Access-Controlled Inner Product Function-Revealing Encryption

Ojaswi Acharya[1], Weiqi Feng[1], Roman Langrehr[2], and Adam O'Neill[1]

[1] Manning CICS, University of Massachusetts Amherst, USA
{oacharya,weiqifeng,adamoneill}@umass.edu
[2] ETH Zurich, Switzerland
roman.langrehr@inf.ethz.ch

Abstract. We extend the concept of access control for functional encryption, introduced by Abdalla *et al.* (ASIACRYPT 2020), to *function-revealing* encryption (Joy and Passelègue, SCN 2018). Here "access control" means that function evaluation is only possible when a specified access policy is met. Specifically, we introduce *access-controlled inner-product function-revealing encryption (AC-IPFRE)* and give two applications.

On the theoretical side, we use AC-IPFRE to show that function-hiding inner-product functional encryption (FH-IPFE), introduced by Bishop *et al.* (ASIACRYPT 2015), is equivalent to IPFRE. To show this, we in particular generically construct AC-IPFRE from IPFRE for the "non-zero inner-product" (NZIP) access policy. This result uses an effective version of Lagrange's Four Square Theorem. One consequence of this result is that lower bounds by Ünal (EUROCRYPT 2020) suggest that, as for FH-IPFE, bilinear pairings will be needed to build IPFRE.

On the practical side, we build an outsourced approximate nearest-neighbor (ANN) search protocol and mitigate its leakage via AC-IPFRE. For this, we construct a practical AC-IPFRE scheme in the generic bilinear group model for a specific access policy for ANN search. To this end, we show that techniques of Wee (TCC 2020) implicitly give the most practical FH-IPFE scheme to date. We implement the resulting outsourced ANN search protocol and report on its performance.

Of independent interest, we show AC-IPFRE for NZIP implies *attribute-hiding* small-universe AC-IPFRE for arbitrary access policies. Previous work on access control for FE did not achieve attribute hiding. Overall, our results demonstrate that AC-IPFRE is of both theoretical and practical interest and set the stage for future work in the area.

1 Introduction

FE AND FRE. Traditional encryption is an all-or-nothing affair: either a receiver has the decryption key and can recover m from its ciphertext ct_m, or it does not, and therefore learns nothing about m from ct_m. To address this, in *functional encryption* [19] different receivers have different "functional keys." Namely, a functional key sk_f associated to a function f allows the receiver to recover $f(m)$ from ct_m. One can be extend to the multi-arity functions f accordingly [26]. Moreover, a functional encryption scheme is called *function-hiding* if sk_f hides f. A related concept to FE is *function-revealing encryption* [33]. Here there are no functional keys; rather, one builds a dedicated scheme for an n-ary g such that from ciphertexts $\mathsf{ct}_{m_1}, \ldots \mathsf{ct}_{m_n}$, *anyone* can recover $f(m_1, \ldots, m_n)$. Here, for $n = 1$ such a scheme is trivial, unless one constrains the length of a ciphertext.

ADDING ACCESS CONTROL. A problem with deploying FE in practice is that *too much* information about the messages can be leaked to the receivers. This is because *every* functional key sk_f is "compatible" with *every* ciphertext ct_m, *i.e.* the receiver can always recover $f(m)$. In many applications, it is desirable that the sender have fine-grained control over which ciphertexts are compatible with a given sk_f. Of course, this is trivially possible using FE for *all* efficient functions, which follows from recent breakthroughs in indistinguishability obfuscation [25,30]. However, we are interested in more practical schemes for simple functionalities.

Access control for *inner-product* FE was previously addressed by Abdalla *et al.* [2] and numerous follow-up works *e.g.* [38,4,5,23]. The idea is to associate ciphertexts and keys to tags, *i.e.* $\mathsf{ct}_{\mathbf{x},s}$ and $\mathsf{sk}_{\mathbf{y},t}$ for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^d$, which reveals $\langle \mathbf{x}, \mathbf{y} \rangle$ iff $a(s,t)$ for an access policy $a(\cdot,\cdot)$ fixed by the scheme. Broadly, in this work we extend their study to inner-product *FRE* (IPFRE). Moreover, even without access control, IPFRE is new to this work.

1.1 Our New Notion: AC-IPFRE

Accordingly, we first introduce (private-key) *inner-product function-revealing encryption* (IPFRE), an instance of the more general notion of function-revealing encryption [33,27]. In IPFRE, there are no function keys, and a ciphertext $\mathsf{ct}_{\mathbf{x}}$ encrypts a vector $\mathbf{x} \in \mathbb{Z}_m^d$. Given ciphertexts $\mathsf{ct}_{\mathbf{x}}, \mathsf{ct}_{\mathbf{y}}$ encrypting $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_m^d$ respectively, anyone can compute $\langle \mathbf{x}, \mathbf{y} \rangle$. Security, which we formulate using both indistinguishability and simulation-based notions, requires nothing else is leaked.

As for IPFE, many applications do not need the inner-product for all pairs of vectors. For example, we will see this is case below in an application to outsourcing approximate nearest-neighbor search. To reduce leakage, we introduce *access-controlled* IPFRE (AC-IPFRE), where there is an *access policy* $a \colon \mathcal{T} \times \mathcal{T} \to \{0,1\}$ such that a ciphertext $\mathsf{ct}_{\mathsf{tag},\mathbf{x}}$ is associated with a tag[3] $\mathsf{tag} \in \mathcal{T}$ in addition to a

[3] We use the term "tag" and not "attribute", because here each ciphertext can only have one tag. This is in contrast to attribute-based encryption, where a ciphertext can have many attributes.

plaintext vector $\mathbf{x} \in \mathbb{Z}_m^d$, and given two ciphertexts $\mathsf{ct}_{\mathsf{tag},\mathbf{x}}, \mathsf{ct}_{\mathsf{tag}',\mathbf{x}'}$ anyone can learn $\langle \mathbf{x}, \mathbf{x}' \rangle$ iff $a(\mathsf{tag}, \mathsf{tag}') = 1$. We are interested in two types of access policies. The first one is the "non-zero inner product" access policy a_{nzip} where tags are vectors in $\mathcal{T} = \mathbb{Z}_m^\tau$ and a outputs 1 on $\mathsf{tag}, \mathsf{tag}'$ iff $\langle \mathsf{tag}, \mathsf{tag}' \rangle \neq 0$. We call this non-zero inner-product AC-IPFRE (NZIP-AC-IPFRE). The second one is the "small tag universe" access policy, where tags come from a polynomial-size set $\mathcal{T}$ and the access policy a is an arbitrary predicate.

1.2 Equivalence of IPFRE and FH-IPFE

Our first application of AC-IPFRE is to show that IPFRE is in fact *equivalent* to function-hiding IPFE (FH-IPFE). This result has implications for assumptions that imply IPFRE, because FH-IPFE constructions are believed to require bilinear maps; in particular, lattice-based constructions will be difficult [49].

From FH-IPFE to IPFRE. We first show that IPFRE can be constructed generically from FH-IPFE, where an IPFRE ciphertext for the vector $\mathbf{x}$ consists of an FH-IPFE ciphertext and an FH-IPFE functional key for the same vector $\mathbf{x}$. This idea goes back to [43,3].

From IPFRE to FH-IPFE. The other direction is more difficult. Here, we first show that we can generically equip an IPFRE with the non-zero inner product (NZIP) access policy. From a technical perspective, the construction exploits the algebraic relation of tensor and inner product. If we further restrict the set of tags $\mathcal{T} \subseteq \mathbb{Z}_m^\tau$ such that it satisfies for all $\mathbf{t}, \mathbf{t}' \in \mathcal{T}$ $\langle \mathbf{t}, \mathbf{t}' \rangle \in \{0, 1\}$, our construction also hides the tags. This might seem restrictive at first, but we show that even with this restriction we can realize arbitrary small tag universe access policies, by constructing a suitable vector $\mathbf{v}(\mathsf{tag})$ for each tag tag. Interestingly, a key ingredient of this construction is *Lagrange's four square theorem*.

In more detail, we can represent any access policy by an undirected graph where the nodes are the tags and there is an edge between nodes $\mathsf{tag}_1, \mathsf{tag}_2$ if $a(\mathsf{tag}_1, \mathsf{tag}_2) = 1$. For every non-loop edge $e = \{\mathsf{tag}_1, \mathsf{tag}_2\}$ with $\mathsf{tag}_1 \neq \mathsf{tag}_2$ we add one dimension with index i to the tag vectors and set $\mathbf{v}(\mathsf{tag}_1)_i = \mathbf{v}(\mathsf{tag}_2)_i = 1$ and 0 for all other tags. These vectors already satisfies $\langle \mathbf{v}(\mathsf{tag}_1), \mathbf{v}(\mathsf{tag}_2) \rangle \in \{0, 1\}$ and $\langle \mathbf{v}(\mathsf{tag}_1), \mathbf{v}(\mathsf{tag}_2) \rangle = 1 \iff a(\mathsf{tag}_1, \mathsf{tag}_2) = 1$, as desired. The challenging part is to enforce $\langle \mathbf{v}(\mathsf{tag}_1), \mathbf{v}(\mathsf{tag}_1) \rangle \in \{0, 1\}$, depending on the whether a loop for tag_1 exists in the access graph or not. To do this, let $s := \langle \mathbf{v}(\mathsf{tag}_1), \mathbf{v}(\mathsf{tag}_1) \rangle$ and s' is $-s$ or $-s+1$ (depending on whether $a(\mathsf{tag}_1, \mathsf{tag}_1) = 1$ or not). Our goal is now to find a vector $\mathbf{v}'(\mathsf{tag}_1) \in \mathbb{Z}_q^\ell$ for some ℓ with $\langle \mathbf{v}'(\mathsf{tag}_1), \mathbf{v}'(\mathsf{tag}_1) \rangle = s'$. With such a vector we can add additionally ℓ dimensions to the tag vectors and append $\mathbf{v}'(\mathsf{tag}_1)$ to $\mathbf{v}(\mathsf{tag}_1)$ and fill it with zeros in $\mathbf{v}(\mathsf{tag}')$ for all vectors $\mathsf{tag}' \neq \mathsf{tag}_1$. Performing this step for all tags, will give us vectors that encode the access policies as needed for the NZIP access policy.

The problem of finding $\mathbf{v}'(\mathsf{tag}_1)$ with $\langle \mathbf{v}'(\mathsf{tag}_1), \mathbf{v}'(\mathsf{tag}_1) \rangle = s'$ is equivalent to writing s as a sum of squares $s' = a_1^2 + a_2^2 + \cdots + a_\ell^2$. A classical result due to Lagrange guarantees us that a solutions with $\ell \leq 4$ exists and it can also be computed efficiently [46,45].

Finally, we can turn an AC-IPFRE (for small tag universe) into an FH-IPFE by using two different tags $\mathsf{tag}_1, \mathsf{tag}_2$. An FH-IPFE ciphertexts consists of an AC-IPFRE ciphertext with tag_1 and an FH-IPFE functional key consists of an AC-IPFRE ciphertext with tag_2. The access policy is satisfied if exactly one of the two tags is tag_1.

Generalizations. All our results also apply to orthogonality revealing encryption. In particular, we also obtain equivalence of orthogonality-revealing encryption and function-hiding predicate-encryption for orthogonality. Moreover, our results apply to schemes that only recover the inner product in the exponent of a group element (as typical for pairing-based schemes). Formally, we show our results for any scheme that only depends on the inner-product of the two inputs. We call this an inner-product-*based* functionality. We then present our formal results for inner-product-*based* functional encryption (IPBFE) and inner-product-*based* function revealing encryption (IPBFRE).

1.3 Using AC-IPFRE for Inner-Product Similarity

Many applications, especially in machine learning, information retrieval, and statistics rely on a notion of *inner-product similarity* between data vectors (which can be defined in several ways). Privacy of the data vectors are a natural concern. AC-IPFRE can be used to add privacy to such applications while protecting the data vectors in a fine-grained manner. We give two concerete examples.

Outsourcing approximate nearest-neighbor search. A number of state-of-the-art retrieval models for search engines, *e.g.*, dense retrieval models [35,52,51], use *nearest neighbor search* to retrieve documents in response to a search query [40]. These models learn distributed representations (vectors) for queries and documents using deep neural networks and compute their inner-product similarity. As these applications deal with nearest neighbor search on massive data collections, *exact* nearest neighbor search, which simply uses brute-force search, is fundamentally impractical. They instead rely on *approximate* nearest-neighbor (ANN) search that indexes the vectors in a data structure. Recent such protocols can run efficiently even on billion scale collections, see [32]. Outsourcing storage and ANN search on the vectors to a powerful server is also desirable.

For privacy-preserving outsourcing of *exact* nearest neighbor search, Kim *et al.* [36] propose to use FH-IPFE. However, FH-IPFE only supports *brute-force* search. Using IPFRE, ANN search on encrypted data can work in exactly the same way as state-of-the-art algorithms for plaintext data (except for how an inner-product between two encrypted vectors is computed). Namely, a user encrypts its vectors under IPFRE, sends the resulting ciphertexts to the server who, using IPFRE evaluation, indexes them into an ANN data structure. Later, the user can encrypt a query vector and send this ciphertext to the server, who using IPFRE evaluation can perform fast search.

Unfortunately, this protocol leaks all pairwise inner-product across data, query, and update vectors. To mitigate the leakage, we can use AC-IPFRE with tag-space $\{\mathsf{data}, \mathsf{query}, \mathsf{update}\}$ and access policy a_{ann} defined as $a_{\mathsf{ann}}(\mathsf{tag}_1, \mathsf{tag}_2) = 1$

unless $\mathsf{tag}_1 = \mathsf{tag}_2 = \mathsf{data}$ or $\mathsf{tag}_1 = \mathsf{tag}_2 = \mathsf{query}$. The idea is to split the database into vectors in the initial dataset and those later added dynamically. The user indexes its initial vectors in *plaintext*, then encrypts each one with tag data, sending the resulting data structure to the server. To query, the user sends an encryption of its query vector with tag query. To update the database, the user sends the encryption of the update vector with tag update. This access policy allows the server to learn only the inner-products necessary to maintain the data structure and answer the queries. Note that the data-structure can leak some information about the initial data vectors to the server. In the case of HSNW [41], the inner-product comparison bits of the initial vectors are leaked, *i.e.*, from three such encrypted vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, the server can tell if $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle \geq \langle \mathbf{x}_1, \mathbf{x}_3 \rangle$.

