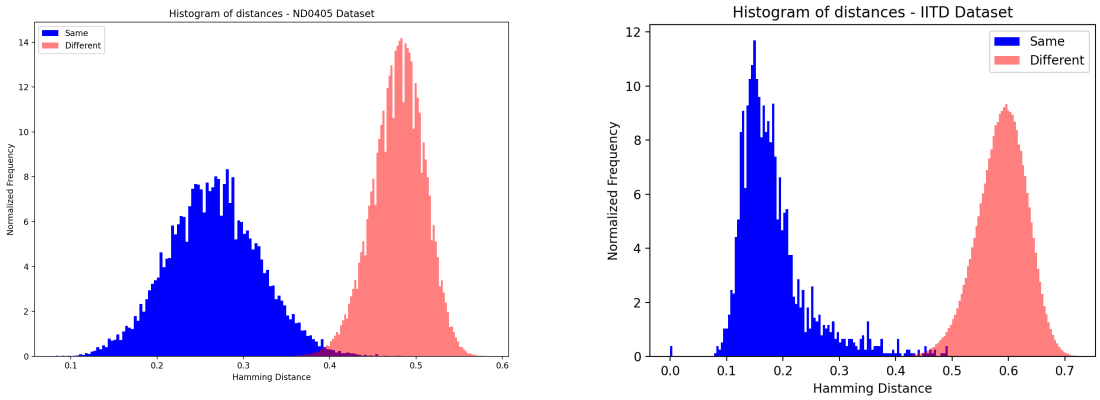
There are two important aspects to consider when discussing iris biometrics. The first is how similar two images of a single iris are over time. There are many reasons for noise between collections of the same iris, these include changes in lighting, collection angle, collection distance, physiological changes, and sensor noise. As described above, when one wishes to match a biometric $r^*$ against a database one considers matches as the set $\{r_i | \mathcal{D}(r_i, r^*) \leq t\}$ for some metric $\mathcal{D}$ and distance parameter $t$. In most iris systems, the distance metric is Hamming distance. In fingerprint systems, the metric is usually set difference, in facial recognition, the metric is usually the L2 distance. Selecting a small $t$ accepts images with less noise and a large $t$ corresponds to accepting images with a lot of noise. This corresponds to the *true positive rate* of the resulting system.

However, $t$ is not a free parameter because of the second important statistical aspect: how similar are the resulting vectors of two different irises? Ideally, the Hamming distance of two different irises would be half the length of the resulting vector meaning that irises are well spread in the high dimensional vector space. Naturally, it is unreasonable to expect all pairs of irises to have distance of exactly half the length of the resulting vector. For biometric search, the relevant question is for a fixed $t$ how frequently will a different iris return a match. This corresponds to the *false positive rate* of the system.

To understand the tradeoff between true positive rate and false positive rate we need to consider a reference dataset and a particular feature extractor. There are many iris datasets collected across a variety of conditions. We consider 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 (both left and right eyes) with 64964 images in total. Additionally, we consider the IITD dataset [KP10]. The IITD dataset consists of 224 persons and 2240 images. The IITD dataset is considered "easier" than the ND-0405 dataset because images are collected in more controlled environments leading to less noise and variation between images.

For the feature extractor, we use the recent pipeline called ThirdEye [AF18, AF19] which is publicly available [Ahm20]. Resulting histograms (when training and testing according to the description in [AF19]) for the ND 0405 and IITD datasets are in Figure 1. In these figures, blue represents comparisons between images of the same iris while red represents comparisons between images of different irises. Note that these two datasets produce different statistics. The x-axis scale is different on the figures. Furthermore, note that in both cases there is overlap between the red and blue histogram indicating that there is a tradeoff between false positive and true positive rates. That is, there is no distance $t$ such that all future readings of the same iris will be marked as the same and that all images of a different iris will be considered different. However, one call still achieve a high true accept rate with a 0% false positive rate.

**Eliminating Useful Leakage** Proximity search considered in a vacuum has leakage similar to any nearest neighbor scheme. Queries will reveal any records that are within distance $t$ of the queried point, records that will be returned in multiple queries can potentially be correlated across those queries, and the number of records returned will be leaked. Consider a system where $t$ was large enough that several records were returned, over time this would be enough information to build a co-occurrence matrix and execute a version of the attack from [CGPR15]. However, if we set $t$ in our system such that at most *1 record* is returned, he co-occurrence matrix will always be all zeroes. We are never given enough information to cross-correlate across different queries in this case and the existing attacks fail. Furthermore, for response length leakage




(a) ND 0405 Histogram

(b) IITD Histogram

Figure 1: Hamming distance distribution for images from the same iris in blue, and different irises in red. Histograms are produced using recent state of the art deep neural network based feature extractor ThirdEye [AF19] (Source at [Ahm20]). Resulting histograms for the ND 0405 and IITD datasets respectively. Comparisons between images of the same iris are in Blue, comparisons between different irises are in Red.

attacks, all queries would either return zero records or one record; this leaves the attack no information to leverage. Essentially, all queries are alike. This leaves query equality leakage as the one remaining leakage type in our scheme. In the case that this is not a leakage pattern that the use case can support, there are compilers that can suppress it [KMO18]. As discussed above, our MPIPE construction has additional leakage, for stored values $r_i, r_j$ and query $r^*$ it reveals

$$\mathcal{D}(r_i, r^*) \stackrel{?}{=} \mathcal{D}(r_j, r^*).$$

We now investigate the largest possible settings of $t$ where at most one record will be returned by each query. This has dual benefits: 1) making access pattern leakage unusable and 2) making MPIPE have no addition leakage. There are two possible settings of distance that are relevant for biometric search. We now consider safe settings of $t$ for two assumptions on the query distribution. In all cases, we assume at most one record corresponding to a particular biometric (an individual iris) is present in the database.

**Biometric Query** If the query value $r^*$ is drawn from the biometric distribution, for a particular distance threshold $t$, the false accept probability of the system corresponds to the probability that there will be additional leakage.

**Arbitrary Query** If the query value $r^*$ is assumed to be arbitrary, then let $t_{min}$ be the minimum observed distance between two different biometrics, one must allow searching at distance at most $t = t_{min}/2$ to ensure that $r^*$ does not match multiple records.

Table 1 summarizes the distances required for access pattern leakages and the corresponding true accept rates for the ND 0405 and IITD datasets. As mentioned above, these values vary based on many environmental conditions. For arbitrary queries we report on true accept rate when setting $t = t_{min}/2$. For biometric queries, we report true accept rate when setting $t$ to be $t_{min}$ which is 0% of leakage, as well as a 1% and 10% probability of leakage. That is, the distance parameter that corresponds to a 1% (resp. 10%) chance of a query matching a *different* biometric.

# 4 From MPIPE to PSE

## 4.1 Multi-point inner product encryption

Secret key inner product predicate encryption [SSW09] allows 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

| Dataset | $t_{min}$ | Biometric Queries TAR at prob. leakage .10 | .01 | .00 | Arbitrary Queries |
|---|---|---|---|---|---|
| ND-0405 | .302 | .887 | .771 | .756 | .014 |
| IITD | .266 | .945 | .909 | .905 | .164 |

Table 1: Summary of true accept rates for queries drawn from *Same* distribution for noise tolerance parameters. The value $t_{min}$ is the smallest recorded distance between different biometrics in the dataset. For biometric queries, we report the true accept rate (TAR) when allowing for a $0\%, 1\%$ and $10\%$ chance of a query matching a record other than the intended biometric.

equality testing for non-zero values. To implement proximity searchable encryption, we need to extend this notion to one that allows to test if the inner product is in a range of values. We introduce a new primitive, *multi-point inner product predicate encryption (MPIPE)*, that achieves this goal.

**Definition 2** (Multi-point inner product predicate encryption)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter. Let $n, q, t \in \mathbb{N}$ such that $n = \mathtt{poly}(\lambda)$, $t = \mathtt{poly}(\lambda)$ and $q$ is a prime. Let $\mathcal{F} = \{f_{\vec{y},D} : \mathbb{Z}_q^n \to \{0,1\} \mid \vec{y} \in \mathbb{Z}_q^n, D \in \mathbb{Z}_q^t\}$ be the set of predicates such that for all vectors $\vec{x} \in \mathbb{Z}_q^n$, $f_{\vec{y},D}(\vec{x}) = 1$ when $\langle \vec{x}, \vec{y} \rangle \in D$ and $f_{\vec{y},D}(\vec{x}) = 0$ otherwise. We define $\mathsf{MPIPE} = (\mathsf{MPIPE.Setup}, \mathsf{MPIPE.Encrypt}, \mathsf{MPIPE.TokGen}, \mathsf{MPIPE.Decrypt})$, a multi-point inner product predicate encryption scheme, as a $\mathsf{PE}$ over $\mathbb{Z}_q^n$ and restricted to the class of predicates $\mathcal{F}$. We require the scheme to have the additional admissibility property in the security game:*

- ***Admissibility:*** *Let $i \in \{1, \cdots, r\}$, $\ell \in \{1, \cdots, t\}$, $j \in \{1, \cdots, s\}$, and consider a token generation query $\left(\vec{y}_i^{\,(0)}, D_i^{(0)}, \vec{y}_i^{\,(1)}, D_i^{(1)}\right)$:*
  - *Let $J_i^{(\beta)} = \left\{ d_\ell^{(\beta)} \in D_i^{(\beta)} \mid \exists\ \vec{x}_j^{\,(\beta)} \text{ such that } \langle \vec{x}_j^{\,(\beta)}, \vec{y}_i^{\,(\beta)} \rangle = d_\ell^{(\beta)} \right\}$ then $|J_i^{(0)}| = |J_i^{(1)}|$.*
  - *$\forall \vec{y}_i^{\,(\beta)}$ there is at most one $\vec{x}_j^{\,(\beta)}$ such that $\langle \vec{x}_j^{\,(\beta)}, \vec{y}_i^{\,(\beta)} \rangle \in D_i^{(\beta)}$.*

Let us now present a construction for $\mathsf{MPIPE}$ built upon $\mathsf{IPE}$.