We formalize the leakage profile of our AC-IPFRE based ANN search protocol and show that any FH-IPFE can be upgraded to an AC-IPFRE for a_{ann} more efficiently than going through our general transform to small tag universe AC-IPFRE. We provide an an implementation and evaluation of our ANN search protocol using the TinyFHIPFE scheme discussed below. We demonstrate that for typical workloads query processing takes about 24s with 20ms of active client time. In comparison, a prior protocol based on oblivious data structures [18] involves many rounds of interaction between the client and server, lasting over 8 minutes. On the other hand, we do not hide access pattern.

Pearson's correlation coefficients. In scenarios where we want to encrypt datasets, but still reveal some statistics about these datasets, IPFRE is also a useful tool. Consider for example a hospital that wants to protect personal medical data through encryption, but reveal certain statistics about this data, like the correlation of obesity with certain diseases, without computing each of these statistics. The standard measure for correlation is Pearson's correlation coefficient $r_{\mathbf{x},\mathbf{y}}$ and, if the data vectors $\mathbf{x}, \mathbf{y}$ are normalized to have mean 0, it can be expressed as a form of inner-product (aka. cosine) similarity, namely $r_{\mathbf{x},\mathbf{y}} = \langle \mathbf{x}, \mathbf{y} \rangle / \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle \langle \mathbf{y}, \mathbf{y} \rangle}$, and thus be computed from the inner products of every pair of vectors, which is exactly what is revealed by IPFRE.

Moreover, it can be desirable to *not* reveal certain correlations coefficients, *e.g.*, to reduce leakage or because it is unethical. For example, revealing the correlation between skin color and intelligence quotient can be seen as unethical. This can be realized with AC-IPFRE. We leave a more in-depth treatment of this application to future work.

1.4 TinyFHIPFE: An Efficient FH-IPFE Scheme

We show that part of an FE construction of Wee [50] can be adapted to give an FH-IPFE that is asymptotically and concretely more efficient than existing FH-IPFE schemes, which we call TinyFHIPFE. We note that Wee does not consider FH-IPFE, and his FE scheme differs substantially from ours as it is public-key, supports quadratic functions, and does not achieve function hiding. TinyFHIPFE scheme is our starting point for the AC-IPFRE scheme we implement as part of our ANN-ODB protocol above.

The intuition behind TinyFHIPFE is as follows. We work in prime-order groups $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T$, generated by g_1, g_2, g_T respectively, and equipped with a asymmetric bilinear pairing. The master secret key will consist of matrices. Let us first describe a partial version of the scheme, and then discuss how to complete it. Here, the master secret key will be $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_p^{2\times d}$. Moreover, a function key for a vector $\mathbf{x} \in \mathbb{Z}_p^d$ will be computed as $[\mathbf{A}^T\mathbf{s}_1 + \mathbf{x}]_{g_1}$ (using the notation $[x]_g := g^x$) for $\mathbf{s}_1 \leftarrow\!\!\$\ \mathbb{Z}_p^2$. Then, a ciphertext is formed analogously but in the other source group; namely, encryption of a vector $\mathbf{y} \in \mathbb{Z}_p^d$ will be computed as $[\mathbf{A}^T\mathbf{s}_2 + \mathbf{y}]_{g_2}$ for random $\mathbf{s}_2 \in \mathbb{Z}_p^2$. Now, to evaluate inner-product, suppose we pair a function key $\mathsf{sk}_{\mathbf{x}}$ and ciphertext $\mathsf{ct}_{\mathbf{y}}$. Then we end up with

$$[\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{s}_1^T\mathbf{A}\mathbf{y} + \mathbf{x}^T\mathbf{A}^T\mathbf{s}_2 + \mathbf{s}_1^T\mathbf{A}\mathbf{A}^T\mathbf{s}_2]_{g_T} \ .$$

Then, we observe that the sum of the last three terms can be computed as the inner product of two 4-dimensional vectors: $\mathbf{s}_1 \| (\mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{A}^T\mathbf{s}_1)$ and $\mathbf{A}\mathbf{y} \| \mathbf{s}_2$. So, to allow the evaluator to compute this sum, we augment the ciphertexts and function keys with encryptions and function keys for these two vectors under a "tiny" instance of the (less efficient) FH-IPFE scheme by Kim *et al.* [36]. Namely, we include two 4×4 matrices $\mathbf{B} \leftarrow\!\!\$\ \mathbf{SL}_4(\mathbb{Z}_p), \mathbf{B}^* \leftarrow (\mathbf{B}^{-1})^T$ in the master secret key and add $\mathbf{B} \cdot (\mathbf{s}_1 \| \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{A}^T\mathbf{s}_1)$ to a function key $\mathsf{sk}_{\mathbf{x}}$ and $\mathbf{B}^* \cdot (\mathbf{A}\mathbf{y} \| \mathbf{s}_2)$ to a ciphertext $\mathsf{ct}_{\mathbf{y}}$. Inner product evaluation in our scheme now additionally pairs these components as in the inner product evaluation for the scheme of [36] and uses the result to obtain $[\langle \mathbf{x}, \mathbf{y} \rangle]_{g_T}$. Finally, assuming $\langle \mathbf{x}, \mathbf{y} \rangle$ is suitably bounded, it can be recovered by taking discrete-log to the base g_T. Details of our TinyFHIPFE scheme are given in Section 5 and the security proof is given in Appendix B.

We assess the efficiency of TinyFHIPFE and compare it with the most efficient FH-IPFE schemes in the table in Table 1. Specifically, we compare to the FH-IPFE schemes of [36] (denoted KLM^+) and of [37] (denoted KKS). In the figure, d denotes the input dimension, $\mathsf{mult}_{\mathbb{Z}_p}$ denotes multiplications in $\mathbb{Z}_p$, $\mathsf{mult}_{\mathsf{G}_1}$ denotes multiplications in G_1, and $\mathsf{exp}_{\mathsf{G}_1}$ denotes exponentiations in G_1. Furthermore, 'PP' in the last column stands for *preprocessing-friendly*, that is, discrete-log is computed with respect to a *fixed* base and hence can be made fast using preprocessing. The size of a function key and the runtime of key-generation (not shown) are comparable to the ciphertext size and runtime of encryption respectively for all FH-IPFE schemes considered.

We can see that TinyFHIPFE has the shortest master secret key for $d \geq 8$ while the sizes of its ciphertext and function keys are comparable to the shortest ones. Similarly, the runtime of the encryption and key generation algorithms are linear in the number of multiplication in $\mathbb{Z}_p$ compared to quadratic number of multiplication in $\mathbb{Z}_p$ in KLM^+ and a linear number of multiplications in the group G_1 and G_2 in KKS. Note that multiplications in $\mathbb{Z}_p$ are much faster than multiplications in G_1 and G_2.

1.5 Discussion and Open Problems

Our work lends insight into the relationship between FE and FRE, which historically developed along different paths. Joye and Passelègue [33] cast FRE as a

Scheme	msk size	ct size	Encryption runtime	PP
KLM$^+$ [36]	$(2d^2)_{\mathbb{Z}_p}$	$(d+1)_{\mathsf{G}_1}$	$d^2 \cdot \mathsf{mult}_{\mathbb{Z}_p} + (d+1) \cdot \mathsf{exp}_{\mathsf{G}_1}$	X
KKS [37]	$(6d+4)_{\mathbb{Z}_p}$	$(2d+8)_{\mathsf{G}_1}$	$(2d+2) \cdot \mathsf{mult}_{\mathsf{G}_1} + (7d+12) \cdot \mathsf{exp}_{\mathsf{G}_1}$	✓
TinyFHIPFE	$(2d+32)_{\mathbb{Z}_p}$	$(d+4)_{\mathsf{G}_1}$	$(4d+16) \cdot \mathsf{mult}_{\mathbb{Z}_p} + (d+4) \cdot \mathsf{exp}_{\mathsf{G}_1}$	✓

Table 1. Performance comparison for TinyFHIPFE.

weakening of multi-input FE (MIFE) [26]. That is, while MIFE supports decryption keys associated to multi-input functions chosen on-the-fly, FRE supports a particular multi-input function fixed at setup.

We define AC-IPFRE with "ideal leakage," meaning only the inner product values are leaked. In such a case, despite the fact that our schemes are private-key, even without function hiding public-key assumptions are necessary [28]. However, unlike FE, it is common in the literature for FRE constructions to have less-than-ideal leakage. An interesting direction is to construct AC-IPFRE with limited leakage that is more efficient or is based on weaker assumptions. A particularly fascinating question is how much leakage one must tolerate to rely on only symmetric-key primitives. Note that Fuchsbauer *et al.* [24] build distance-comparison-preserving encryption for Euclidean distance based on symmetric-key primitives; however, their scheme is only shown to resist specific attacks.

To further reduce the leakage in our outsourced ANN search protocol, an open problem is to build FRE for inner-product *comparison*. Indeed, ANN search only requires such comparison and does not need to learn the raw inner-products. We also leave it for future work to analyze the quality of our protocol's leakage profile under recent frameworks such as [17]. Finally, access control may be useful for other forms of FRE.

1.6 Other Related Work

Inner product functional encryption. IPFE (for inner-product recovered in the exponent) was first introduced by Abdalla *et al.* [1], with the function hiding case first considered by Bishop *et al.* [14]. Bishop *et al.* give a weakly function-hiding scheme under SXDH. Other FH-IPFE schemes include a SIM-secure scheme by Kim *et al.* [36] proven in the generic bilinear group model, and IND-secure schemes under SXDH by Tomida *et al.* [48] and Datta *et al.* [22]. These schemes can be plugged into our generic transform to give AC-IPFRE. A scheme from class groups [20] does not recover inner-product in the exponent and hence supports unrestricted inner-product, but is not function hiding.

Function-revealing encryption. FRE grew out of work on deterministic encryption [10], order-preserving encryption [16], and property-preserving encryption [43]. FRE was first defined (independently) by Joye and Passelègue [33] and Haagh *et al.* [27].[4] Joye and Passelègue study FRE for orthogonality and

[4] "Preserving" vs. "revealing" terminology reflects whether the evaluation operation on ciphertexts is required to be the same as the evaluation operation on plaintexts.

cardinality of set intersection, while Haagh *et al.* study FRE for partial ordering and its application to skyline queries. None of these works consider inner-product.

1.7 Organization of the Paper

We start with preliminaries in Section 2. Then, in Section 3, we present the definitions of our two new primitives: inner-product based function-revealing encryption (IPBFRE) in Section 3.1 and access-controlled IPBFRE in Section 3.2, as well as inner-product-based functional encryption in Section 3.3. In Section 4 we present generic constructions of AC-IPBFRE for the non-zero inner product functionality in Section 4.1 and for small tag universes in Section 4.2. We present TinyFHIPFE, our function-hiding inner-product functional encryption scheme in Section 5. Finally, in Section 6 we present an AC-IPFRE for a_{ann} access policy in Section 6.1 and our ANN-ODB protocol with its leakage profile in Section 6.2.

2 Preliminaries

NOTATION. If $\mathbf{v}$ is a vector then $|\mathbf{v}|$ is its length (the number of its coordinates) and $\mathbf{v}[i]$ is its i-th coordinate. For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_m^d$ we use $\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^\top \mathbf{y}$ to denote the standard scalar product. For vectors $\mathbf{x} \in \mathbb{Z}_m^d, \mathbf{y} \in \mathbb{Z}_m^\tau$ we use $\mathbf{x} \otimes \mathbf{y} := \mathrm{vec}(\mathbf{x}\mathbf{y}^\top)$ to denote the standard tensor product for vectors. Here, vec is the vectorization operator takes a matrix and returns a vector obtained by stacking all columns of the input matrix on top of each other. By $\mathbf{x} \| \mathbf{y}$ we denote the concatenation of vectors $\mathbf{x}, \mathbf{y}$ length-wise.

Strings are identified with vectors over $\{0,1\}$. By ϕ we denote the empty string or vector. If S is a finite set, then $|S|$ denotes it size. If X is a finite set, we let $x \leftarrow\!\!\$\, X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. If A is an algorithm, we let $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots; \omega)$ denote running A on inputs $x_1, \ldots$ and coins ω, with oracle access to $\mathrm{O}_1, \ldots$, and assigning the output to y. By $y \leftarrow\!\!\$\, A^{\mathrm{O}_1,\cdots}(x_1, \ldots)$ we denote picking ω at random and letting $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots; \omega)$. We let $\mathbf{Out}(A^{\mathrm{O}_1,\cdots}(x_1, \ldots))$ denote the set of all possible outputs of A when run on inputs $x_1, \ldots$ and with oracle access to $\mathrm{O}_1, \ldots$. Running time is worst case, which for an algorithm with access to oracles means across all possible replies from the oracles. We use $\perp$ (bot) as a special symbol to denote rejection, and it is assumed to not be in $\{0,1\}^*$.

A function $\nu\colon \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p\colon \mathbb{N} \to \mathbb{R}$ there is a $\lambda_p \in \mathbb{N}$ such that $\nu(\lambda) \leq 1/p(\lambda)$ for all $\lambda \geq \lambda_p$. "PT" stands for "polynomial time," whether randomized or deterministic. By 1^λ we denote the unary representation of the integer security parameter $\lambda \in \mathbb{N}$.

GAMES. We use the code-based game-playing framework of BR [11]. By $\Pr[\mathrm{G} \Rightarrow y]$ we denote the probability that the execution of game G results in the output being y. In games, integer variables, set variables, boolean variables and string variables are assumed initialized, respectively, to 0, the empty set $\emptyset$, the boolean false and $\perp$.

3 Definitions

We focus in this work on inner-product based functionalities.

Definition 1 (Inner-product based function) *A function* $f : \mathbb{Z}_m^d \times \mathbb{Z}_m^d \to \mathcal{Y}$ *(for an arbitrary codomain* $\mathcal{Y}$*) is* inner-product based *if it can be expressed as* $f(\mathbf{x}, \mathbf{y}) = g(\langle \mathbf{x}, \mathbf{y} \rangle)$ *for a function* $g : \mathbb{Z}_m \to \mathcal{Y}$*.*

Some examples of inner-product based functions of interest are the function computing the inner product, the inner product in the exponent (where $g(x) = g_T^x$ for a group generator g_T of the target group of a pairing) or the function testing for orthogonality.

3.1 Inner-Product Based Function Revealing Encryption

Syntax. A (private-key) *inner-product based function-revealing encryption (IPBFRE) scheme* for a functionality $f(\mathbf{x}, \mathbf{y}) = g(\langle \mathbf{x}, \mathbf{y} \rangle)$ is a tuple of PT algorithms $\mathsf{IPBFRE} = (\mathsf{PPGen}, \mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval})$ that work as follows:

- $\mathsf{PPGen}(1^\lambda)$: On input a unary encoding of the security parameter $\lambda \in \mathbb{N}$, the public parameters generation algorithm outputs public parameters pp (in particular including a modulus $m \in \mathbb{Z}_+$), which is implicitly input to all algorithms.
- $\mathsf{Setup}(1^d)$: On input a unary encoding of the plaintext dimension $d \in \mathbb{N}$, the setup algorithm outputs a master secret key msk.
- $\mathsf{Enc}(\mathsf{msk}, \mathbf{x})$: On inputs a master secret key msk, and an input vector $\mathbf{x} \in \mathbb{Z}_m^d$, the ecnryption algorithm outputs a ciphertext ct.
- $\mathsf{Eval}(\mathsf{ct}_1, \mathsf{ct}_2)$: On input two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$, the evaluation algorithm outputs a string y.

Remark 1. When the function computed is inner-product i.e., $f(\mathbf{x}, \mathbf{y}) = g(\langle \mathbf{x}, \mathbf{y} \rangle) = \langle \mathbf{x}, \mathbf{y} \rangle$, we call the scheme Inner-Product Function Revealing Encryption (IPFRE).

Correctness. We require *evaluation correctness* meaning that

$$\Pr[\mathsf{Eval}(\mathsf{Enc}(\mathsf{msk}, \mathbf{x}_1), \mathsf{Enc}(\mathsf{msk}, \mathbf{x}_2)) = g(\langle \mathbf{x}_1, \mathbf{x}_2 \rangle)] = 1$$

for all $\lambda, d \in \mathbb{N}$, all $\mathsf{pp} \in \mathbf{Out}(\mathsf{PPGen}(1^\lambda))$ and all $\mathsf{msk} \in \mathbf{Out}(\mathsf{Setup}(\mathsf{pp}, 1^d))$, where $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{Z}_m^d$ and probability is taken over the coins of the Enc algorithm.

Indistinguishability-based security. First, we define an indistinguishability-based security definition for IPBFRE in Fig. 1. For an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$, define its IND-advantage for all $\lambda \in \mathbb{N}$ as:

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{IPBFRE},\mathrm{A}}(\lambda) = \Pr[\mathbf{G}^{\text{ind-1}}_{\mathsf{IPBFRE},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}^{\text{ind-0}}_{\mathsf{IPBFRE},\mathrm{A}}(\lambda) \Rightarrow 1] \,.$$

We say that an IPBFRE scheme is *IND-secure* if $\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{IPBFRE},\mathrm{A}}(\cdot)$ is negligible for all PT adversaries A.

SIMULATION SECURITY. Next, we give a simulation-based security definition for IPBFRE in Fig. 1. For an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ and a simulator $\mathrm{S} = (\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3)$, we define the SIM-advantage of A, S for all $\lambda \in \mathbb{N}$ as follows:

$$\mathsf{Adv}^{\mathrm{sim}}_{\mathsf{IPBFRE},\mathrm{A},\mathrm{S}}(\lambda) = \Pr[\mathbf{G}^{\text{sim-1}}_{\mathsf{IPBFRE},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{IPBFRE},\mathrm{A},\mathrm{S}}(\lambda) \Rightarrow 1] \ .$$

We say that an IPBFRE scheme is *SIM-secure* if for every PT adversary A there exists a PT simulator S such that $\mathsf{Adv}^{\mathrm{sim}}_{\mathsf{IPBFRE},\mathrm{A},\mathrm{S}}(\cdot)$ is negligible.

Game $\mathbf{G}^{\text{ind-b}}_{\mathsf{IPBFRE},\mathrm{A}}(\lambda)$

MAIN:
1 $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PPGen}(1^\lambda)$
2 $(d, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
3 $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^d)$
4 $b' \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{Enc}\mathrm{O}(\cdot,\cdot)}(st)$
5 **if** $\exists\, j, k \le i : g(\langle \mathbf{x}_0^j, \mathbf{x}_0^k \rangle) \neq g(\langle \mathbf{x}_1^j, \mathbf{x}_1^k \rangle)$ **then**
6 **return** $\perp$
7 **else return** $(b = b')$

ENCO($\mathbf{x}_0, \mathbf{x}_1$):
8 $i \leftarrow i + 1$
9 $(\mathbf{x}_0^i, \mathbf{x}_1^i) \leftarrow (\mathbf{x}_0, \mathbf{x}_1)$
10 $\mathsf{ct} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, \mathbf{x}_b)$
11 **return** ct

Game $\mathbf{G}^{\text{sim-1}}_{\mathsf{IPBFRE},\mathrm{A}}(\lambda)$

MAIN:
1 $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PPGen}(1^\lambda)$
2 $(d, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
3 $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^d)$
4 $b \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{Enc}\mathrm{O}(\cdot)}(st)$
5 **return** b

ENCO($\mathbf{x}$):
6 $\mathsf{ct} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, \mathbf{x})$
7 **return** ct

Game $\mathbf{G}^{\text{sim-0}}_{\mathsf{IPBFRE},\mathrm{A},\mathrm{S}}(\lambda)$

MAIN:
1 $\mathcal{C}_{\mathsf{ip}} \leftarrow \emptyset;\ i \leftarrow 0$
2 $(\mathsf{pp}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}_1(1^\lambda)$
3 $(d, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
4 $st_{\mathrm{S}} \leftarrow\!\!\$\ \mathrm{S}_2(1^d, st_{\mathrm{S}})$
5 $b \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{Enc}\mathrm{O}(\cdot)}(st_{\mathrm{A}})$
6 **return** b

ENCO($\mathbf{x}$):
7 $i \leftarrow i + 1$
8 $\mathbf{x}_i \leftarrow \mathbf{x}$
9 **for** $j \le i$ **do**
10 $c_{i,j} \leftarrow g(\langle \mathbf{x}, \mathbf{x}_j \rangle)$
11 $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}_{\mathsf{ip}} \cup \{((i,j), c_{i,j})\}$
12 $(\mathsf{ct}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}_3(\mathcal{C}_{\mathsf{ip}}, st_{\mathrm{S}})$
13 **return** ct

Fig. 1. Games defining IND-security (left) and SIM-security (middle and right) for IPBFRE.

3.2 Access-Controlled IPBFRE

We introduce IPBFRE with *access policies*, following Abdalla *et al.* in the FE setting [2]. An *access policy* is a PT function $a : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$. Here, the arguments in the access policy are public parameters, auxiliary information[5], the first tag, and the second tag respectively. We assume access policies are *symmetric* in the the last two arguments meaning $a(\mathsf{pp}, \mathsf{aux}, x, y) = a(\mathsf{pp}, \mathsf{aux}, y, x)$ for all $\mathsf{pp}, \mathsf{aux}, x, y \in \{0,1\}^*$. We introduce *access-controlled* IPBFRE (AC-IPBFRE), which is defined like IPBFRE except

[5] Looking ahead, the auxiliary information for non-zero inner product access policy will be the dimension of tag vectors.

that ciphertexts are equipped with tags. The inner-product of the plaintexts is revealed by the evaluation algorithm if and only if the tags of the two ciphertexts combine to satisfy the given access policy.

Formally, an AC-IPBFRE for access policy a is a tuple of three algorithms $\mathsf{AC\text{-}IPBFRE}[a] = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval})$.

- $\mathsf{PPGen}(1^\lambda)$: On input a unary encoding of the security parameter λ, the public parameters generation algorithm outputs the public parameters pp (that in particular includes the modulus $m \in \mathbb{Z}_+$). The public parameters are implicitly input to all other algorithms.
- $\mathsf{Setup}(1^d, \mathsf{aux})$: On input a unary encoding of the plaintext dimension d and some auxiliary information about the access policy aux, the setup algorithm outputs a master secret key msk.
- $\mathsf{Enc}(\mathsf{msk}, \mathsf{tag}, \mathbf{x})$: On inputs a master secret key msk, a tag tag, and a message vector $\mathbf{x}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Eval}(\mathsf{ct}_1, \mathsf{ct}_2)$: On input two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$, the evaluation algorithm outputs a scalar y or $\perp$.

Remark 2. When the function computed is inner-product i.e.,$f(\mathbf{x}, \mathbf{y}) = g(\langle \mathbf{x}, \mathbf{y} \rangle) = \langle \mathbf{x}, \mathbf{y} \rangle$, we call the scheme Access-Controlled Inner-Product Function Revealing Encryption (AC-IPFRE).

Correctness. We require *evaluation correctness* meaning that

$$\Pr[\mathsf{Eval}(\mathsf{Enc}(\mathsf{msk}, \mathsf{tag}_1, \mathbf{x}_1), \mathsf{Enc}(\mathsf{msk}, \mathsf{tag}_2, \mathbf{x}_2)) = g(\langle \mathbf{x}_1, \mathbf{x}_2 \rangle)] = 1$$

for all $\lambda, d \in \mathbb{N}$, all $\mathsf{aux} \in \{0,1\}^*$, all $\mathsf{pp} \in \mathbf{Out}(\mathsf{PPGen}(1^\lambda))$, all $\mathsf{msk} \in \mathbf{Out}(\mathsf{Setup}(1^\lambda, \mathsf{aux}))$, all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{Z}_m^d$ and all $\mathsf{tag}_1, \mathsf{tag}_2 \in \{0,1\}^*$ such that $a(\mathsf{pp}, \mathsf{aux}, \mathsf{tag}_1, \mathsf{tag}_2) = 1$, where the probability is taken over the coins of the Enc algorithm.

Security Definitions Our security definitions guarantee that nothing about the tags is leaked, except whether they satisfy the access policy, and nothing about the message vectors is leaked, except the inner product for pairs of vectors encrypted with tags satisfying the access policy.

We give an indistinguishability-based security definition for AC-IPBFRE in Fig. 2. For an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$, define its IND-advantage for all $\lambda \in \mathbb{N}$ as:

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{AC\text{-}IPBFRE},\mathrm{A}}(\lambda) = \Pr[\mathbf{G}^{\text{ind-1}}_{\mathsf{AC\text{-}IPBFRE},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}^{\text{ind-0}}_{\mathsf{AC\text{-}IPBFRE},\mathrm{A}}(\lambda) \Rightarrow 1] \ .$$

We say that an AC-IPBFRE scheme is *IND-secure* if $\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{AC\text{-}IPBFRE},\mathrm{A}}(\cdot)$ is negligible for all PT adversaries A.

Next, we give a simulation-based security definition for AC-IPBFRE in Fig. 3. For an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ and a simulator $\mathrm{S} = (\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3)$, we define the SIM-advantage of A, S for all $\lambda \in \mathbb{N}$ as:

$$\mathsf{Adv}^{\mathrm{sim}}_{\mathsf{AC\text{-}IPBFRE},\mathrm{A},\mathrm{S}}(\lambda) = \Pr[\mathbf{G}^{\text{sim-1}}_{\mathsf{AC\text{-}IPBFRE},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{AC\text{-}IPBFRE},\mathrm{A},\mathrm{S}}(\lambda) \Rightarrow 1] \ .$$

Game $\mathbf{G}^{\text{ind-b}}_{\mathsf{AC\text{-}IPBFRE},A}(\lambda)$

MAIN:
1 $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PPGen}(1^\lambda)$
2 $(d, \mathsf{aux}, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
3 $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^d, \mathsf{aux})$
4 $b' \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{Enco}(\cdot,\cdot,\cdot)}(st)$
5 **if** $\exists\ j,k \le i : (a(\mathsf{pp}, \mathsf{tag}_0^j, \mathsf{tag}_0^k) \neq a(\mathsf{pp}, \mathsf{tag}_1^j, \mathsf{tag}_1^k) \vee (a(\mathsf{pp}, \mathsf{tag}_0^j, \mathsf{tag}_0^k) = 1 \wedge g(\langle \mathbf{x}_0^j, \mathbf{x}_0^k \rangle))) \neq g(\langle \mathbf{x}_1^j, \mathbf{x}_1^k \rangle)$ **then**
6 **return** $\perp$
7 **else return** $b = b'$

$\textsc{Enco}(\mathbf{x}_0, \mathbf{x}_1, \mathsf{tag}_0, \mathsf{tag}_1)$:
8 $i \leftarrow i + 1$
9 $(\mathbf{x}_0^i, \mathbf{x}_1^i, \mathsf{tag}_0^i, \mathsf{tag}_1^i) \leftarrow (\mathbf{x}_0, \mathbf{x}_1, \mathsf{tag}_0, \mathsf{tag}_1)$
10 $\mathsf{ct} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, (\mathbf{x}_b, \mathsf{tag}_b))$
11 **return** ct

Fig. 2. Game defining IND-security for AC-IPBFRE.