**Construction 1** (MPIPE)**.** *Fix the security parameter $\lambda \in \mathbb{N}$ and let $n \in \mathbb{N}^+$ be a parameter. Let $\mathsf{IPE} = (\mathsf{IPE.Setup}, \mathsf{IPE.TokGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ be an IPE scheme over $\mathbb{Z}_q^{n+1}$ and let $\pi : \mathbb{Z}_q^t \to \mathbb{Z}_q^t$ be a random permutation. Let $\vec{x}$ and $\vec{y}$ be two vectors in $\mathbb{Z}_q^n$ and $D \in \mathbb{Z}_q^t$ be the set of desired values for their inner product.*

- $\mathsf{MPIPE.Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pp})$*: Takes as input $1^\lambda$ and calls $(\mathsf{sk}_{\mathsf{IPE}}, \mathsf{pp}_{\mathsf{IPE}}) \leftarrow \mathsf{IPE.Setup}(1^\lambda)$. Outputs $\mathsf{sk} = \mathsf{sk}_{\mathsf{IPE}}$ and $\mathsf{pp} = \mathsf{pp}_{\mathsf{IPE}}$.*
- $\mathsf{MPIPE.TokGen}(\mathsf{sk}, \vec{y}, D) \to \mathsf{tk}_{\vec{y},D}$*: Takes as inputs the secret key $\mathsf{sk}$, the vector $\vec{y}$ and the set of values $D$. Computes $D^* = \pi(D) = \{d_1, \cdots, d_t\}$. For $1 \le i \le t$, calls $\mathsf{tk}_i \leftarrow \mathsf{IPE.TokGen}(\mathsf{sk}_{\mathsf{IPE}}, \vec{y} \mid\mid d_i)$. Outputs $\mathsf{tk}_{\vec{y},D} = (\mathsf{tk}_1, \cdots, \mathsf{tk}_t)$.*
- $\mathsf{MPIPE.Encrypt}(\mathsf{sk}, \vec{x}) \to \mathsf{ct}_{\vec{x}}$*: Takes as inputs $\mathsf{sk}$ and the vector $\vec{x}$. Calls $\mathsf{ct} \leftarrow \mathsf{IPE.Encrypt}(\mathsf{sk}_{\mathsf{IPE}}, \vec{x} \mid\mid \text{-1})$. Outputs $\mathsf{ct}_{\vec{x}} = \mathsf{ct}$.*
- $\mathsf{MPIPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_{\vec{x}}) \to b$*: Takes as inputs the public parameters $\mathsf{pp}$, the token $\mathsf{tk}_{\vec{y},D}$ and the ciphertext $\mathsf{ct}_{\vec{x}}$. For each $\mathsf{tk}_i \in \mathsf{tk}_{\vec{y},D}$, calls $b_i \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}_{\mathsf{IPE}}, \mathsf{tk}_i, \mathsf{ct}_{\vec{x}})$. If $b_i = 1$ outputs $b = 1$, otherwise pass to $\mathsf{tk}_{i+1}$. If $b_i = 0$ for every $\mathsf{tk}_i \in \mathsf{tk}_{\vec{y},D}$, outputs $b = 0$.*

**Note on Admissibility** Query admissibility in $\mathsf{MPIPE}$ is a strict subset of admissibility in $\mathsf{IPE}$. Consider for example some

$$\vec{y}^{\,(0)}, \vec{y}^{\,(1)}, \vec{x}^{\,(0)}, \vec{x}^{\,(1)} \in \mathbb{Z}_q^n$$

and $D^{(0)} = \{d_1^{(0)}, d_2^{(0)}\}$, $D^{(1)} = \{d_1^{(1)}, d_2^{(1)}\}$ such that $\langle \vec{x}^{\,(0)}, \vec{y}^{\,(0)} \rangle = d_1^{(0)}$ and $\langle \vec{x}^{\,(1)}, \vec{y}^{\,(1)} \rangle = d_2^{(1)}$. Then for all $d_\ell^{(0)} \in D^{(0)}$, $d_\ell^{(1)} \in D^{(1)}$:

$$\begin{aligned}&(\langle \vec{x}^{\,(0)}, \vec{y}^{\,(0)} \rangle \stackrel{?}{\in} D^{(0)}) = (\langle \vec{x}^{\,(1)}, \vec{y}^{\,(1)} \rangle \stackrel{?}{\in} D^{(1)}) \\ &\implies (\langle \vec{x}^{\,(0)}, \vec{y}^{\,(0)} \rangle \stackrel{?}{=} d_1^{(0)}) = (\langle \vec{x}^{\,(1)}, \vec{y}^{\,(1)} \rangle \stackrel{?}{=} d_1^{(1)})\end{aligned}$$

Moreover, notice that in this construction, $\forall i, j \in \{1, \cdots, s\}$ and if $\langle \vec{x}_i^{\,(\beta)}, \vec{y}^{\,(\beta)} \rangle, \langle \vec{x}_j^{\,(\beta)}, \vec{y}^{\,(\beta)} \rangle \in D^{(\beta)}$, the following equality predicate is leaked:

$$\langle \vec{x}_i^{\,(\beta)}, \vec{y}^{\,(\beta)} \rangle = \langle \vec{x}_j^{\,(\beta)}, \vec{y}^{\,(\beta)} \rangle$$

**Theorem 1** (MPIPE main theorem)**.** *Let* $\mathsf{IPE} = (\mathsf{IPE.Setup}, \mathsf{IPE.TokGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ *be a function hiding inner product predicate encryption scheme over* $\mathbb{Z}_q^{n+1}$*. Then there exists* $\mathsf{MPIPE} = (\mathsf{MPIPE.Setup}, \mathsf{MPIPE.TokGen}, \mathsf{MPIPE.Encrypt}, \mathsf{MPIPE.Decrypt})$*, an function hiding multi-point inner product predicate encryption scheme over* $\mathbb{Z}_q^n$*, such that for any PPT adversary* $\mathcal{A}_{\mathsf{MPIPE}}$ *for* $\mathsf{Exp}_{IND}^{\mathsf{MPIPE}}$*, there exists a PPT adversary* $\mathcal{A}_{\mathsf{IPE}}$ *for* $\mathsf{Exp}_{IND}^{\mathsf{IPE}}$*, such that for any security parameter* $\lambda \in \mathbb{N}$*,*

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

*Proof.* The correctness of the scheme follows from the correctness of the underlying $\mathsf{IPE}$ scheme. Assume $\langle \vec{x}, \vec{y} \rangle \in D$ then there exists a unique $\mathsf{tk}_i \in \mathsf{tk}_{\vec{y},D}$, such that $b \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_i, \mathsf{ct}_{\vec{x}})$ and $b = 1$ with overwhelming probability by the correctness of the $\mathsf{IPE}$ scheme. Now assume $\langle \vec{x}, \vec{y} \rangle \notin D$, then for any $\mathsf{tk}_i \in \mathsf{tk}_{\vec{y},D}$, such that $b \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_i, \mathsf{ct}_{\vec{x}})$, we have $b = 1$ with negligible probability. Then considering the worst case where either $\langle \vec{x}, \vec{y} \rangle$ is equal to the value in $D$ corresponding to the last token or no value in $D$:

$$\begin{aligned}&\Pr[\mathsf{MPIPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_{\vec{x}}) = (\langle \vec{x}, \vec{y} \rangle \stackrel{?}{\in} D)] \\ &\geq 1 - t \times \Pr[\mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_i, \mathsf{ct}_{\vec{x}}) \neq (\langle \vec{x}, \vec{y} \rangle \stackrel{?}{=} d_i)] \\ &\geq 1 - t \times \mathsf{negl}(\lambda).\end{aligned}$$

Let $\mathcal{A}_{\mathsf{MPIPE}}$ be a PPT adversary for experiment $\mathsf{Exp}_{IND}^{\mathsf{MPIPE}}$. Let $r = \mathtt{poly}(\lambda)$ be the number of token queries, $s = \mathtt{poly}(\lambda)$ be the number of encryption queries and $\beta \stackrel{\$}{\leftarrow} \{0, 1\}$. Let $\mathcal{C}_{\mathsf{IPE}}$ be the challenger for the $\mathsf{IPE}$ experiment we build a PPT adversary $\mathcal{A}_{\mathsf{IPE}}$ for the experiment $\mathsf{Exp}_{IND}^{\mathsf{IPE}}$ as follows:

1. $\mathcal{A}_{\mathsf{IPE}}$ receives $\mathsf{pp}$ from $\mathcal{C}_{\mathsf{IPE}}$ and forwards it to $\mathcal{A}_{\mathsf{MPIPE}}$.
2. $\mathcal{A}_{\mathsf{IPE}}$ receives $r$ key generation queries and $s$ encryption queries from $\mathcal{A}_{\mathsf{MPIPE}}$. For $1 \leq i \leq r$, $\mathcal{A}_{\mathsf{IPE}}$ receives the non-zero vectors $\vec{y}_i^{\,(0)}, \vec{y}_i^{\,(1)} \in \mathbb{Z}_q^n$ and $D_i^{(0)}, D_i^{(0)} \in \mathbb{Z}_q^t$. For $1 \leq j \leq s$, $\mathcal{A}_{\mathsf{IPE}}$ receives the non-zero vectors $\vec{x}_j^{\,(0)}, \vec{x}_j^{\,(1)} \in \mathbb{Z}_q^n$.
3. For $1 \leq i \leq r$:
   (a) $\mathcal{A}_{\mathsf{IPE}}$ creates $D_i^{(0)*}$ by reordering the elements in $D_i^{(0)}$ such that for all $\ell \in \{1, \cdots, t\}$ and $d_\ell^{(0)} \in D_i^{(0)*}, d_\ell^{(1)} \in D_i^{(1)}$ we have
   $$\big(\langle \vec{x}_j^{\,(0)}, \vec{y}_i^{\,(0)} \rangle \stackrel{?}{=} d_\ell^{(0)}\big) = \big(\langle \vec{x}_j^{\,(1)}, \vec{y}_i^{\,(1)} \rangle \stackrel{?}{=} d_\ell^{(1)}\big).$$
   ($\mathcal{A}_{\mathsf{IPE}}$ can always find a permutation to make this last condition true as queries that are admissible to $\mathsf{MPIPE}$ are also admissible to $\mathsf{IPE}$.)
   (b) $\mathcal{A}_{\mathsf{IPE}}$ samples a random permutation $\psi_i : \mathbb{Z}_q^t \to \mathbb{Z}_q^t$.
   (c) For $1 \leq \ell \leq t$, $\mathcal{A}_{\mathsf{IPE}}$ creates $\vec{y}_i^{\,(\beta)} || \; d_\ell^{(\beta)}$ with $d_\ell^{(\beta)} \in D_i^{(\beta)*}$ and $\beta \in \{0, 1\}$, then computes
   $$R_i^{(\beta)} = \psi_i\Big( \vec{y}_i^{\,(\beta)} || \; d_1^{(\beta)}, \cdots, \vec{y}_i^{\,(\beta)} || \; d_t^{(\beta)} \Big)$$
   and sets $R_i = (R_i^{(0)}, R_i^{(1)})$.

4. For $1 \leq j \leq s$, $\mathcal{A}_{\mathsf{IPE}}$ sets $S_j = (\vec{x}_j^{(0)} || \text{ -1}, \ \vec{x}_j^{(1)} || \text{ -1})$.

5. $\mathcal{A}_{\mathsf{IPE}}$ sends the token generation queries $R_1, \cdots, R_r$ and the encryption queries $S_1, \cdots, S_s$ to $\mathcal{C}_{\mathsf{IPE}}$ and receives back a set of tokens $T^{(\beta)} = \mathsf{tk}_{1,1}^{(\beta)}, \cdots, \mathsf{tk}_{r,t}^{(\beta)}$ and a set of encrypted vectors $C^{(\beta)} = \mathsf{ct}_1^{(\beta)}, \cdots, \mathsf{ct}_s^{(\beta)}$ such that $\mathsf{tk}_{i,\ell}^{(\beta)} \leftarrow \mathsf{IPE.TokGen}(\mathsf{sk}, \vec{y}_i^{(\beta)} \ || \ d_\ell^{(\beta)})$ and $\mathsf{ct}_j^{(\beta)} \leftarrow \mathsf{IPE.Encrypt}(\mathsf{sk}, \vec{x}_j^{(\beta)} \ || \text{ -1})$. $\mathcal{A}_{\mathsf{IPE}}$ forwards $T^{(\beta)}$ and $C^{(\beta)}$ to $\mathcal{A}_{\mathsf{MPIPE}}$.

6. $\mathcal{A}_{\mathsf{IPE}}$ receives $\beta' \in \{0, 1\}$ from $\mathcal{A}_{\mathsf{MPIPE}}$ and returns it.

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

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

□

**Construction 2** (MPIPE variant)**.** *Fix the security parameter $\lambda \in \mathbb{N}$. Let $\mathsf{IPE} = (\mathsf{IPE.Setup}, \mathsf{IPE.TokGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ be an IPE scheme over $\mathbb{Z}_q^{n+t}$ and let $\pi, \varphi : \mathbb{Z}_q^t \to \mathbb{Z}_q^t$ be two random permutations. Let $\vec{x}$ and $\vec{y}$ be two vectors in $\mathbb{Z}_q^n$ and $D \in \mathbb{Z}_q^t$ be the set of desired values for their inner product.*

- $\mathsf{MPIPE.Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pp})$*: Takes as input $1^\lambda$ and calls $(\mathsf{sk}_{IPE}, \mathsf{pp}_{IPE}) \leftarrow \mathsf{IPE.Setup}(1^\lambda)$. Outputs $\mathsf{sk} = \mathsf{sk}_{IPE}$ and $\mathsf{pp} = \mathsf{pp}_{IPE}$.*

- $\mathsf{MPIPE.TokGen}(\mathsf{sk}, \vec{y}, D) \to \mathsf{tk}_{\vec{y},D}$*: Takes as inputs the secret key $\mathsf{sk}$, a vector $\vec{y}$ and a set of values $D$. Computes $D^* = \pi(D) = (d_1, \cdots, d_t)$ and sets $\vec{y}^* = \vec{y} \ || \ D^*$. Calls $\mathsf{tk} \leftarrow \mathsf{IPE.TokGen}(\mathsf{sk}_{IPE}, \vec{y}^*)$. Outputs $\mathsf{tk}_{\vec{v},D} = \mathsf{tk}$.*

- $\mathsf{MPIPE.Encrypt}(\mathsf{sk}, \vec{x}) \to \mathsf{ct}_{\vec{x}}$*: Takes as inputs $\mathsf{sk}$ and a vector $\vec{x}$. For $i \in \{1, \cdots, t\}$ creates $\vec{x}_i^* \in \mathbb{Z}_q^{n+t}$ such that*

$$\vec{x}_i^* = (\vec{x} \ || \ \overbrace{0, \cdots, 0}^{i-1}, \text{-1}, \overbrace{0, \cdots, 0}^{t-i}).$$

  *Calls $\mathsf{ct}_i \leftarrow \mathsf{IPE.Encrypt}(\mathsf{sk}_{IPE}, \vec{x}_i^*)$. Outputs $\mathsf{ct}_{\vec{x}} = \varphi(\mathsf{ct}_1, \cdots, \mathsf{ct}_t)$.*

- $\mathsf{MPIPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_{\vec{x}}) \to b$*: Takes as inputs the public parameters $\mathsf{pp}$, the token $\mathsf{tk}_{\vec{y},D}$ and the ciphertext $\mathsf{ct}_{\vec{x}}$. For each $ct_i \in \mathsf{ct}_{\vec{x}}$, calls $b_i \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}_{IPE}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_i)$. If $b_i = 1$ outputs $b = 1$, otherwise pass to $\mathsf{ct}_{i+1}$. If $b_i = 0$ for every $\mathsf{ct}_i \in \mathsf{ct}_{\vec{x}}$, outputs $b = 0$.*

**Correctness** *Let $\mathsf{pp}, \mathsf{sk}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_{\vec{x}}$ be defined as in the above construction and let $\vec{x}, \vec{y} \in \mathbb{Z}_q^n$, $D \in Z_q^t$ and $\vec{x}^*, \vec{y}^* \in \mathbb{Z}_q^{n+t}$. Assume $\langle \vec{x}, \vec{y} \rangle \in D$ then there exists a unique $\mathsf{ct}_i \in \mathsf{ct}_{\vec{x}}$ such that $b_i \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_i)$ and $b_i = 1$ with overwhelming probability. Now assume $\langle \vec{x}, \vec{y} \rangle \notin D$, then for any $\mathsf{ct}_i \in \mathsf{ct}_{\vec{x}}$, $b_i \leftarrow \mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_i)$ and $b_i = 1$ with negligible probability. Then considering the worst case where either $\langle \vec{x}, \vec{y} \rangle$ is equal to the last ciphertext or no ciphertext in $\mathsf{ct}_{\vec{x}}$:*

$$\begin{aligned}
&\Pr[\mathsf{MPIPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_{\vec{x}}) = (\langle \vec{x}, \vec{y} \rangle \stackrel{?}{\in} D)] \\
&\quad \geq 1 - t \times \Pr[\mathsf{IPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_i) \neq (\langle \vec{x}, \vec{y} \rangle \stackrel{?}{=} d_l)] \\
&\quad \geq 1 - t \times \mathsf{negl}(\lambda)
\end{aligned}$$