We say that an AC-IPBFRE scheme is *SIM-secure* if for every PT adversary A there exists a PT simulator S such that $\mathsf{Adv}^{\text{sim}}_{\mathsf{AC\text{-}IPBFRE},\mathrm{A},\mathrm{S}}(\cdot)$ is negligible.

3.3 Inner Product Based Functional Encryption

A *private-key inner-product based functional encryption* (IPBFE) scheme is a tuple of algorithms $\mathsf{IPBFE} = (\mathsf{PPGen}, \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ that work as follows:

- $\mathsf{PPGen}(1^\lambda)$: On input a unary encoding of the security parameter λ, the public parameters generation algorithm outputs the public parameters pp (that in particular includes the modulus $m \in \mathbb{Z}_+$). The public parameters are implicitly input to all other algorithms.
- $\mathsf{Setup}(1^d)$: On input a unary encoding of the plaintext dimension 1^d, the setup algorithm outputs the master secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$: On input the master secret key msk and a vector $\mathbf{x} \in \mathbb{Z}_m^d$, the key generation algorithm outputs a function key $\mathsf{sk}_\mathbf{x}$.
- $\mathsf{Enc}(\mathsf{msk}, \mathbf{y})$: On input the master secret key msk and a vector $\mathbf{y} \in \mathbb{Z}_m^d$, the encryption algorithm outputs a ciphertext $\mathsf{ct}_\mathbf{y}$.
- $\mathsf{Eval}(\mathsf{sk}_\mathbf{x}, \mathsf{ct}_\mathbf{y})$: On input a secret key $\mathsf{sk}_\mathbf{x}$ and a ciphertext $\mathsf{ct}_\mathbf{y}$, the decryption algorithm outputs a string z.

CORRECTNESS. We require *decryption correctness* meaning that

$$\Pr[\mathsf{Eval}(\mathsf{pp}, \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x}), \mathsf{Enc}(\mathsf{msk}, \mathbf{y})) = g(\langle \mathbf{x}, \mathbf{y} \rangle)] = 1$$

for all $\lambda, d \in \mathbb{N}$, all $\mathsf{pp} \in \mathbf{Out}(\mathsf{PPGen}(1^\lambda))$ and all $\mathsf{msk} \in \mathbf{Out}(\mathsf{Setup}(\mathsf{pp}, 1^d))$, where $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_m^d$ and probability is taken over the coins of the Enc and the KeyGen algorithm.

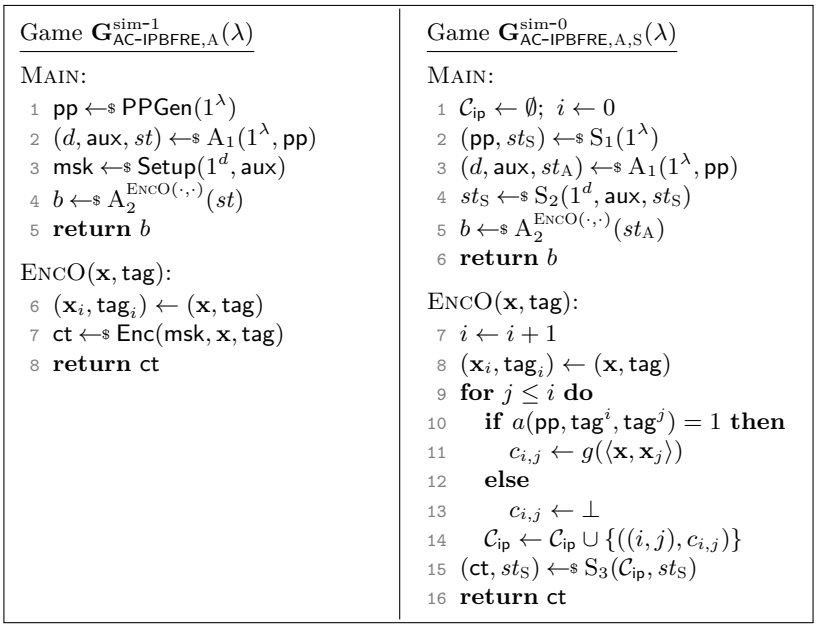


Fig. 3. Games defining SIM-security for AC-IPBFRE.

Security Definitions Again, we consider both indistinguishability and simulation-based definitions.

INDISTINGUISHABILITY-BASED SECURITY. We give an indistinguishability-based security definition for Function-Hiding IPBFE (FH-IPBFE) in Fig. 4. For an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$, define its IND-advantage for all $\lambda \in \mathbb{N}$ as:

$$\mathsf{Adv}^{\text{ind}}_{\text{FH-IPBFE},\mathrm{A}}(\lambda) = \Pr[\mathbf{G}^{\text{ind-1}}_{\text{FH-IPBFE},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}^{\text{ind-0}}_{\text{FH-IPBFE},\mathrm{A}}(\lambda) \Rightarrow 1] \ .$$

We say that an FH-IPBFE scheme is *IND-secure* if $\mathsf{Adv}^{\text{ind}}_{\text{FH-IPBFE},\mathrm{A}}(\cdot)$ is negligible for all PT adversaries A.

SIMULATION SECURITY. Next, we give a simulation-based security definition for FH-IPBFE in Fig. 5. For an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ and a simulator $\mathrm{S} = (\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3, \mathrm{S}_4)$, we define the SIM-advantage of A, S for all $\lambda \in \mathbb{N}$ as:

$$\mathsf{Adv}^{\text{sim}}_{\text{FH-IPBFE},\mathrm{A},\mathrm{S}}(\lambda) = \Pr[\mathbf{G}^{\text{sim-1}}_{\text{FH-IPBFE},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}^{\text{sim-0}}_{\text{FH-IPBFE},\mathrm{A},\mathrm{S}}(\lambda) \Rightarrow 1] \ .$$

We say that an FH-IPBFE scheme is *SIM-secure* if for every PT adversary A there exists a PT simulator S such that $\mathsf{Adv}^{\text{sim}}_{\text{FH-IPBFE},\mathrm{A},\mathrm{S}}(\cdot)$ is negligible.

4 Generic Constructions of AC-IPBFRE

We show in this Section that non-zero inner product AC-IPBFRE (NZIP-AC-IPBFRE), an AC-IPBFRE where tags are vectors and decryption is possible

Game $\mathbf{G}^{\text{ind-b}}_{\mathsf{FH\text{-}IPBFE},\mathrm{A}}(\lambda)$

MAIN:
1 $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PPGen}(1^\lambda)$
2 $(d, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
3 $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^d)$
4 $b' \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{EncO}(\cdot,\cdot),\textsc{KeyGenO}(\cdot,\cdot)}(st)$
5 **if** $\exists\ i' \le i, j' \le j : g(\langle \mathbf{x}_0^{i'}, \mathbf{y}_0^{j'} \rangle) \neq g(\langle \mathbf{x}_1^{i'}, \mathbf{y}_1^{j'} \rangle)$ **then**
6 **return** $\bot$
7 **else return** $(b = b')$

KEYGENO$(\mathbf{x}_0, \mathbf{x}_1)$:
8 $j \leftarrow j + 1$
9 $(\mathbf{x}_0^j, \mathbf{x}_1^j) \leftarrow (\mathbf{x}_0, \mathbf{x}_1)$
10 $\mathsf{sk} \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x}_b)$
11 **return** sk

ENCO$(\mathbf{y}_0, \mathbf{y}_1)$:
12 $i \leftarrow i + 1$
13 $(\mathbf{y}_0^i, \mathbf{y}_1^i) \leftarrow (\mathbf{y}_0, \mathbf{y}_1)$
14 $\mathsf{ct} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, \mathbf{y}_b)$
15 **return** ct

Fig. 4. Games defining IND-security for FH-IPBFE.

if their inner product is non-zero, and AC-IPBFRE for small tag universe and arbitrary access policies can be generically constructed from IPBFRE.

4.1 From IPBFRE to NZIP-AC-IPBFRE

Here, we present a transformation that converts any FRE for an inner-product based functionality $f(\mathbf{x}, \mathbf{y}) = g(\langle \mathbf{x}, \mathbf{y} \rangle)$ into an access-controlled FRE for the same functionality where tags are vectors (possibly of a different dimension than the message vectors, but over the same field as the message vectors) and evaluation of two ciphertexts is possible if the inner-product of their tags is non-zero. A similar transformation has been considered in [23]. Formally, the message space is $\mathcal{M} = \mathbb{Z}_m^d$. The tag space $\mathcal{T}$ can be set to any subset of $\mathbb{Z}_p^\tau$ that satisfies[6]

$$\forall \mathbf{u}, \mathbf{v} \in \mathcal{T} : \langle \mathbf{u}, \mathbf{v} \rangle \in \{0, 1\}$$

[6] For the inner-product, inner-product in the exponent and orthogonality functionality, this requirement can be relaxed to $\forall \mathbf{u}, \mathbf{v} \in \mathcal{T} : \langle \mathbf{u}, \mathbf{v} \rangle \in \{0\} \cup \mathbb{Z}_m^*$. For the inner-product and inner-product in the exponent functionality, this requires keeping track of the tags and dividing the result (in the exponent) by the inner product of the tags in Eval. Note that for all these functionalities g is multiplicatively homomorphic. Such a scheme however does not hide the tags any more. The tags can be encrypted with an IPFRE scheme to only leak the inner products between the tags. This variant has been considered in [23].

Game $\mathbf{G}^{\text{sim-1}}_{\mathsf{FH\text{-}IPBFE},\mathrm{A}}(\lambda)$

MAIN:

1 $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PPGen}(1^\lambda)$
2 $(d, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
3 $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^d)$
4 $b \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{EncO}(\cdot),\textsc{KeyGenO}(\cdot)}(st)$
5 **return** b

KEYGENO($\mathbf{x}$):

6 $\mathsf{sk} \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$
7 **return** sk

ENCO($\mathbf{y}$):

8 $\mathsf{ct} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, \mathbf{y})$
9 **return** ct

Game $\mathbf{G}^{\text{sim-0}}_{\mathsf{FH\text{-}IPBFE},\mathrm{A},\mathrm{S}}(\lambda)$

MAIN:

1 $\mathcal{C}_{\mathsf{ip}} \leftarrow \emptyset;\ i \leftarrow 0$
2 $E, K \leftarrow \emptyset$
3 $(\mathsf{pp}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}_1(1^\lambda)$
4 $(d, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
5 $st_{\mathrm{S}} \leftarrow\!\!\$\ \mathrm{S}_2(1^d, st_{\mathrm{S}})$
6 $b \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{EncO}(\cdot),\textsc{KeyGenO}(\cdot)}(st_{\mathrm{A}})$
7 **return** b

KEYGENO($\mathbf{x}$):

8 $i \leftarrow i + 1;\ K \leftarrow K \cup \{i\}$
9 $\mathbf{x}_i \leftarrow \mathbf{x}$
10 **for** $j \leq i$ **do**
11 **if** $j \in E$: $c_{i,j} \leftarrow g(\langle \mathbf{x}, \mathbf{y}_j \rangle)$
12 **else** : $c_{i,j} \leftarrow \perp$
13 $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}_{\mathsf{ip}} \cup \{((i,j), c_{i,j})\}$
14 $(\mathsf{sk}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}_4(\mathcal{C}_{\mathsf{ip}}, st_{\mathrm{S}})$
15 **return** sk

ENCO($\mathbf{y}$):

16 $i \leftarrow i + 1;\ E \leftarrow E \cup \{i\}$
17 $\mathbf{y}_i \leftarrow \mathbf{y}$
18 **for** $j \leq i$ **do**
19 **if** $j \in K$: $c_{i,j} \leftarrow g(\langle \mathbf{x}_j, \mathbf{y} \rangle)$
20 **else** : $c_{i,j} \leftarrow \perp$
21 $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}_{\mathsf{ip}} \cup \{((i,j), c_{i,j})\}$
22 $(\mathsf{ct}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}_3(\mathcal{C}_{\mathsf{ip}}, st_{\mathrm{S}})$
23 **return** ct

Fig. 5. Games defining SIM-security for FH-IPBFE.

The auxiliary information for the access policy is $\tau \in \mathbb{N}$ and a description of $\mathcal{T}$ with access policy

$$a_{\mathsf{nzip}}(\mathsf{pp}, \tau, \mathbf{u}, \mathbf{v}) := \begin{cases} 1 & \text{if } \mathbf{u}^\top \mathbf{v} \neq 0 \text{ and } \mathbf{u} \in \mathcal{T} \text{ and } \mathbf{v} \in \mathcal{T} \\ 0 & \text{otherwise.} \end{cases} .$$

Let $\mathsf{IPBFRE} = (\mathsf{PPGen}, \mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval})$ be an FRE scheme with message-space $\mathbb{Z}_p^d$ for an inner-product based functionality $f(\mathbf{x}, \mathbf{y}) = g(\langle \mathbf{x}, \mathbf{y} \rangle)$. Define the associated NZIP-AC-IPBFRE scheme $\mathsf{NZIP\text{-}AC\text{-}IPBFRE}[\mathsf{IPBFRE}] = (\mathsf{PPGen}, \mathsf{Setup}', \mathsf{Enc}', \mathsf{Eval}')$ with message-space $\mathbb{Z}_p^d$, tag-space $\mathbb{Z}_p^\tau$, and access policy a_{nzip} as in Fig. 6.

$\underline{\mathsf{Setup}'(1^\lambda, 1^d, 1^\tau)}$:
1 **return** $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^{d \cdot \tau})$

$\underline{\mathsf{Enc}'(\mathsf{msk}, \mathbf{u} \in \mathbb{Z}_p^\tau, \mathbf{x} \in \mathbb{Z}_p^d)}$:
2 **return** $\mathsf{Enc}(\mathsf{msk}, \mathbf{u} \otimes \mathbf{x})$

$\underline{\mathsf{Eval}'(\mathsf{ct}_1, \mathsf{ct}_2)}$:
3 **return** $\mathsf{Eval}(\mathsf{ct}_1, \mathsf{ct}_2)$

Fig. 6. Transform from FRE for an inner-product based functionality to NZIP-AC-FRE for the same functionality.