*and this construction is thus correct.*

**Equality leakage** *Notice that contrary to Construction 1, in this MPIPE variant,* $\forall i, j \in \{1, \cdots, s\}$ *and even if* $\langle \vec{x}_i^{(\beta)}, \vec{y}^{(\beta)} \rangle, \langle \vec{x}_j^{(\beta)}, \vec{y}^{(\beta)} \rangle \in D^{(\beta)}$*, the following equality is not leaked:*

$$\langle \vec{x}_i^{(\beta)}, \vec{y}^{(\beta)} \rangle = \langle \vec{x}_j^{(\beta)}, \vec{y}^{(\beta)} \rangle$$

**Theorem 2.** *Let* IPE *be an IND-secure function hiding point-testing inner product encryption scheme over* $\mathbb{Z}_q^{n+t}$*. Then for* MPIPE.Setup$(1^\lambda)$*,* MPIPE.TokGen$(\mathsf{sk}, \vec{y}, D)$*,* MPIPE.Encrypt$(\mathsf{sk}, \vec{x})$ *and* MPIPE.Decrypt$(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_{\vec{x}})$ *built as described in Construction 2,* MPIPE *is a single token IND-secure function hiding multi-point inner product encryption scheme over* $\mathbb{Z}_q^n$*.*

*Proof.* Let $s = \mathtt{poly}(\lambda)$ be the number of encryption queries. Let $\mathcal{O}_{\mathsf{IPE}}$ be an oracle for the experiment $\mathsf{Exp}_{IND}^{\mathsf{IPE}}$ and $\mathcal{A}_{\mathsf{MPIPE}}$ be a PPT adversary for the experiment $\mathsf{Exp}_{IND}^{\mathsf{MPIPE}}$. Then we can build a PPT adversary $\mathcal{A}_{\mathsf{IPE}}$ for the experiment $\mathsf{Exp}_{IND}^{\mathsf{IPE}}$ as follows:

1. $\mathcal{A}_{\mathsf{IPE}}$ receives $\mathsf{pp}$ from $\mathcal{O}_{\mathsf{IPE}}$ and forwards it to $\mathcal{A}_{\mathsf{MPIPE}}$.
2. $\mathcal{A}_{\mathsf{IPE}}$ receives a token generation query and $s$ encryption queries from $\mathcal{A}_{\mathsf{MPIPE}}$: $\mathcal{A}_{\mathsf{IPE}}$ receives the non-zero vectors $\vec{y}^{(0)}, \vec{y}^{(1)} \in \mathbb{Z}_q^n$ and $D^{(0)}, D^{(1)} \in \mathbb{Z}_q^t$. For $1 \leq j \leq s$, $\mathcal{A}_{\mathsf{IPE}}$ receives the non-zero vectors $\vec{x}_j^{(0)}, \vec{x}_j^{(1)} \in \mathbb{Z}_q^n$.
3. $\mathcal{A}_{\mathsf{IPE}}$ samples a random permutation $\varphi : \mathbb{Z}_q^t \to \mathbb{Z}_q^t$ and creates $\vec{y}^{(0)*} = \vec{y}^{(0)} \mid\mid \varphi\big(D^{(0)}\big)$ and $\vec{y}^{(1)*} = \vec{y}^{(1)} \mid\mid \varphi\big(D^{(1)}\big)$.
4. For $1 \leq j \leq s$:
   (a) $\mathcal{A}_{\mathsf{IPE}}$ samples a random permutation $\psi_j : \mathbb{Z}_q^t \to \mathbb{Z}_q^t$.
   (b) For $1 \leq \ell \leq t$, $\mathcal{A}_{\mathsf{IPE}}$ creates
   $$\vec{x}_{j,\ell}^{(\beta)*} = \big(\vec{x}_j^{(\beta)} \mid\mid \overbrace{0, \cdots, 0}^{\ell-1}, -1, \overbrace{0, \cdots, 0}^{t-\ell}\big).$$
   (c) If $\langle \vec{x}_{j,\ell}^{(1)*}, \vec{y}^{(1)*} \rangle = 0$ and $\langle \vec{x}_{j,k}^{(0)*}, \vec{y}^{(0)*} \rangle = 0$ for some $k \neq \ell$ then $\mathcal{A}_{\mathsf{IPE}}$ swaps $\vec{x}_{j,\ell}^{(0)*}$ with $\vec{x}_{j,k}^{(0)*}$ to ensure that for $\ell = 1, \cdots, t$
   $$\big(\langle \vec{x}_{j,\ell}^{(0)}, \vec{y}^{(0)} \rangle \stackrel{?}{=} 0\big) = \big(\langle \vec{x}_{j,\ell}^{(1)}, \vec{y}^{(1)} \rangle \stackrel{?}{=} 0\big).$$
   (d) $\mathcal{A}_{\mathsf{IPE}}$ computes $S_j^{(0)} = \psi_j(\vec{x}_{j,1}^{(0)*}, \cdots, \vec{x}_{j,t}^{(0)*})$ and $S_j^{(1)} = \psi_j(\vec{x}_{j,1}^{(1)*}, \cdots, \vec{x}_{j,t}^{(1)*})$, and sets $S_j = (S_j^{(0)}, S_j^{(1)})$.
5. $\mathcal{A}_{\mathsf{IPE}}$ sends the token generation query $(\vec{y}^{(0)*}, \vec{y}^{(1)*})$ and the encryption queries $S_1, \cdots, S_s$ to $\mathcal{O}_{\mathsf{IPE}}$ and receives back a token $\mathsf{tk}^{(\beta)}$ and a set of ciphertexts $C^{(\beta)} = \mathsf{ct}_{1,1}^{(\beta)}, \cdots, \mathsf{ct}_{s,t}^{(\beta)}$ such that
   $$\mathsf{tk}^{(\beta)} \leftarrow \mathsf{IPE.TokGen}(\mathsf{sk}, \vec{y}^{(\beta)*}),$$
   $$\mathsf{ct}_{j,\ell}^{(\beta)} \leftarrow \mathsf{IPE.Encrypt}(\mathsf{sk}, \vec{x}_{j,\ell}^{(\beta)*}).$$
   $\mathcal{A}_{\mathsf{IPE}}$ forwards $\mathsf{tk}^{(\beta)}$ and $C^{(\beta)}$ to $\mathcal{A}_{\mathsf{MPIPE}}$.
6. $\mathcal{A}_{\mathsf{IPE}}$ receives $\beta' \in \{0, 1\}$ from $\mathcal{A}_{\mathsf{MPIPE}}$ and returns it.

Since the number of encryption queries sent by $\mathcal{A}_{\mathsf{IPE}}$, $s \times t$, remains polynomial in the security parameter $\lambda$, the advantage of $\mathcal{A}_{\mathsf{MPIPE}}$ is

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

□

## 4.2 Proximity searchable encryption

We now turn to *proximity searchable encryption (PSE)*, a variant of searchable encryption, that supports proximity queries. Specifically, we show that one can use MPIPE to construct PSE for Hamming distance. At a high level, each keyword is encoded as a $\{-1, 1\}$ vector, which in turn is encrypted with MPIPE. Keywords are similarly encoded and tokens generated as part of the underlying MPIPE scheme.

**Definition 3** (History)**.** *Let $W \in \mathcal{W}$ be a list of keywords drawn from space $\mathcal{W}$, let $\mathcal{F}$ be the set of all predicates over $\mathcal{W}$, a q-query history over $\mathcal{W}$ is a tuple $\mathsf{History} = (\mathcal{W}, F)$, with $F = (f_1, \cdots, f_q)$ a list of $q$ predicates, $f_i \in \mathcal{F}$.*

**Definition 4** (Access pattern)**.** *Let $W$ be a list of keywords, the access pattern induced by a q-query history $H = (\mathcal{W}, F)$ is the tuple $\mathsf{AccPatt}(H) = (f_1(W), \cdots, f_q(W))$*

Many searchable encryption schemes also leaks when queries are repeated (*query equality leakage*) but this is not the case with our construction.

**Definition 5** (Proximity Searchable Encryption)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter. Consider a database $\mathcal{DB} = (M_1, \cdots, M_\ell)$ containing $\ell$ documents $M_i$ and a list of keywords $W = (\vec{w}_1, \cdots, \vec{w}_\ell)$, such that $\vec{w}_i \in \mathbb{Z}_q^n$ is the keyword related to $M_i$. Let $\mathcal{F} = \{f_{\vec{y},t} \mid \vec{y} \in \mathbb{Z}_q^n,\ t \in N\}$ be a family of predicates such that, for a keyword $\vec{w} \in \mathbb{Z}_q^n$, $f_{\vec{y},t}(\vec{w}) = 1$ if $d(\vec{w}, \vec{y}) \leq t$, 0 otherwise, where $d(\vec{w}, \vec{y})$ denotes the Hamming distance between $\vec{w}$ and $\vec{y}$. The tuple of algorithms $\mathsf{PSE} = (\mathsf{PSE.Setup}, \mathsf{PSE.BuildIndex}, \mathsf{PSE.Trapdoor}, \mathsf{PSE.Search})$ defines a proximity searchable encryption scheme:*

- $\mathsf{PSE.Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pp})$*: Takes as input $1^\lambda$ and outputs a secret key $\mathsf{sk}$ and public parameters $\mathsf{pp}$.*
- $\mathsf{PSE.BuildIndex}(\mathsf{sk}, W) \to I_{\mathcal{W}}$*: Takes as inputs the secret $\mathsf{sk}$ and the list of keywords $W$. Outputs a searchable encrypted index $I_W$.*
- $\mathsf{PSE.Trapdoor}(\mathsf{sk}, f_{\vec{y},t}) \to \mathsf{tk}_{\vec{y},t}$*: Takes as inputs the secret $\mathsf{sk}$ and a predicate $f_{\vec{y},t} \in \mathcal{F}$. Outputs a query $Q_{\vec{y},t}$.*
- $\mathsf{PSE.Search}(\mathsf{pp}, Q_{\vec{y},t}, I_{\mathcal{W}}) \to J_{\mathcal{W},\vec{y},t}$*: Takes as inputs the public parameters $\mathsf{pp}$, the query $Q_{\vec{y},t}$ and the encrypted index $I_{\mathcal{W}}$. Outputs $J_{\mathcal{W},\vec{y},t} \subseteq \{1, \cdots, \ell\}$.*

*We require the scheme to have the following properties:*

- ***Correctness:*** *Let $\mathsf{PSE} = (\mathsf{PSE.Setup}, \mathsf{PSE.BuildIndex}, \mathsf{PSE.Trapdoor}, \mathsf{PSE.Search})$ be an PSE scheme of security parameter $\lambda \in \mathbb{N}$ and $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{PSE.Setup}(1^\lambda)$. For any $\mathcal{W}$ let $I_{\mathcal{W}}$ denote the random variable resulting from $\mathsf{PSE.BuildIndex}(\mathsf{sk}, \mathcal{W})$. Similarly, for any $f_{\vec{y},t} \in \mathcal{F}$ let $Q_{\vec{y},t}$ denote the random variable resulting from $\mathsf{PSE.Trapdoor}(\mathsf{sk}, f_{\vec{y},t})$. Let $J_{\mathcal{W},\vec{y},t} \subseteq \{1, \cdots, \ell\}$ denote the set of integers such that $i \in J_{\mathcal{W},\vec{y},t}$ if $f_{\vec{y},t}(\vec{w}_i) = 1$,*

  $$J_{\mathcal{W},\vec{y},t} = \{i \mid f_{\vec{y},t}(\vec{w}_i) = 1\}$$

  *PSE is correct if we have:*

  $$\Pr[\mathsf{PSE.Search}(\mathsf{pp}, Q_{\vec{y},t}, I_{\mathcal{W}}) = J_{\mathcal{W},\vec{y},t}] \geq 1 - \mathsf{negl}(\lambda)$$

- ***Security for admissible queries:*** *Any PPT adversary $\mathcal{A}$ has only $\mathsf{negl}(\lambda)$ advantage in the following game $\mathsf{Exp}_{IND}^{\mathsf{PSE}}$ with challenger $\mathcal{C}$. Let $l = \mathtt{poly}(\lambda)$ and $q = \mathtt{poly}(\lambda)$.*

  1. *$\mathcal{C}$ draws $\beta \xleftarrow{\$} \{0,1\}$, computes $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{PSE.Setup}(1^\lambda)$ and sends $\mathsf{pp}$ to $\mathcal{A}$.*
  2. *$\mathcal{A}$ chooses two q-query histories $\mathsf{History}^{(0)}, \mathsf{History}^{(1)}$ such that $\mathsf{History}^{(\beta)} = (W^{(\beta)}, F^{(\beta)})$, $\beta \in \{0,1\}$. $\mathcal{A}$ sends $\mathsf{History}^{(0)}$ and $\mathsf{History}^{(1)}$ to $\mathcal{C}$ and immediately loses the game if*

     $$\mathsf{AccPatt}(\mathsf{History}^{(0)}) \neq \mathsf{AccPatt}(\mathsf{History}^{(1)})$$

     *or if there exist $f_j \in F$ and $\vec{x}_i \in \mathcal{W}$ such that $f_j(\vec{w}_i) > 1$. Otherwise, it receives back an encrypted index $I^{(\beta)}$ and a list of trapdoors $Q^{(\beta)}$.*

*3. $\mathcal{A}$ outputs $\beta' \in \{0,1\}$ and her advantage in the game is defined to be:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Exp}_{IND}^{\mathsf{PSE}}}(\lambda) = \Big| \Pr[\mathcal{A}(1^\lambda, I^{(0)}, Q^{(0)}) = 1] - \Pr[\mathcal{A}(1^\lambda, I^{(1)}, Q^{(1)}) = 1] \Big|$$

*Then a proximity searchable encryption scheme PSE is said to be secure if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Exp}_{IND}^{\mathsf{PSE}}}$ is negligible in $\lambda$.*

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. Using the previously defined MPIPE primitive, we can then build a PSE scheme for the Hamming distance.

**Construction 3** (Proximity Searchable Encryption)**.** *Fix the security parameter $\lambda \in \mathbb{N}$. Let $\mathsf{MPIPE} = (\mathsf{MPIPE.Setup}, \mathsf{MPIPE.TokGen}, \mathsf{MPIPE.Encrypt}, \mathsf{MPIPE.Decrypt})$ be an MPIPE scheme. Let $\vec{w}_i \in \mathbb{Z}_q^n$ and $\mathcal{W} = \vec{w}_1, \cdots, \vec{w}_\ell$ be the list of keywords. Let $\mathcal{F}$ be the set of all predicates such that for any $\vec{w}_i \in \mathcal{W}$, $f_{\vec{y},t}(\vec{w}_i) = 1$ if the Hamming distance between $\vec{w}_i$ and the query vector $\vec{y} \in \mathbb{Z}_q^n$ is less or equal to some chosen threshold $t \in \mathbb{Z}_q$, $f_{\vec{y},t}(\vec{w}_i) = 0$ otherwise. We can then construct a proximity searchable encryption scheme for the Hamming distance as follows:*

- $\mathsf{PSE.Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pp})$*: Takes as input $1^\lambda$ and outputs $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{MPIPE.Setup}(1^\lambda)$.*
- $\mathsf{PSE.BuildIndex}(\mathsf{sk}, \mathcal{W}) \to I_{\mathcal{W}}$*: Takes as inputs the secret key $\mathsf{sk}$ and the keywords list $\mathcal{W}$. For each keyword $\vec{w}_i \in \mathcal{W}$, $i \in \{1, \cdots, \ell\}$, encodes it as $\vec{w}_i^* \in \{-1,1\}^n$ and computes $\mathsf{ct}_i \leftarrow \mathsf{MPIPE.Encrypt}(\mathsf{sk}, \vec{w}_i^*)$. Outputs $I_{\mathcal{W}} = \mathsf{ct}_1, \cdots, \mathsf{ct}_\ell$.*
- $\mathsf{PSE.Trapdoor}(\mathsf{sk}, f_{\vec{y},t}) \to Q_{\vec{y},t}$*: Takes as inputs $\mathsf{sk}$ and a predicate $f_{\vec{y},t}$. Creates $D = \{d_1, \cdots, d_t\}$ such that $d_j = n - 2j$ with $0 \le j \le t$. Encodes $\vec{y}$ as $\vec{y}^* \in \{-1,1\}^n$ and computes $\mathsf{tk} \leftarrow \mathsf{MPIPE.TokGen}(\mathsf{sk}, \vec{y}^*, D)$. Outputs $Q_{\vec{y},t} = \mathsf{tk}$.*
- $\mathsf{PSE.Search}(\mathsf{pp}, Q_{\vec{y},t}, I_{\mathcal{W}}) \to J_{\mathcal{W},\vec{y},t}$*: Creates an empty set $J_{\mathcal{W},\vec{y},t}$. For each $\mathsf{ct}_i \in I_{\mathcal{W}}$, computes $b \leftarrow \mathsf{MPIPE.Decrypt}(\mathsf{pp}, \mathsf{tk}, \mathsf{ct}_i)$ and adds $i$ to $J_{\mathcal{W},\vec{y},t}$ if $b = 1$. Outputs $J_{\mathcal{W},\vec{y},t}$.*

**Note on Admissibility** Query admissibility in PSE is a strict subset of admissibility in MPIPE. For example, for all $i \in \{1, \cdots, \ell\}$, $j \in \{1, \cdots, q\}$ we have

$$\begin{aligned} \tau(H^{(0)}) = \tau(H^{(1)}) &\implies f_j^{(0)}(\vec{w}_i{}^{(0)}) = f_j^{(1)}(w_i{}^{(1)}) \\ &\implies |J_j^{(0)}| = |J_j^{(1)}| \end{aligned}$$

$$\text{for } J_j^{(\beta)} = \{d_k^{(\beta)} \in D_j^{(\beta)} \mid \exists\ \vec{w}_i{}^{(\beta)} \text{ such that } \langle \vec{y}_j{}^{(\beta)}, \vec{w}_i{}^{(\beta)} \rangle = d_k^{(\beta)}\}$$

Notice that this restriction naturally holds for databases of biometric readings for which we want the distance between any two pair of records to be greater than twice the distance threshold defined for the search $\forall\ (\vec{w}_i, \vec{w}_j) \in \mathcal{W},\ d(\vec{w}_i, \vec{w}_j) > 2t$.