Proposition 1. *If* IPBFRE *is evaluation correct, then* NZIP-AC-IPBFRE[IPBFRE] *is evaluation correct.*

Proof. Let $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^\tau$ with $\mathbf{u}^\top \mathbf{v} = 1$, $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{Z}_p^d$, $\mathsf{ct}_1 \in \mathbf{Out}(\mathsf{Enc}(\mathsf{msk}, \mathbf{u} \otimes \mathbf{x}_1))$, and $\mathsf{ct}_2 \in \mathbf{Out}(\mathsf{Enc}(\mathsf{msk}, \mathbf{v} \otimes \mathbf{x}_2))$. Then, by evaluation correctness of IPBFRE, $\mathsf{Eval}(\mathsf{ct}_1, \mathsf{ct}_2)$ outputs $g((\mathbf{u} \otimes \mathbf{x}_1)^\top (\mathbf{v} \otimes \mathbf{x}_2)) = g(\mathbf{u}^\top \mathbf{v} \cdot \mathbf{x}_1^\top \mathbf{x}_2) = g(\mathbf{x}_1^\top \mathbf{x}_2)$.

Theorem 1 (IND-security). *If* IPBFRE *is IND-secure, then* NZIP-AC-IPBFRE[IPBFRE] *is IND-secure. Concretely, for every adversary* A *against IND-security of* NZIP-AC-IPBFRE[IPBFRE], *there exists an adversary* B *against IND-security of* IPBFRE *with roughly the same runtime such that*

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE}[\mathsf{IPBFRE}],\mathrm{A}}(\lambda) \;\leq\; \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{IPBFRE},\mathrm{B}}(\lambda) \; .$$

Proof. Let A be an adversary against IND-security of NZIP-AC-IPBFRE[IPBFRE]. We construct an adversary B against IND-security of IPBFRE in Fig. 7.

It is easy to see that when A plays the $\mathbf{G}^{\text{ind-b}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE}[\mathsf{IPBFRE}],A}(\lambda)$ game, it perfectly simulates the game $\mathbf{G}^{\text{ind-}b}_{\mathsf{IPBFRE},\mathrm{B}}(\lambda)$ for B and outputs the same bit as B.

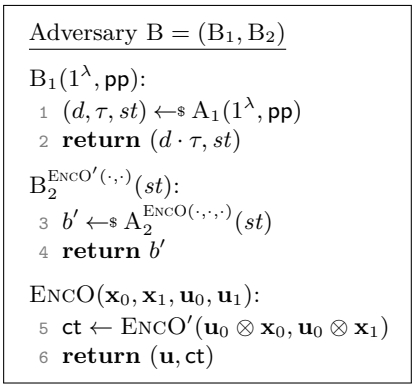


Fig. 7. Adversary against IND-security of IPBFRE.

This shows

$$\begin{aligned}&\Pr[\mathbf{G}^{\text{ind-b}}_{\mathsf{IPBFRE},\mathrm{B}}(\lambda) \Rightarrow 1 \mid \mathbf{G}^{\text{ind-b}}_{\mathsf{IPBFRE},\mathrm{B}}(\lambda) \not\Rightarrow \bot]\\ =\ &\Pr[\mathbf{G}^{\text{ind-b}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE[IPBFRE]},\mathrm{A}}(\lambda) \Rightarrow 1 \mid \mathbf{G}^{\text{ind-b}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE[IPBFRE]},\mathrm{A}}(\lambda) \not\Rightarrow \bot]\ .\end{aligned} \tag{1}$$

To show that the advantage of B is at least as high as the advantage of A, what remains to show is that B causes the game to output $\bot$ only when A does so.

Therefore, let $\mathbf{z}_0^j, \mathbf{z}_1^j, \mathbf{u}_0^j, \mathbf{u}_1^j$ be the input of the j-th $\textsc{Enc}$ oracle query of B_2. The IND game for IPBFRE returns $\bot$ if there are indices j, k with $g(\langle \mathbf{z}_0^j, \mathbf{z}_0^k \rangle) \neq g(\langle \mathbf{z}_1^j, \mathbf{z}_1^k \rangle)$. Using that $\mathbf{z}_b^j = \mathbf{u}_b^j \otimes \mathbf{x}_b^j$, we get

$$\begin{aligned}g(\langle \mathbf{u}_0^j, \mathbf{u}_0^k \rangle \langle \mathbf{x}_0^j, \mathbf{x}_0^k \rangle) &= g(\langle \mathbf{u}_0^j \otimes \mathbf{x}_0^j, \mathbf{u}_0^k \otimes \mathbf{x}_0^k \rangle) = g(\langle \mathbf{z}_0^j, \mathbf{z}_0^k \rangle)\\ &\neq g(\langle \mathbf{z}_1^j, \mathbf{z}_1^k \rangle) = g(\langle \mathbf{u}_1^j \otimes \mathbf{x}_1^j, \mathbf{u}_1^k \otimes \mathbf{x}_1^k \rangle) = g(\langle \mathbf{u}_1^j, \mathbf{u}_1^k \rangle \langle \mathbf{x}_1^j, \mathbf{x}_1^k \rangle)\end{aligned} \tag{2}$$

which implies $\langle \mathbf{u}_0^j, \mathbf{u}_0^k \rangle \neq 0$ or $\langle \mathbf{u}_1^j, \mathbf{u}_1^k \rangle \neq 0$ and thus $\langle \mathbf{u}_0^j, \mathbf{u}_0^k \rangle \neq \langle \mathbf{u}_1^j, \mathbf{u}_1^k \rangle$ or $\langle \mathbf{u}_0^j, \mathbf{u}_0^k \rangle = \langle \mathbf{u}_1^j, \mathbf{u}_1^k \rangle = 1$. In the latter case, we must have $g(\langle \mathbf{x}_0^j, \mathbf{x}_0^k \rangle) \neq g(\langle \mathbf{x}_1^j, \mathbf{x}_1^k \rangle)$ to satisfy inequality (2). This shows, that also the IND game for NZIP-AC-IPBFRE[IPBFRE] outputs $\bot$ here.

With this, we see

$$\Pr[\mathbf{G}^{\text{ind-1}}_{\mathsf{IPBFRE},\mathrm{B}}(\lambda) \Rightarrow \bot] \leq \Pr[\mathbf{G}^{\text{ind-1}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE[IPBFRE]},\mathrm{A}}(\lambda) \Rightarrow \bot]\ . \tag{3}$$

Combining equations (1) and (3) yields the result.

Theorem 2 (SIM-security). *If* IPBFRE *is SIM-secure, then* NZIP-AC-IPBFRE[IPBFRE] *is SIM-secure. Concretely, for every adversary* A *against SIM-security of* NZIP-AC-IPBFRE[IPBFRE]*, there exists an adversary* B *against SIM-security of* IPBFRE *with roughly the same runtime as* A*, such that for every simulator* S_B *there exists a simulator* S_A *with*

$$\mathsf{Adv}^{\text{sim}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE[IPBFRE]},\mathrm{A},\mathrm{S_A}}(\lambda) \ \leq\ \mathsf{Adv}^{\text{sim}}_{\mathsf{IPBFRE},\mathrm{B},\mathrm{S_B}}(\lambda)\ .$$

Proof. Let $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ be an adversary against the SIM-security of NZIP-AC-IPBFRE[IPBFRE]. We first construct an adversary $\mathrm{B} = (\mathrm{B}_1, \mathrm{B}_2)$ against the SIM-security of IPBFRE in Fig. 8. Next, by SIM-security of IPBFRE, for an adversary B there exists a simulator $S_{\mathrm{B}} = (S_{\mathrm{B},1}, S_{\mathrm{B},2}, S_{\mathrm{B},3})$. Then, consider the simulator $S_{\mathrm{A}} = (S_{\mathrm{A},1}, S_{\mathrm{A},2}, S_{\mathrm{A},3})$ for NZIP-AC-IPBFRE[IPBFRE] for an adversary A also given in Fig. 8.

Adversary $\mathrm{B} = (\mathrm{B}_1, \mathrm{B}_2)$

$\mathrm{B}_1(1^\lambda, \mathsf{pp})$:
1 $(d, \tau, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
2 **return** $(d \cdot \tau, st)$

$\mathrm{B}_2^{\textsc{EncO}'(\cdot)}(st)$:
3 $b \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{EncO}(\cdot)}(st)$
4 **return** b

$\textsc{EncO}(\mathbf{x}, \mathbf{u})$:
5 $\mathsf{ct} \leftarrow \textsc{EncO}'(\mathbf{x} \otimes \mathbf{u})$
6 **return** ct

Simulator $S_{\mathrm{A}} = (S_{\mathrm{A},1}, S_{\mathrm{A},2}, S_{\mathrm{A},3})$

$S_{\mathrm{A},1}(1^\lambda)$:
1 $(\mathsf{pp}, st) \leftarrow\!\!\$\ \mathrm{S}_{\mathrm{B},1}(1^\lambda)$
2 **return** (pp, st)

$\mathrm{S}_{\mathrm{A},2}(1^d, 1^\tau, st)$:
3 $st \leftarrow\!\!\$\ \mathrm{S}_{\mathrm{B},2}(1^{d\cdot\tau}, st)$
4 **return** st

$\mathrm{S}_{\mathrm{A},3}(\mathcal{C}_{\mathsf{ip}}, st)$:
5 $\mathcal{C}'_{\mathsf{ip}} \leftarrow \mathcal{C}_{\mathsf{ip}} \cup \{((i,j), g(0)) \mid j \le i \wedge \nexists c_{i,j} : ((i,j), c_{i,j}) \in \mathcal{C}_{\mathsf{ip}}\}$
6 $(\mathsf{ct}, st) \leftarrow\!\!\$\ \mathrm{S}_{\mathrm{B},3}(\mathcal{C}'_{\mathsf{ip}}, st)$
7 **return** (ct, st)

Fig. 8. Adversary against SIM-security of IPBFRE and simulator for SIM security of NZIP-AC-IPBFRE[IPBFRE].

The simulator stage $\mathrm{S}_{A,3}$ gets a set $\mathcal{C}_{\mathsf{ip}}$ of all function evaluations where the corresponding tag vectors satisfy the access policy and adds them to the new set $\mathcal{C}'_{\mathsf{ip}}$. Additionally, for every pair of vectors that has no inner-product value stored in $\mathcal{C}'_{\mathsf{ip}}$ (i.e. pairs of vectors that did not satisfy the access policy), it stores them together with $g(0)$ as inner product value in $\mathcal{C}'_{\mathsf{ip}}$.[7] Since for any vectors $\mathbf{x}, \mathbf{y}$ encrypted under tags $\mathbf{u}, \mathbf{v}$ that do not satisfy the access policy

$$g(\langle \mathbf{x} \otimes \mathbf{u}, \mathbf{y} \otimes \mathbf{v} \rangle) = g(\langle \mathbf{x}, \mathbf{y} \rangle \langle \mathbf{u}, \mathbf{v} \rangle) = g(\langle \mathbf{x}, \mathbf{y} \rangle \cdot 0) = g(0)$$

storing the value $g(0)$ in $\mathcal{C}'_{\mathsf{ip}}$ provides the correct function evaluation for the underlying IPBFRE.

With this, we see

$$\Pr[\mathbf{G}^{\text{sim-1}}_{\text{NZIP-AC-IPBFRE[IPBFRE]},\mathrm{A}}(\lambda) \Rightarrow 1] = \Pr[\mathbf{G}^{\text{sim-1}}_{\text{IPBFRE},\mathrm{B}}(\lambda) \Rightarrow 1]$$

[7] For the relaxed version where the inner product of tags can be in $\{0\} \cup \mathbb{Z}_m^*$ and g is multiplicatively homomorphic the simulator has to compute the set $\mathcal{C}'_{\mathsf{ip}}$ as

$$\mathcal{C}'_{\mathsf{ip}} \leftarrow \{((i,j), c_{i,j} \cdot g((\mathbf{u}_i)^\top \mathbf{u}_j)) \mid ((i,j), c_{i,j}) \in \mathcal{C}_{\mathsf{ip}} \wedge (i, \mathbf{u}_i) \in \mathcal{T} \wedge (j, \mathbf{u}_j) \in \mathcal{T}\} \cup \{((i,j), g(0)) \mid \nexists c_{i,j} : ((i,j), c_{i,j}) \in \mathcal{C}_{\mathsf{ip}}\},$$

where $\mathcal{T}$ contains all tuples of indices and tags, which is an additional input to the simulator in the non-tag hiding variant of the SIM-security game.

and

$$\Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE}[\mathsf{IPBFRE}],\mathrm{A},\mathrm{S_A}}(\lambda) \Rightarrow 1] = \Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{IPBFRE},\mathrm{B},\mathrm{S_B}}(\lambda) \Rightarrow 1] \ .$$

Subtracting yields the result.

4.2 From NZIP-AC-IPBFRE to small universe AC-IPBFRE

We show here how an NZIP-AC-IPBFRE can be used to realize AC-IPFRE for general acceess policies in the small tag-universe case. The construction relies on the following effective version of Lagrange's four square therorem.

Theorem 3. *There exists a PT algorithm* $\mathsf{LagrangeSoS}$ *that inputs an arbitrary natural number* $x \in \mathbb{N}$ *and outputs* (a_1, a_2, a_3, a_4) *such that* $a_1^2 + a_2^2 + a_3^2 + a_4^2 = x$.

Proof. The existence of such a solution follows from Lagrange's four square theorem: Every positive integer is the sum of four squares. Several proofs of this classical theorem can be found for example in [29, §20.5].