Let us now introduce the second main theorem stating that PSE can be built from IPE.

**Theorem 3** (PSE main theorem)**.** *Let $\mathsf{IPE} = (\mathsf{IPE.Setup}, \mathsf{IPE.TokGen}, \mathsf{IPE.Encrypt}, \mathsf{IPE.Decrypt})$ be an IND-secure function hiding inner product predicate encryption scheme over $\mathbb{Z}_q^n$. 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.* Theorem 1 shows that IPE yields MPIPE and Lemma 1 proves that PSE can be built from MPIPE. Thus, IPE yields PSE. □

**Lemma 1.** *Let MPIPE be an IND-secure function hiding multi-point inner product encryption scheme over $\mathbb{Z}_q^n$. Then there exists $\mathsf{PSE} = (\mathsf{PSE.Setup}, \mathsf{PSE.BuildIndex}, \mathsf{PSE.Trapdoor}, \mathsf{PSE.Search})$ an IND-secure proximity searchable encryption scheme for the Hamming distance, over $\mathbb{Z}_q^n$.*

*Proof.* The correctness of the scheme follows from the correctness of the underlying $\mathsf{MPIPE}$ scheme. Assume there exists $\vec{w}_i \in \mathcal{W}$ such that $f_{\vec{y},t}(\vec{w}_i) = 1$, that is $d(\vec{y}, \vec{w}_i) \leq t$ with $d(\vec{y}, \vec{w}_i)$ the Hamming distance between vectors $\vec{y}$ and $\vec{w}_i$. For $b$ is sampled from $\mathsf{MPIPE.Decrypt}(\mathsf{pp}, \mathsf{tk}, \mathsf{ct}_i)$ and $b = 1$ with overwhelming probability. Now assume that for all $\vec{w}_i$ in $\mathcal{W}$ we have $f_{\vec{y},t}(\vec{w}_i) = 0$, then $b \leftarrow \mathsf{MPIPE.Decrypt}(\mathsf{pp}, \mathsf{tk}, \mathsf{ct}_i)$ and $b = 0$ with high probability. We thus have

$$\begin{aligned}
&\Pr[\mathsf{PSE.Search}(\mathsf{pp}, Q_{\vec{y},t}, I_{\mathcal{W}}) = J_{\mathcal{W},\vec{y},t}] \\
&\quad \geq 1 - t \times \Pr[\mathsf{MPIPE.Decrypt}(\mathsf{pp}, \mathsf{tk}_{\vec{y},D}, \mathsf{ct}_{\vec{w}}) \neq (\langle \vec{w}, \vec{y} \rangle \stackrel{?}{\in} D)] \\
&\quad \geq 1 - t \times \mathsf{negl}(\lambda)
\end{aligned}$$

and this construction is thus correct.

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 $\mathcal{C}_{\mathsf{MPIPE}}$ be an challenger for $\mathsf{Exp}^{\mathsf{MPIPE}}_{IND}$. Then there exists a PPT adversary $\mathcal{A}_{\mathsf{MPIPE}}$ for the experiment $\mathsf{Exp}^{\mathsf{MPIPE}}_{IND}$ which works as follows:

1. $\mathcal{A}_{\mathsf{MPIPE}}$ receives $\mathsf{pp}$ from $\mathcal{C}_{\mathsf{MPIPE}}$ and forwards it to $\mathcal{A}_{\mathsf{PSE}}$.
2. $\mathcal{A}_{\mathsf{MPIPE}}$ receives two q-query histories $H^{(0)}, H^{(1)}$ from $\mathcal{A}_{\mathsf{PSE}}$. ($\mathcal{A}_{\mathsf{MPIPE}}$ can always build admissible queries from $H^{(0)}, H^{(1)}$ as their admissibility is a strict subset of admissible queries for $\mathsf{MPIPE}$.) $\mathcal{A}_{\mathsf{MPIPE}}$ parses $H^{(\beta)} = (\mathcal{W}^{(\beta)}, F^{(\beta)})$, $\beta \in \{0,1\}$ as follows: $\mathcal{A}_{\mathsf{MPIPE}}$ creates the queries $S = (\vec{w}_1^{(0)}, \vec{w}_1^{(1)}), \cdots, (\vec{w}_\ell^{(0)}, \vec{w}_\ell^{(0)})$, where $\vec{w}_i^{(\beta)} \in \mathcal{W}^{(\beta)}$ and $i \in \{1, \cdots, \ell\}$. From each $f_j^{(\beta)} \in F^{(\beta)}$, $j \in \{1, \cdots, q\}$, $\mathcal{A}_{\mathsf{MPIPE}}$ extracts a vector $\vec{y}_j^{(\beta)} \in \mathbb{Z}_q^n$ and a distance threshold $t_j^{(\beta)} \in \mathbb{N}$. $\mathcal{A}_{\mathsf{MPIPE}}$ creates the key generation queries $R = (\vec{y}_1^{(0)}, \vec{y}_1^{(1)}, D_1^{(0)}, D_1^{(1)}), \cdots, (\vec{y}_q^{(0)}, \vec{y}_q^{(1)}, D_q^{(0)}, D_q^{(1)})$ where $D_j^{(\beta)} = \{1, \cdots, t_j^{(\beta)}\}$.
3. $\mathcal{A}_{\mathsf{MPIPE}}$ sends $R$ and $S$ to $\mathcal{C}_{\mathsf{MPIPE}}$ and receives back a set of tokens $T^{(\beta)} = \mathsf{tk}_1^{(\beta)}, \cdots, \mathsf{tk}_q^{(\beta)}$ and a set of encrypted keywords $C^{(\beta)} = \mathsf{ct}_1^{(\beta)}, \cdots, \mathsf{ct}_\ell^{(\beta)}$ such that

$$\begin{aligned}
\mathsf{tk}_j^{(\beta)} &\leftarrow \mathsf{MPIPE.TokGen}(\mathsf{sk}, \vec{y}_j^{(\beta)}, D_j^{(\beta)}), \\
\mathsf{ct}_i^{(\beta)} &\leftarrow \mathsf{MPIPE.Encrypt}(\mathsf{sk}, \vec{w}_i^{(\beta)}).
\end{aligned}$$

   $\mathcal{A}_{\mathsf{MPIPE}}$ forwards $T^{(\beta)}$ and $C^{(\beta)}$ to $\mathcal{A}_{\mathsf{PSE}}$, respectively as the encrypted index $I^{(\beta)}$ and the list of queries $Q^{(\beta)}$.
4. $\mathcal{A}_{\mathsf{MPIPE}}$ receives $\beta' \in \{0,1\}$ from $\mathcal{A}_{\mathsf{PSE}}$ and returns it.

The advantage of $\mathcal{A}_{\mathsf{PSE}}$ is then

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

□

# 5 IPE with reduced key sizes

As described in the introduction, we show a general technique for reducing the size of keys of $\mathsf{IPE}$ schemes that use dual pairing vector spaces (DPVS). The key idea of the technique is to use a different pair of bases for portions of the vectors and add blinding factors so that these intermediate values are not useful until combined. We show this technique starting from the $\mathsf{IPE}$ 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. So this scheme is public key and is not function hiding.[5] Thus, it cannot be directly used to instantiate $\mathsf{PSE}$, we use it as an example due to the relative simplicity of the scheme.

[5] This corresponds to three changes to Definition 1:

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

## 5.1 Dual Pairing Vector Spaces and Decisional Linear Assumption

**Definition 6** (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 $1^\lambda$ and outputs a description of bilinear pairing groups $(q, \mathbb{G}, \mathbb{G}_T, g, e)$ with security parameter $\lambda$.*

We use the symmetric version of dual pairing vector spaces [OT15] where the pairing is based on symmetric bilinear groups defined in Definition 6.

**Definition 7** (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 $1^\lambda$ ($\lambda \in \mathbb{N}$) 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 2.** *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, $\mathcal{G}_\beta^{\mathsf{DLIN}}$, which on input $1^\lambda$:*

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

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

## 5.2 Construction

This construction is an adaptation of Okamoto and Takashima's IPE scheme [OT12, Section 4] (setting $\alpha = 1$ in Figure 2 yields the original scheme). As in the original construction, we first need to describe a random dual orthonormal bases generator, $\mathcal{G}_{\mathsf{ob}}^{\mathsf{IPE}^*}$, which will be called in the main construction's $\mathsf{Setup}$ algorithm to generate the master keys. This is different from the previous generator as it generates $\alpha$ sets of bases.

**Construction 4** (Dual Orthonormal Bases Generator)**.** *Let* $\mathcal{G}_{\mathsf{dpvs}}$ *be a symmetric dual-pairing vector space generator as described in Section 5.1. Let* $\lambda, N, \alpha \in \mathbb{N}$*, where* $\lambda$ *is the security parameter,* $N$ *is the dimension of the vector space and* $\alpha$ *is the number of dual orthonormal bases pairs to generate. Then on inputs* $1^{\lambda}$*,* $N$ *and* $\alpha$*, the orthonormal bases generator* $\mathcal{G}_{ob}^{IPE^*}$ *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,* $\psi \xleftarrow{\$} \mathbb{F}_q^{\times}$.
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* $1 \leq \ell \leq \alpha$*:*
   (a) *Sample a random map, as described in Lemma 2,* $X_\ell = (\chi_{\ell,i,j}) \xleftarrow{\$} GL(N, \mathbb{F}_q)$ *and set* $(\vartheta_{\ell,i,j}) = \psi \cdot (X_\ell^T)^{-1}$*, where* $1 \leq i, j \leq N$.
   (b) *For* $1 \leq i \leq N$*, 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* $(\boldsymbol{a}_1, \cdots, \boldsymbol{a}_N) = \mathbb{A}$.
   (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 5.** *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 2.*

*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 2. 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

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

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 $\mathsf{sk} = \{\hat{\mathbb{B}}^*_\ell\}_{\ell=1,\cdots,\alpha}$.

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

1. Sample $\omega \leftarrow \mathbb{F}_q$
2. Divide $\vec{x}$ in $\alpha$ smaller vectors of length $n/\alpha$, such that $\vec{x} = (\vec{x}_1, \cdots, \vec{x}_\alpha)$.
3. For $1 \le \ell \le \alpha$, sample $\zeta_\ell, \varphi_\ell \xleftarrow{\$} \mathbb{F}_q$ and set $\boldsymbol{c}_\ell := (\overbrace{\zeta_\ell}^{1}, \overbrace{\omega\vec{x}_\ell}^{n/\alpha}, 0^{3n/\alpha}, \overbrace{\varphi_\ell}^{1})_{\mathbb{B}_\ell}$
4. Set $c_0 := m g_{\mathsf{T}}^{(\sum_{\ell=1}^{\alpha} \zeta_\ell)}$
5. Return $\mathsf{ct}_{\vec{x}} := (c_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_\alpha)$

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

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

$\mathsf{Decrypt}(\mathsf{pk}, \mathsf{ct}_{\vec{x}}, \mathsf{sk}_{\vec{v}})$ :

Return $m' = \prod_{\ell=1}^{\alpha} e(\boldsymbol{c}_\ell, \boldsymbol{k}_\ell) \;/\; c_0$

Figure 2: Description of modified IPE algorithms.

| Component | Number of Group Elements |
|---|---|
| Secret Key | $\alpha(2n/\alpha+1)(4n/\alpha+2) = 8n^2/\alpha + 8n + 2\alpha$ |
| Public Key | $\alpha(n/\alpha+2)(4n/\alpha+2) = 4n^2/\alpha + 10n + 4\alpha$ |
| Ciphertext | $\alpha(4n/\alpha+2) = 4n + 2\alpha$ |
| Token | $\alpha(4n/\alpha+2) = 4n + 2\alpha$ |

Table 2: Sizes in Group Elements of Each Component of Revised Scheme. The value $\alpha$ 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.

key being $2n + 1$. Ciphertexts and tokens are a single vector. By splitting into $\alpha$ bases we introduce an $\alpha$ 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].

**Theorem 4.** *The IPE construction in Figure 2 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 $\lambda \in \mathbb{N}$*

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

*where $\nu \in \mathbb{N}$ is the maximum number of key queries $\mathcal{A}$ can make.*[6]

We give a high level intuition here. 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 4.* 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 3.** *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.*[7]

*Proof of Lemma 3.* 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 3.

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 independent from $\beta$. 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

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

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

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

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 $\omega, \zeta_\ell, \varphi_\ell \xleftarrow{\$} \mathbb{F}_q$.

[6] 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.

[7] 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.

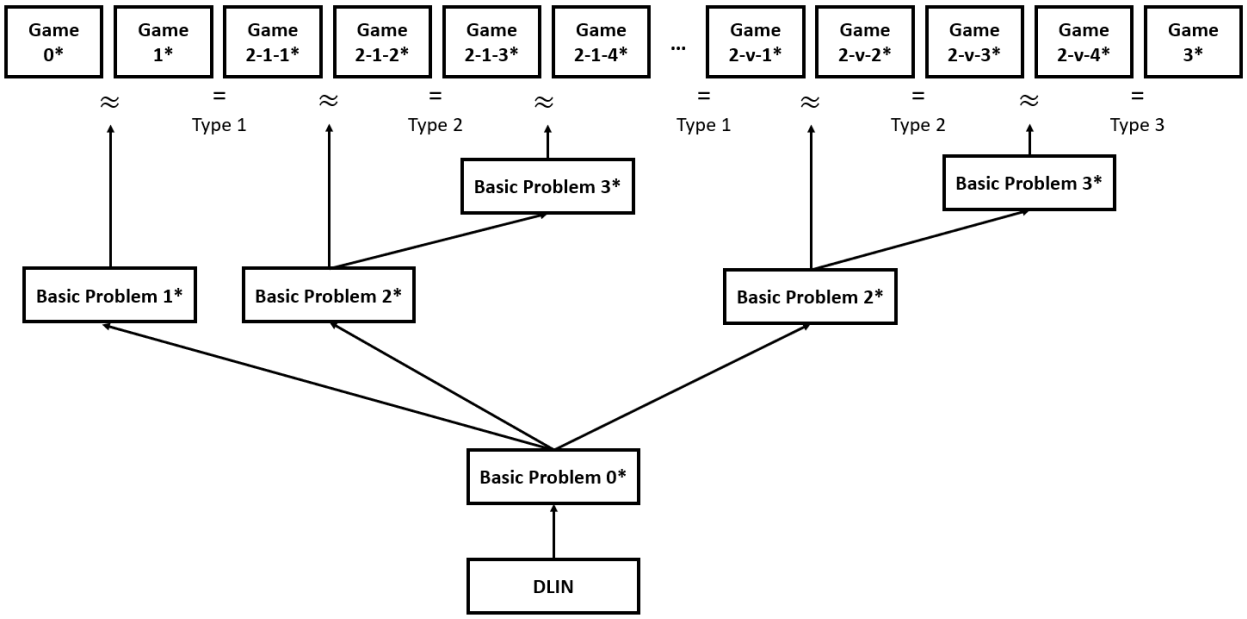



Figure 3: Structure of reductions

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

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

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

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

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

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

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

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

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

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

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-$\nu$-2*, except that the challenge ciphertexts are now

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

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 $\mathsf{Adv}_{\mathcal{A}}^{(g*)}(\lambda)$, 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

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

In the above, bounds on $\mathsf{Adv}_{\mathcal{D}_1}^{\mathsf{bp1}^*}(\lambda), \mathsf{Adv}_{\mathcal{D}_{2\text{-}h\text{-}1}}^{\mathsf{bp2}^*}(\lambda)$ and $\mathsf{Adv}_{\mathcal{D}_{2\text{-}h\text{-}1}}^{\mathsf{bp3}^*}(\lambda)$ are described in Lemmas 5, 6 and 7 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 $\xi$ and $\kappa$ 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.

**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 $1^\lambda$ and $\alpha$:

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 2, $X_\ell = (\chi_{\ell,1}, \chi_{\ell,2}, \chi_{\ell,3}) \xleftarrow{\$} GL(3, \mathbb{F}_q)$ 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 $\beta$ 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 4.** *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 $\lambda \in \mathbb{N}$,*

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

*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[8] 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 $\alpha$ random linear transformations $W_1, \cdots, W_\alpha$ on $\mathbb{V}$, as shown in Lemma 2.

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

   (a) $\mathcal{D}$ calculates

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

   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 $\boldsymbol{y}_\ell^{(\beta)} = (W_\ell^{-1})^T(\boldsymbol{w}^{(\beta)})$.

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

For the moment assume that $\eta$ and $\kappa$ 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

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

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

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

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

When $\beta = 1$ and $Y^{(1)} = \psi g$ where $\psi \xleftarrow{\$} \mathbb{F}_q$, 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 $W_\ell$ 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:

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

□

**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 $1^\lambda$, $\alpha$ and $n$:

1. Samples $\omega, z \xleftarrow{\$} \mathbb{F}_q$.
2. Samples $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \mathbb{B}_\ell^*\}_{\ell=1,\cdots,\alpha}) \leftarrow \mathcal{G}_{\mathsf{ob}}^{\mathsf{IPE}^*}(1^\lambda, N)$.
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 $\gamma_\ell \xleftarrow{\$} \mathbb{F}_q$.
   (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. Returns $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \hat{\mathbb{B}}_\ell^*, \boldsymbol{g}_{\ell,1}^{(\beta)}, \{\boldsymbol{g}_{\ell,i}\}_{i=2,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$.

Then Basic Problem 1* consists in guessing $\beta$ given

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

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

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

**Lemma 5.** *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 $\lambda \in \mathbb{N}$,*

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

See Appendix A.1 for the proof.

**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 $\beta$ 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

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

**Lemma 6.** *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,*

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

See Appendix A.2 for the proof.