Now, Rabin and Shallit presented algorithms that can find such $a_1, \ldots, a_4$ for m with expected time $\mathcal{O}((\log m)^2)$ if the extended Riemann hypothesis is true or $\mathcal{O}((\log m)^2 \log\log m)$ without relying on unproven conjectures [46]. Pollack and Treviño improved the expected runtime to $\mathcal{O}((\log m)^2 (\log\log m)^{-1})$ [45].[8]

AC-IPFRE FOR GENERAL ACCESS POLICIES. We demonstrate how a NZIP-AC-IPFRE can be used to realize any access policy with an a priori fixed, polynomial number of different tags. Let $G = (V, E)$ be an undirected graph (with loops) expressing this access policy. That is, V is the set of tags and $\{u, v\} \in E$ iff it is possible to compute the inner product between ciphertexts with tag u and ciphertexts with tag v with Eval. In particular, the node u has a loop ($\{u\} \in E$) iff it is possible to compute the inner product between two ciphertexts both associated with tag u.

We now construct vectors to realize this access policy with a NZIP-AC-IPFRE. Therefore, let $L := \{e \in E \mid |e| = 1\}$ be the set of loops and fix an ordering $\{v_1, \ldots, v_n\} = V$ on the set of nodes and $\{e_1, \ldots, e_m\} = E \setminus L$ on the set of non-loop edges. The construction uses dimension $\tau := m + 4n$. We define the tag vector $\mathbf{u}_i$ as follows:

[8] Technically, we require worst-case polynomial runtime, which can be obtained by relaxing to a negligible correctness error.

$$\mathbf{u}_i := \begin{pmatrix} b_{i,1} \\ \vdots \\ b_{i,m} \\ 0 \\ \vdots \\ 0 \\ a_{i,1} \\ a_{i,2} \\ a_{i,3} \\ a_{i,4} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} m \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} 4(i-1) \\ \\ \\ \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} 4(n-i) \end{matrix} \quad \text{where} \quad \begin{aligned} b_{i,k} &:= \begin{cases} 1 & \text{if } v_i \in e_k \\ 0 & \text{otherwise} \end{cases} \\ \mu_i &:= \sum_{k=1}^{m} b_{i,k} \\ b_i &:= \begin{cases} 1 & \text{if } \{v_i\} \in E \\ 0 & \text{otherwise} \end{cases} \\ (a_{i,j})_{j\in[4]} &\leftarrow\!\!\$\ \mathsf{LagrangeSoS}(mq - \mu_i + b_i) \end{aligned}$$

The summand mq in the argument of $\mathsf{LagrangeSoS}$ avoids a negative argument and can be replaced by any mutliple of q that is sufficiently large.

The following theorem formalizes that these vectors, when used as tag vectors for an NZIP-AC-IPFRE, realize the access structured represented by the graph G.

Theorem 4. *Let $G = (V, E)$ be an undirected graph (with loops), $V =: \{v_1, \ldots, v_n\}$, and $(\mathbf{u}_i)_{i\in[n]}$ constructed as described above. Then for all $i, j \in [n]$*

$$\{v_i, v_j\} \in E \Rightarrow \langle \mathbf{u}_i, \mathbf{u}_j \rangle = 1$$
$$\{v_i, v_j\} \notin E \Rightarrow \langle \mathbf{u}_i, \mathbf{u}_j \rangle = 0$$

Proof. We first focus on the non-loop edges.

Case 1: Assume $\{v_i, v_j\} \in E$ and $i \neq j$. Then there exists an index $k \in [m]$ such that $e_k = \{v_i, v_j\}$ and thus $b_{i,k} = b_{j,k} = 1$. For all $k' \in [m]$ with $k \neq k'$ we have $b_{i,k'} = 0$ or $b_{j,k'} = 0$, because $e_{k'} \neq \{v_i, v_j\}$. This yields

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \left\langle \begin{pmatrix} b_{i,1} \\ \vdots \\ b_{i,m} \end{pmatrix}, \begin{pmatrix} b_{j,1} \\ \vdots \\ b_{j,m} \end{pmatrix} \right\rangle = 1 \,.$$

Case 2: Assume $\{v_i, v_j\} \notin E$ and $i \neq j$. Then for all $k \in [m]$ we have $b_{i,k} = 0$ or $b_{j,k} = 0$, because $e_k \neq \{v_i, v_j\}$. This yields

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \left\langle \begin{pmatrix} b_{i,1} \\ \vdots \\ b_{i,m} \end{pmatrix}, \begin{pmatrix} b_{j,1} \\ \vdots \\ b_{j,m} \end{pmatrix} \right\rangle = 0 \,.$$

Next, we consider the loops. For this, note that

$$\langle \mathbf{u}_i, \mathbf{u}_i \rangle = \left\langle \begin{pmatrix} b_{i,1} \\ \vdots \\ b_{i,m} \\ 0 \\ \vdots \\ 0 \\ a_{i,1} \\ a_{i,2} \\ a_{i,3} \\ a_{i,4} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} b_{i,1} \\ \vdots \\ b_{i,m} \\ 0 \\ \vdots \\ 0 \\ a_{i,1} \\ a_{i,2} \\ a_{i,3} \\ a_{i,4} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \right\rangle = \mu_i + a_1^2 + a_2^2 + a_3^2 + a_4^2 = b_i \ .$$

Case 3: Assume $\{v_i\} \in E$. Then by definition $b_i = 1$ and thus $\langle \mathbf{u}_i, \mathbf{u}_i \rangle = 1$.

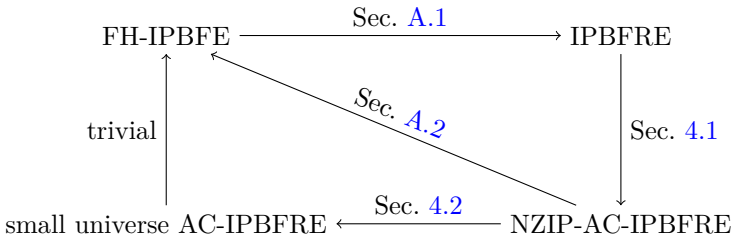
Case 4: Assume $\{v_i\} \notin E$. Then by definition $b_i = 0$ and thus $\langle \mathbf{u}_i, \mathbf{u}_i \rangle = 0$.

4.3 Equivalence of FH-IPBFE and IPBFRE

Our construction of general AC-IPBFRE from NZIP-AC-IPBFRE in Section 4.2 can be directly used as FH-IPBFE by using a universe with two tags. FH-IBPFE ciphertexts are then AC-IPBFRE ciphertexts with one tag and FH-IPFRE functional keys are AC-IPBFRE ciphertexts with the other tags. We given an optimized variant of this idea in Section A.2

Next, we describe how an an FH-IPBFE can be generically transformed to an IPBFRE. Our transformation simply includes both a ciphertext and function key of the FH-IPBFE in an IPBFRE ciphertext and is provided in the Appendix A.1.

Thus, we have showen the following implications that give us the equivalence between FH-IPBFRE, IPBFRE, NZIP-AC-IPBFRE and AC-IBFRE with small tag universe:



5 The TinyFHIPFE Scheme

Here we give a construction of FH-IPFE that borrows some techniques of Wee [50] in the context of (non-function-hiding) quadratic FE and is more efficient than prior constructions.

Bilinear groups. We consider asymmetric bilinear groups. An asymmetric *bilinear-group generator* $\mathcal{G}$ takes as input 1^λ and outputs a tuple $(p, g_1, g_2, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T, \mathsf{e})$. Here, $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T$ are groups of order p, where p is a prime of λ bits, and g_1, g_2 generate $\mathsf{G}_1, \mathsf{G}_2$, respectively. The mapping $\mathsf{e}\colon \mathsf{G}_1 \times \mathsf{G}_2 \to \mathsf{G}_T$ is an efficiently computable, non-degenerate bilinear map. In particular, it satisfies $\mathsf{e}(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$, and $\mathsf{e}(g_1, g_2)$ generates G_T.

For a group G of order p generated by g, for $x \in \mathbb{Z}_p$ we write g^x as $[x]_g$ We omit the subscript g when it is evident from the context. For brevity, we also write $\prod_{i\in[d]} \mathsf{e}(\mathbf{a}[i], \mathbf{b}[i])$ as $\mathsf{e}(\mathbf{a}, \mathbf{b})$, where $\mathbf{a} \in \mathsf{G}_1^d$ and $\mathbf{b} \in \mathsf{G}_2^d$.

Bounded IPFE. To capture ours and prior FH-IPFE constructions where inner-product is recovered in the exponent of the target group, we introduce the notion of *bounded* IPFE. In bounded IPFE, the Eval algorithm takes an additional input 1^B for a bound $B \in \mathbb{N}$ and uses an algorithm Rec that on input $(1^B, \mathsf{G}_T, g_T, [z]_{g_T})$ outputs $m \in \mathbb{Z}_p$ such that $m = \mathsf{DLog}_{\mathsf{G}_T, g_T}([z]_{g_T})$ if $m \le B$ or $\perp$. Evaluation correctness requires that

$$\Pr[\mathsf{Eval}(1^B, \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x}), \mathsf{Enc}(\mathsf{msk}, \mathbf{y})) = \langle \mathbf{x}, \mathbf{y} \rangle] = 1$$

for all $B, \lambda, d \in \mathbb{N}$, all $\mathsf{pp} \in \mathbf{Out}(\mathsf{PPGen}(1^\lambda))$ and all $\mathsf{msk} \in \mathbf{Out}(\mathsf{Setup}(\mathsf{pp}, 1^d))$, where $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_m^d$ is such that $\langle \mathbf{x}, \mathbf{y} \rangle \le B$. The probability is taken over the coins of the Enc algorithm.

5.1 Our Construction

Let $\mathcal{G}_{\mathsf{asym}}$ be an asymmetric bilinear-group generator and $\mathbf{SL}_4(\mathbb{Z}_p)$ be the special linear group of degree 4 over $\mathbb{Z}_p$. We define our bounded FH-IPFE scheme named $\mathsf{TinyFHIPFE}$ in Fig. 9. (Recall pp is implicitly input to the remaining algorithms.) Note that, like previous FH-IPFE schemes, it recovers inner-product in the exponent. We find bounded inner-product is sufficient for the datasets considered in Appendix 7. $\mathsf{TinyFHIPFE}$ uses a short master secret key linear in the input vector dimension and a uniform random vector is sampled during each encryption. In order to reduce the size of the ciphertext, aside from the linear components encoding the input vector, other components of the ciphertext are carefully chosen to make the cross terms constant size.

Proposition 2. $\mathsf{TinyFHIPFE}$ *is evaluation correct.*

Proof. Let $\lambda, d \in \mathbb{N}$. Let $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{TinyFHIPFE.PPGen}(1^\lambda)$, $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{TinyFHIPFE.Setup}(1^\lambda, 1^d)$ $\mathsf{ct} \leftarrow \mathsf{TinyFHIPFE.Enc}(\mathsf{msk}, \mathbf{y})$ and $\mathsf{sk} \leftarrow \mathsf{TinyFHIPFE.KeyGen}(\mathsf{msk}, \mathbf{x})$ for $\mathbf{x}, \mathbf{y} \in$

$\underline{\mathsf{PPGen}(1^\lambda)}$:

$(p, g_1, g_2, g_T, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T, \mathsf{e}) \leftarrow\!\!\$\ \mathcal{G}_{\mathsf{asym}}(1^\lambda)$
$\mathsf{pp} \leftarrow (p, g_1, g_2, g_t, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T, \mathsf{e})$
return pp

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

$\mathbf{B} \leftarrow\!\!\$\ \mathbf{SL}_4(\mathbb{Z}_p)$
$\mathbf{B}^* \leftarrow (\mathbf{B}^{-1})^T$
$\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_p^{2\times d}$
$\mathsf{msk} \leftarrow (\mathbf{A}, \mathbf{B}, \mathbf{B}^*)$
return msk

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})}$:

$\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_p^2$
$(\mathbf{A}, \mathbf{B}, \mathbf{B}^*) \leftarrow \mathsf{msk}$
$\mathsf{sk}_\mathbf{x} \leftarrow [\mathbf{A}^T\mathbf{s} + \mathbf{x}]_{g_1}$
$\mathsf{sk}_\mathbf{k} \leftarrow [\mathbf{B} \cdot (\mathbf{s} \| \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{A}^T\mathbf{s})]_{g_1}$
$\mathsf{sk} \leftarrow (\mathsf{sk}_\mathbf{x}, \mathsf{sk}_\mathbf{k})$
return sk

$\underline{\mathsf{Enc}(\mathsf{msk}, \mathbf{y})}$:

$\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_p^2$
$(\mathbf{A}, \mathbf{B}, \mathbf{B}^*) \leftarrow \mathsf{msk}$
$\mathsf{ct}_\mathbf{y} \leftarrow [\mathbf{A}^T\mathbf{s} + \mathbf{y}]_{g_2}$
$\mathsf{ct}_\mathbf{c} \leftarrow [\mathbf{B}^* \cdot (\mathbf{A}\mathbf{y} \| \mathbf{s})]_{g_2}$
$\mathsf{ct} \leftarrow (\mathsf{ct}_\mathbf{y}, \mathsf{ct}_\mathbf{c})$
return ct

$\underline{\mathsf{Eval}(1^B, \mathsf{sk}, \mathsf{ct})}$:

$(\mathsf{sk}_\mathbf{x}, \mathsf{sk}_\mathbf{k}) \leftarrow \mathsf{sk}$
$(\mathsf{ct}_\mathbf{y}, \mathsf{ct}_\mathbf{c}) \leftarrow \mathsf{ct}$
$[z]_{g_T} \leftarrow \mathsf{e}(\mathsf{sk}_\mathbf{x}, \mathsf{ct}_\mathbf{y}) / \mathsf{e}(\mathsf{sk}_\mathbf{k}, \mathsf{ct}_\mathbf{c})$
$z \leftarrow \mathsf{Rec}(1^B, \mathsf{G}_T, g_T, [z]_{g_T})$
return z

Fig. 9. The TinyFHIPFE Construction.

$\mathbb{Z}_p^d$, we have $(\mathsf{ct}_\mathbf{y}, \mathsf{ct}_\mathbf{c}) \leftarrow \mathsf{ct}$ and $(\mathsf{sk}_\mathbf{x}, \mathsf{sk}_\mathbf{k}) \leftarrow \mathsf{sk}$, then

$$\begin{aligned}
\mathsf{e}(\mathsf{sk}_\mathbf{x}, \mathsf{ct}_\mathbf{y}) &= e(g_1, g_2)^{\langle \mathbf{A}^T\mathbf{s}_1 + \mathbf{x}, \mathbf{A}^T\mathbf{s}_2 + \mathbf{y} \rangle} \\
&= [\langle \mathbf{A}^T\mathbf{s}_1 + \mathbf{x}, \mathbf{A}^T\mathbf{s}_2 + \mathbf{y} \rangle]_{g_T} \\
&= [\langle \mathbf{A}^T\mathbf{s}_1, \mathbf{A}^T\mathbf{s}_2 \rangle + \langle \mathbf{A}^T\mathbf{s}_1, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{A}^T\mathbf{s}_2 \rangle + \langle \mathbf{x}, \mathbf{y} \rangle]_{g_T} \\
&= [\mathbf{s}_1^T\mathbf{A}\mathbf{A}^T\mathbf{s}_2 + \mathbf{s}_1^T\mathbf{A}\mathbf{y} + \mathbf{x}^T\mathbf{A}^T\mathbf{s}_2 + \langle \mathbf{x}, \mathbf{y} \rangle]_{g_T} \ .
\end{aligned}$$

Similarly, we have that:

$$\begin{aligned}
\mathsf{e}(\mathsf{sk}_\mathbf{k}, \mathsf{ct}_\mathbf{c}) &= [\langle \mathbf{B} \cdot (\mathbf{s}_1 \| \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{A}^T\mathbf{s}_1),\ \mathbf{B}^* \cdot (\mathbf{A}\mathbf{y} \| \mathbf{s}_2) \rangle]_{g_T} \\
&= [(\mathbf{s}_1^T\mathbf{A}\mathbf{y} + \mathbf{x}^T\mathbf{A}^T\mathbf{s}_2 + \mathbf{s}_1^T\mathbf{A}\mathbf{A}^T\mathbf{s}_2)]_{g_T} \ .
\end{aligned}$$

Thus, $[z]_{g_T} = \mathsf{e}(\mathsf{sk}_\mathbf{x}, \mathsf{ct}_\mathbf{y}) / e(\mathsf{sk}_\mathbf{k}, \mathsf{ct}_\mathbf{c}) = g_T^{\langle \mathbf{x}, \mathbf{y} \rangle}$. Finally, by definition of Rec, $\mathsf{Rec}(1^B, \mathsf{G}_T, g_T, [z]_{g_T})$ outputs $\langle \mathbf{x}, \mathbf{y} \rangle$ when $\langle \mathbf{x}, \mathbf{y} \rangle \leq B$.

Remark 3. We can also construct a similarly structured IPFRE scheme by letting each ciphertext contain three components all in a symmetric group source: $\mathsf{ct}_\mathbf{x}, \mathsf{sk}_\mathbf{k}, \mathsf{ct}_\mathbf{c}$. Correctness of the scheme would follow similar to the correctness of TinyFHIPFE. Here each ciphertext has $d + 8$ group elements.

Security. We show SIM-security of TinyFHIPFE in the generic bilinear group model. Note that multi-message SIM-security, even for non-FH IPFE, is impossible [6], so our reliance on idealized models to show SIM-security, which is

the most meaningful security notion in practice, is essential here. Prior work on FH-IPFE by Kim *et al.* [36] similarly analyze their FH-IPFE scheme in the generic bilinear group model (but do not give an explicit advantage bound).

Theorem 5. TinyFHIPFE *is simulation-secure in the generic bilinear group model. In particular, for every adversary* A *that makes* q_t *total queries to its oracles, there exists a simulator* S *such that*

$$\mathbf{Adv}^{\text{sim-ggm}}_{\mathsf{IPFRE},\mathrm{A},\mathrm{S}}(\lambda) \leq \frac{(4q_t(d+4))^2}{p} \ .$$

The description of the bilinear generic group model and the details of the proof are provided in Appendix B.

6 Outsourced Database Protocols for ANN Search

In this Section, first we present a generic construction of an AC-IPFRE for the access policy a_{ann}. Then, we present our ODB protocol for ANN that uses this AC-IPFRE. The formal definitions of an ANN data structure and an outsourced database protocol are in the Appendix C along with the security proofs of AC-IPFRE[FH-IPFE] and ANN-ODB[AC-IPFRE].

6.1 The AC-IPFRE[FH-IPFE] scheme

We give a transform that lifts any IPFRE to an AC-IPFRE for the access policy a_{ann}. This transform is tailored to a_{ann} and is more efficient than going through the general transform from IPFRE to small tag universe AC-IPFRE in Section 4. We discuss the choice of the access policy function in the introduction, so here we directly present the function a_{ann} with tag-space $\{\mathsf{data}, \mathsf{query}, \mathsf{update}\}$. (The first two arguments of the function namely pp and aux are ignored.):

$$a_{\mathsf{ann}}(\mathsf{tag}_1, \mathsf{tag}_2) = \begin{cases} 0 & \text{if } \mathsf{tag}_1 = \mathsf{tag}_2 = \mathsf{data}, \\ 0 & \text{if } \mathsf{tag}_1 = \mathsf{tag}_2 = \mathsf{query}, \\ 1 & \text{otherwise.} \end{cases}$$

Our construction. Our construction gives an AC-IPFRE for the access policy a_{ann} defined based on any FH-IPFE scheme. In particular, to allow ciphertexts whose tag is update, we include both the ciphertext and function key for the input vector. Define an AC-IPFRE scheme AC-IPFRE[FH-IPFE] as in Fig. 10.

Correctness. Evaluation of inner product requires evaluating a $\mathsf{sk}_{\mathbf{x}}$ component from one ciphertext with a $\mathsf{ct}_{\mathbf{x}}$ component from another ciphertext respectively. We can see by observing the cases in lines 18-21 that the inner product evaluation is possible for each case where the access function allows it. Correctness of lines 14 and 16 follow from evaluation correctness of the underlying FH-IPFE scheme.

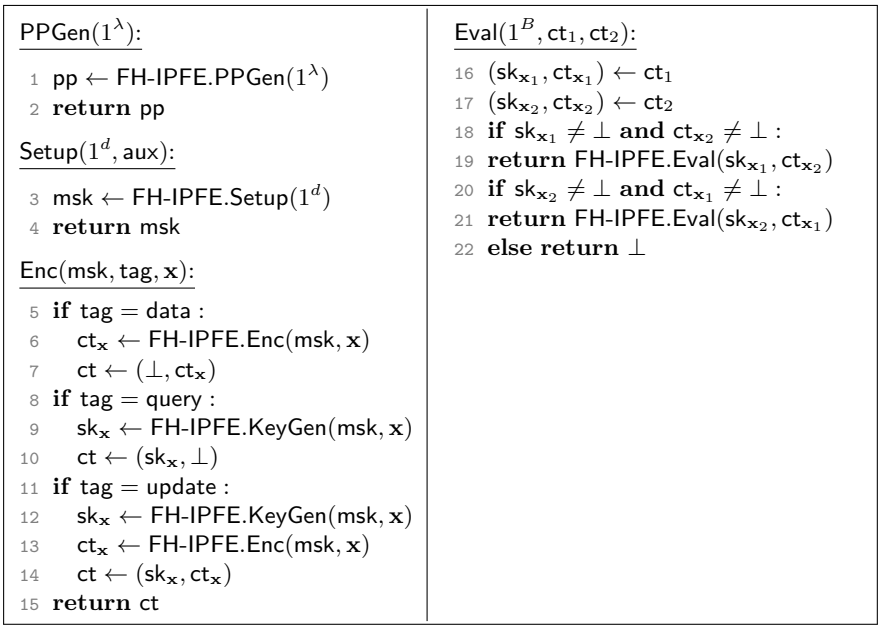


Fig. 10. The AC-IPFRE[FH-IPFE] construction.

Remark 4. Alternatively, one can construct an IPFRE scheme as discussed in Remark 3 and use it to construct an AC-IPFRE by omitting one of the components. More precisely, an IPFRE ciphertext contains three components: $\mathsf{ct}_\mathbf{x}, \mathsf{ct}_\mathbf{c}, \mathsf{ct}_\mathbf{k}$. An AC-IPFRE ciphertext omits the components $\mathsf{ct}_\mathbf{c}$ and $\mathsf{ct}_\mathbf{k}$ for ciphertext with tags data and query respectively and keeps the ciphertext as is for tag update. While the ciphertexts have smaller size with respect to the number of group elements, we do not present this construction due to the difference in performance of symmetric and asymmetric groups for the same security parameter.

Security. Security of AC-IPFRE[FH-IPFE] follows generically from the security of the underlying FH-IPFE. This security proof is given in the Appendix C.3.

6.2 Our ANN ODB Protocol

Here, we present our ANN ODB protocol that uses an AC-IPFRE for the access policy a_{ann}. The communication flow of the protocol is given in Fig. 11 and the algorithms of the protocol are given in Fig. 12. Note that the user and the server use different version of the comparison oracles because of the difference in their databases.

Security. First we give the leakage profile of the protocol ANN-ODB[AC-IPFRE] in Fig. 13.

Algorithm 1

Client U		**Server** S
$(\mathcal{DS}_{\mathsf{ANN}}, \mathcal{D}_S) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^n, \mathcal{D})$	$\xrightarrow{(\mathcal{DS}_{\mathsf{ANN}}, \mathcal{D}_S)}$	
$\mathsf{ct}_{\mathbf{q}} \leftarrow\!\!\$\ \mathsf{PreQuery}(\mathbf{q}, \mathsf{ID}_{\mathbf{q}})$	$\xrightarrow{\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}_{\mathbf{q}}}$	
	$\xleftarrow{\mathsf{ID}}$	$\mathsf{ID} \leftarrow \mathsf{Query}(\mathcal{DS}_{\mathsf{ANN}}, (\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}_{\mathbf{q}}))$
$\mathsf{ct}_{\mathbf{u}} \leftarrow\!\!\$\ \mathsf{PreUpdate}(\mathbf{u}, \mathsf{ID}_{\mathbf{u}})$	$\xrightarrow{\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}_{\mathbf{u}}}$	
		$\mathsf{Update}(\mathcal{DS}_{\mathsf{ANN}}, (\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}_{\mathbf{u}}))$
..... Oracles		
$\textsc{CompO}_U(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{ID}_3)$		$\textsc{CompO}_S(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{ID}_3)$

Fig. 11. Communication flow of ANN ODB protocol.

ANN-ODB[AC-IPFRE] Protocol

$\mathsf{Setup}(1^\lambda, 1^d, 1^n, \mathcal{D}_{\mathsf{init}})$:
1 $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{AC\text{-}IPFRE.PPGen}(1^\lambda)$
2 $\mathsf{msk} \leftarrow \mathsf{AC\text{-}IPFRE.Setup}(1^\lambda, 1^d)$
3 $\mathsf{DS}_{\mathsf{ANN}}.\mathsf{Initialize}(p, d, \mathsf{params})$
4 $\mathcal{D} \leftarrow\!\!\$\ \mathsf{Shuffle}(\mathcal{D}_{\mathsf{init}})$
5 $\{(\mathbf{x}_i, \mathsf{ID}_i)\}_{i\in[n]} \leftarrow \mathcal{D}$
6 **for** $i \in [n]$:
7 $\quad \mathcal{DS}_{\mathsf{ANN}}.\mathsf{Insert}^{\textsc{CompO}(\cdot,\cdot,\cdot)}(\mathsf{ID}_i)$
8 $\quad \mathsf{ct}_i \leftarrow\!\!\$\ \mathsf{AC\text{-}IPFRE.Enc}(\mathsf{msk}, \mathsf{data}, \mathcal{D}[\mathsf{ID}_i])$
9 $\quad \mathcal{D}_S[\mathsf{ID}_i] \leftarrow \mathsf{ct}_i$
10 **return** $(\mathsf{DS}_{\mathsf{ANN}}, \mathcal{D}_S)$

$\mathsf{PreQuery}(\mathbf{q}, \mathsf{ID}_{\mathbf{q}})$:
11 $\mathsf{ct}_{\mathbf{q}} \leftarrow\!\!\$\ \mathsf{AC\text{-}IPFRE.Enc}(\mathsf{msk}, \mathsf{query}, \mathbf{q})$
12 **return** $(\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}_{\mathbf{q}})$

$\mathsf{Query}(\mathsf{DS}_{\mathsf{ANN}}, (\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}_{\mathbf{q}}))$:
13 $\mathsf{ID} \leftarrow \mathsf{DS}_{\mathsf{ANN}}.\mathsf{Search}^{\textsc{CompO}(\cdot,\cdot,\cdot)}(\mathsf{ID}_{\mathbf{q}})$
14 **return** ID

$\mathsf{PreUpdate}(\mathbf{u}, \mathsf{ID}_{\mathbf{u}})$:
15 $\mathsf{ct}_{\mathbf{u}} \leftarrow\!\!\$\ \mathsf{AC\text{-}IPFRE.Enc}(\mathsf{msk}, \mathsf{update}, \mathbf{u})$
16 **return** $(\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}_{\mathbf{u}})$

$\mathsf{Update}(\mathsf{DS}_{\mathsf{ANN}}, (\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}_{\mathbf{u}}))$:
17 $\mathsf{DS}_{\mathsf{ANN}}.\mathsf{Insert}^{\textsc{CompO}(\cdot,\cdot,\cdot)}(\mathsf{ID}_{\mathbf{u}})$
18 **return** $\perp$

$\textsc{CompO}_U(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{ID}_3)$:
19 **for** $i \in [3]$:
20 $\quad \mathbf{x}_i \leftarrow \mathcal{D}[\mathsf{ID}_i]$
21 **if** $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle > \langle \mathbf{x}_2, \mathbf{x}_3 \rangle$: **return** 1
22 **else** : **return** 0

$\textsc{CompO}_S(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{ID}_3)$:
23 **for** $i \in [3]$:
24 $\quad \mathsf{ct}_i \leftarrow \mathcal{D}_S[\mathsf{ID}_i]$
25 $\mathsf{ip}_1 \leftarrow \mathsf{AC\text{-}IPFRE.Eval}(\mathsf{ct}_1, \mathsf{ct}_2)$
26 $\mathsf{ip}_2 \leftarrow \mathsf{AC\text{-}IPFRE.Eval}(\mathsf{ct}_1, \mathsf{ct}_3)$
27 **if** $\mathsf{ip}_1 > \mathsf{ip}_2$: **return** 1
28 **else** : **return** 0

$\mathsf{Shuffle}(\mathcal{D}_{\mathsf{old}})$:
29 $\mathcal{D}_{\mathsf{new}} \leftarrow \emptyset$
30 $n \leftarrow \mathsf{length}(\mathcal{D}_{\mathsf{old}})$
31 $\sigma \leftarrow\!\!\$\ \mathsf{S}_n$
32 **for** $i \in [n]$:
33 $\quad \mathcal{D}_{\mathsf{new}}[\mathsf{ID}_i] \leftarrow \mathcal{D}_{\mathsf{old}}[\mathsf{ID}_{\sigma(i)}]$
34 **return** $\mathcal{D}_{\mathsf{new}}$

Fig. 12. ANN-ODB[AC-IPFRE] protocol.