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

Basic Problem 3* consists in guessing $\beta$ 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 7.** *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 $\lambda \in \mathbb{N}$,*

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

See Appendix A.3 for the proof. This completes the proof of Lemma 3. □

**Acknowledgements** The authors wish to thank Daniel Wichs for important preliminary discussion. In addition, the authors thank Sohaib Ahmad for running iris data analysis. 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

[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.

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In *Advances in Cryptology – EUROCRYPT 2017*, pages 601–626, Cham, 2017. Springer International Publishing.

[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.

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

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *Advances in Cryptology – CRYPTO 2008*, pages 360–378, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

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

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *Advances in Cryptology – ASIACRYPT 2015*, pages 470–491, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[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.

[CJJ+13] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Annual cryptology conference*, pages 353–373. Springer, 2013.

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

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

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

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *IACR International Workshop on Public Key Cryptography*, pages 245–277. Springer, 2018.

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

[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.

[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.

[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.

[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.

[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.

[Kha18] Rachna Khaira. Rs 500, 10 minutes, and you have access to billion aadhaar details. 2018.

[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+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.

[KM18] Seny Kamara and Tarik Moataz. Sql on structurally-encrypted databases. In *Advances in Cryptology – ASIACRYPT 2018*, pages 149–180, Cham, 2018. Springer International Publishing.

[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.

[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.

[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.

[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.

[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.

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

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

[RKV95] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 71–79, 1995.

[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.

[TAO16] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In *Information Security*, pages 408–425, Cham, 2016. Springer International Publishing.

[TT20] Junichi Tomida and Katsuyuki Takashima. Unbounded inner product functional encryption from bilinear maps. *Japan Journal of Industrial and Applied Mathematics*, 04 2020.

[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.

[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.

# A IPE key reduction proofs

## A.1 Proof of Lemma 5

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 $\mathsf{pp}$ 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 $1 \le \ell \le \alpha$ :
   (a) Sample a random linear transformation $W_\ell$ 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 $2 \le i \le n$, compute $\boldsymbol{g}_{\ell,i} = W_\ell(0^i, \delta\xi g, 0^{N-i-1})$.
   (d) Compute:

$$\begin{aligned}
\boldsymbol{d}_{\ell,1} &= W_\ell(0, \boldsymbol{b}^*_{\ell,1}, 0^{N-4}), \\
\boldsymbol{d}_{\ell,n/\alpha+1} &= W_\ell(0, \boldsymbol{b}^*_{\ell,2}, 0^{N-4}), \\
\boldsymbol{d}_{\ell,N} &= W_\ell(0, \boldsymbol{b}^*_{\ell,3}, 0^{N-4}), \\
\boldsymbol{d}_{\ell,i} &= W_\ell(0^{i+1}, \xi g, 0^{N-i-2}), && i = 0, 2 \le i \le n/\alpha \\
\boldsymbol{d}_{\ell,i} &= W_\ell(0^i, \xi g, 0^{N-i-1}), && n/\alpha + 2 \le i \le N-1.
\end{aligned}$$

   (e) Consider the following vectors ( $\boldsymbol{d}^*_{\ell,n/\alpha+1}$ is not efficiently computable)

$$\begin{aligned}
\boldsymbol{d}^*_{\ell,1} &= (W^{-1}_\ell)^T(0, \boldsymbol{b}_{\ell,1}, 0^{N-4}), \\
\boldsymbol{d}^*_{\ell,n/\alpha+1} &= (W^{-1}_\ell)^T(0, \boldsymbol{b}_{\ell,2}, 0^{N-4}), \\
\boldsymbol{d}^*_{\ell,N} &= (W^{-1}_\ell)^T(0, \boldsymbol{b}_{\ell,3}, 0^{N-4}), \\
\boldsymbol{d}^*_{\ell,i} &= (W^{-1}_\ell)^T(0^{i+1}, \kappa g, 0^{N-i-2}), && i = 0, 2 \le i \le n/\alpha, \\
\boldsymbol{d}^*_{\ell,i} &= (W^{-1}_\ell)^T(0^i, \kappa g, 0^{N-i-1}), && n/\alpha + 2 \le i \le N-1.
\end{aligned}$$

   (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 $\xi g$, $\mathcal{A}_{\mathsf{bp0}^*}$ 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 $\kappa g$, $\mathcal{A}_{\mathsf{bp0}^*}$ 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

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

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 $3 \times 3$ 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:

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

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 $\gamma \stackrel{def}{=} \sigma$:

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

## A.2 Proof of Lemma 6

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 $(\mathsf{pp}, \{\hat{\mathbb{B}}_\ell, \mathbb{B}^*_\ell, \boldsymbol{y}_\ell^{(\beta)}, \boldsymbol{f}_\ell\}_{\ell=1,\cdots,\alpha}, \kappa g, \xi g, \delta\xi g) \leftarrow \mathcal{G}_\beta^{\mathsf{bp0}^*}(1^\lambda, \alpha)$.
2. Extract $g_T$ and $\mathsf{param}_{\mathbb{G}}(q, \mathbb{G}, \mathbb{G}_T, G, e)$ from $\mathsf{pp}$, 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 $W_\ell = (w_{\ell,1}, \cdots, w_{\ell,N}) \stackrel{\$}{\leftarrow} GL(N, \mathbb{F}_q)$.
   (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 $\boldsymbol{h}_{\ell,i}^{(\beta)} = (W_\ell^{-1})^T(0, 0^{3(i-1)}, \boldsymbol{y}_\ell^{(\beta)}, 0^{3(N-i)}, 0)$.
   (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. Sends $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{D}^*_\ell, \hat{\mathbb{D}}_\ell, \{\boldsymbol{h}_{\ell,i}^{(\beta)}, \boldsymbol{g}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$ to $\mathcal{A}_{\mathsf{bp2}^*}$.
5. Return $\beta'$ from $\mathcal{A}_{\mathsf{bp2}^*}$.

From $\hat{\mathbb{B}}_\ell = (\boldsymbol{b}_{\ell,1}, \boldsymbol{b}_{\ell,3})$ and $\xi g$, $\mathcal{A}_{\mathsf{bp0}^*}$ 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 $\kappa g$, $\mathcal{A}_{\mathsf{bp0}^*}$ 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 $1 \le i \le n/\alpha$:

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

We then have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{bp2}^*}_{\mathcal{A}_{\mathsf{bp2}^*}}(\lambda) &= \mathsf{Adv}^{\mathsf{bp0}^*}_{\mathcal{A}_{\mathsf{bp0}^*}}(\lambda)\\ &\leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{D}}(\lambda) + \frac{5}{q}.\end{aligned}$$

## A.3 Proof of Lemma 7

Basic Problem 3* can be decomposed into two experiments, Experiment 3-1 and 3-2 (Definitions 8 and 9 respectively). We will show that these two games are close and then use the triangle inequality. We now define these experiments.

**Definition 8** (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 $(\mathsf{param}_{\mathbb{V}}, \{\mathbb{B}_\ell, \mathbb{B}^*_\ell\}_{1\leq\ell\leq\alpha}) \leftarrow \mathcal{G}^{\mathsf{IPE}^*}_{ob}(1^\lambda, N, \alpha)$.*
2. *For $1 \leq \ell \leq \alpha$, 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 $1 \leq \ell \leq \alpha$, for $1 \leq i \leq n/\alpha$ set:*

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

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

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

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

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

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

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

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

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

*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*

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

**Lemma 8.** *For any PPT distinguisher* $\mathcal{D}$ *and for any security parameter* $\lambda \in \mathbb{N}$,

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

*Proof.* Sample $\theta \xleftarrow{\$} \mathbb{F}_q$. Then for $1 \leq i \leq n/\alpha$ 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 $1 \leq \ell \leq \alpha$, 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

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

In the above, $\tau' = \text{-}\theta\tau$, $\tilde{\omega}' = \omega' + \theta\omega''$ and $\tilde{\kappa}' = \kappa' + \theta\kappa''$. Notice that since $\theta$, $\omega'$ and $\kappa'$ are sampled independently and uniformly, then $\tau'$, $\tilde{\omega}'$ 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 9.** *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* $\lambda \in \mathbb{N}$,

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

*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 $(\mathsf{param}_V, \{\hat{\mathbb{B}}_\ell, \mathbb{B}^*_\ell, \{\boldsymbol{h}^{(\beta)}_{\ell,i}, \boldsymbol{g}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$, $\mathcal{A}_{\mathsf{bp2}^*}$ 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

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

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

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

$\mathcal{A}_{\mathsf{bp2}^*}$ sends $(\mathsf{param}_{\mathbb{V}}, \{\hat{D}_\ell, \hat{D}^*_\ell, \{\boldsymbol{h}^{(\beta)}_{\ell,i}, \boldsymbol{g}_{\ell,i}, \boldsymbol{f}_{\ell,i}\}_{i=1,\cdots,n/\alpha}\}_{\ell=1,\cdots,\alpha})$ to $\mathcal{D}$ and receives back $\beta' \in \{0,1\}$. $\mathcal{A}_{\mathsf{bp2}^*}$ outputs $\beta'$. Thus,

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

Since $\delta, \tau, \omega, \sigma, \eta_1$ and $\eta_2$ 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 8, 9 and 6, 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

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