Leakage Profile LP

$\mathcal{L}_{\mathsf{Setup}}(\mathcal{D}, \mathsf{insert\text{-}order})$:
for $i, j, k \in [n]$:
  $\mathbf{x}_i \leftarrow \mathcal{D}[\mathsf{ID}_i]$; $\mathbf{x}_j \leftarrow \mathcal{D}[\mathsf{ID}_j]$; $\mathbf{x}_k \leftarrow \mathcal{D}[\mathsf{ID}_k]$
  if $\langle \mathbf{x}_i, \mathbf{x}_j \rangle > \langle \mathbf{x}_i, \mathbf{x}_k \rangle$:
    $\mathsf{ip\text{-}comp}[\mathsf{ID}_i][\mathsf{ID}_j][\mathsf{ID}_k] \leftarrow 1$
  else :
    $\mathsf{ip\text{-}comp}[\mathsf{ID}_i][\mathsf{ID}_j][\mathsf{ID}_k] \leftarrow 0$
if $\mathsf{insert\text{-}order} = \perp$:
  $\mathsf{insert\text{-}order} \leftarrow \{1, \ldots, n\}$
$\ell_{\mathsf{Setup}} \leftarrow \mathsf{ip\text{-}comp} \cup \mathsf{insert\text{-}order}$
return ℓ_{Setup}

$\mathcal{L}_{\mathsf{Query}}(\mathbf{q}, \mathcal{D}, \mathcal{Q})$:
$((\mathsf{ID}_1, \mathbf{x}_1), \ldots, (\mathsf{ID}_n, \mathbf{x}_n)) \leftarrow \mathcal{D}$
for $i \in [n]$:
  $\mathsf{ip\text{-}val}[\mathsf{ID}_i] \leftarrow \langle \mathbf{q}, \mathcal{D}[\mathsf{ID}_i] \rangle$
$\ell_{\mathsf{Query}} \leftarrow \mathsf{ip\text{-}val}$
return ℓ_{Query}

$\mathcal{L}_{\mathsf{Update}}(\mathbf{u}, \mathcal{D}, \mathcal{Q})$:
$((\mathsf{ID}_1, \mathbf{x}_1), \ldots, (\mathsf{ID}_n, \mathbf{x}_n)) \leftarrow \mathcal{D}$
$((\mathsf{ID}_{n+1}, \mathbf{q}_1), \ldots, (\mathsf{ID}_{n+m}, \mathbf{q}_m)) \leftarrow \mathcal{Q}$
for $i \in [n]$:
  $\mathsf{ip\text{-}val}[\mathsf{ID}_i] \leftarrow \langle \mathbf{u}, \mathbf{x}_i \rangle$
for $i \in [m]$:
  $\mathsf{ip\text{-}val}[\mathsf{ID}_{n+i}] \leftarrow \langle \mathbf{u}, \mathbf{q}_i \rangle$
$\ell_{\mathsf{Update}} \leftarrow \mathsf{ip\text{-}val}$
return ℓ_{Update}

Fig. 13. Leakage Profile of ANN-ODB[AC-IPFRE].

Theorem 6. *Our ANN-ODB protocol* ANN-ODB[AC-IPFRE] *is SIM-secure under the leakage profile* $\mathsf{LP} = (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Query}}, \mathcal{L}_{\mathsf{Update}})$ *given in Fig. 13, given that* AC-IPFRE *is a SIM-secure for access policy* a_{ann}.

The proof follows easily from SIM-security of the underlying AC-IPFRE and has been deferred to the Appendix C.4.

7 Experimental Evaluation

In this Section, we first evaluate the performance of the proposed TinyFHIPFE scheme and compare it with prior schemes [36,37] to demonstrate its computational efficiency. We then implement the proposed ANN-ODB protocol, initiated by AC-IPFRE[TinyFHIPFE], to illustrate its practicality. For simplicity, we denote this protocol as Π in the following discussion.

Implementation setup. For the implementation of the underlying bilinear pairing group, we utilized the Relic library [8], and configured the Curve25519 [13]. Our experiments were conducted on a laptop equipped with an Intel I7-9750H CPU and 16 GB of memory, running Ubuntu 22.04.

7.1 FH-IPFE Benchmarks

We benchmark the proposed TinyFHIPFE against FH-IPFE schemes proposed in [36,37], denoted as KLM$^+$ and KKS, respectively. We choose KLM$^+$ as a comparison target due to its compact ciphertext length and efficient Eval algorithm. However, its Setup algorithm is slow because it computes the inverse of a matrix of size $d \times d$, which takes time $\mathcal{O}(d^3)$. KKS attempt to address this issue by using $\mathcal{O}(d)$ master secret key but their ciphertexts and function keys are longer than both KLM$^+$ and TinyFHIPFE. As illustrated in Fig. 14, the proposed TinyFHIPFE exhibits even better performance. Our Setup time is almost always negligible, involving sampling of only $2d+16$ random elements from $\mathbb{Z}_p$ and a constant-time operation of finding the inverse of a 4×4 matrix.

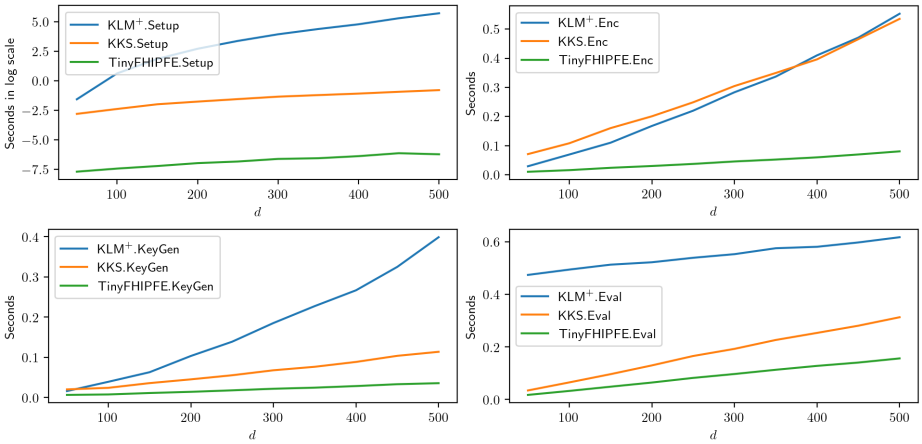


Fig. 14. Benchmarking FH-IPFE schemes.

The proposed TinyIPFRE also outperforms in Enc and KeyGen operations. While KLM$^+$'s Enc and KeyGen involve d^2 multiplications in $\mathbb{Z}_p$, which are cheaper than group multiplication and exponentiation, their performance diminishes as plaintext length increases. For example, for plaintexts of length 500, KLM$^+$.Enc runs roughly 7.5x slower than TinyFHIPFE.Enc. KKS performs a linear number of operations for both Enc and KeyGen, yet they have worse runtime because they perform more expensive group exponentiation. Namely, on average, KKS.Enc is 6.5x slower than TinyFHIPFE.Enc, and KKS.KeyGen is 2.6x slower than TinyFHIPFE.KeyGen.

To benchmark performance of the Eval operation, we bound the resulting inner-product by 10,000. The results presented here assume *preprocessing*, wherein

a lookup table is constructed for TinyFHIPFE and KKS that allows solving the discrete logarithm to a fixed based in constant time. Notably, KLM^+ does not support such preprocessing because its evaluation algorithm does not solve discrete logarithm with respect to a *fixed base*. In this case, KKS.Eval runs approximately 2x than TinyFHIPFE.Eval, with KLM^+.Eval falling significantly further behind. Overall, we fine that TinyFHIPFE exhibits significantly better performance than prior schemes, particularly for long plaintexts.

7.2 ANN-ODB Evaluation

In the following evaluation, we conduct experiments using the following datasets: SIFT [31], MNIST [39], NYTimes [42], and GloVe [44]. These datasets are commonly used for benchmarking ANN algorithms [9] and their details are shown in Table 2. The "max inner product" is the maximum value of all pairwise inner products of vectors from the dataset. This value is useful later when we build lookup tables for solving discrete log problem in constant time.

	SIFT	MNIST	NYTimes	GloVe
Dimension	128	784	256	100
Train Size	1,000,000	60,000	290,000	1,183,514
Distance Metric	Euclidean	Euclidean	Angular	Angular
Max Inner Product	24649	65025	74540	6326734

Table 2. Overview of datasets used in evaluation.

Choice of ANN data structure. Recall that we require the underlying ANN data structure to be inner product based. Fortunately, this is the case for the state-of-the-art Hierarchical Navigable Small Worlds (HNSW) algorithm [41,9]. In the evaluation below we use the following parameters to setup the HNSW algorithm: $M = 32$, where M is the number of connections made for each new vertex during construction, $efConstruction = 30$ where *efConstruction* is the number of candidate neighbors explored during construction, and $efSearch = 256$, where *efSearch* is the number of candidate neighbors explored during search.

Evaluation Results. We present the evaluation results of Π in Table 3. We divide the setup oracle into three phases: the index phase, the encryption phase, and the preprocessing phase. In the index phase, the user indexes the plaintext data vectors using the HNSW algorithm, which is independent of the choice of the underlying encryption scheme. The preprocessing time is time needed to build the lookup table. The query time evaluation encompasses both the time taken by the client U to encrypt the query and the time it takes for the server to respond. Note that we omit network latency in this evaluation since there is only one round of communication involved. To determine the query time, we index the entire training set and then select 50 vectors from the testing set to query,

		SIFT	MNIST	NYTimes	GloVe
Setup	Index	315.2s	48.6s	144.7s	362.9s
	Encryption	55m	4m	17m	68m
	Preprocessing	8.1s	23.3s	25.7s	2111.2s
Query		8.5s	23.5s	11.4s	7.3s
Update		7.6s	22.2s	10.1s	6.6s

Table 3. Evaluation of our ANN-ODB protocol.

computing the average time of querying one vector. Similarly, the update protocol evaluates the time taken by the client U to encrypt an update query and the time it takes for the server to add it to the index. In this scenario, we index the first 90% of the training dataset and add 50 more vectors from the training set to the index, calculating the average time of updating one datapoint. Notably, during both the query and update phases, the client remains extremely lightweight, with execution times of only 20ms and 38ms, respectively. Additionally, the vast majority of the computational workload is handled on the server side.

Comparison to other systems. Boldyreva and Tang [18] present a protocol that operates in the same setting as ours. They present a construction based on locality-sensitive hashing, symmetric encryption, and an oblivious map, which achieves strong security, in particular hiding access pattern. However, their scheme significantly less practical than ours. In particular, running a search query on the MNIST dataset involves an interactive phase between the client and the server, where at least 11 calls to their most expensive component are made, lasting over 8 minutes in duration. Also note they achieve a 95% recall rate on SIFT, whereas we always achieve the baseline recall rate, here 98.3%.

Acknowledgements

We thank Hamed Zamani for helpful conversations about recent applications of approximate nearest-neighbor search.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz [34], pp. 733–751. https://doi.org/10.1007/978-3-662-46447-2_33 7
2. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, Part III. Lecture Notes in Computer Science, vol. 12493, pp. 467–497. Springer, Cham, Switzerland, Daejeon, South Korea (Dec 7–11, 2020). https://doi.org/10.1007/978-3-030-64840-4_16 2, 10

3. Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., Sahai, A.: On the practical security of inner product functional encryption. In: Katz [34], pp. 777–798. https://doi.org/10.1007/978-3-662-46447-2_35 3
4. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021, Part IV. Lecture Notes in Computer Science, vol. 12828, pp. 208–238. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84259-8_8 2
5. Agrawal, S., Goyal, R., Tomida, J.: Multi-party functional encryption. In: Nissim, K., Waters, B. (eds.) TCC 2021: 19th Theory of Cryptography Conference, Part II. Lecture Notes in Computer Science, vol. 13043, pp. 224–255. Springer, Cham, Switzerland, Raleigh, NC, USA (Nov 8–11, 2021). https://doi.org/10.1007/978-3-030-90453-1_8 2
6. Agrawal, S., Libert, B., Maitra, M., Titiu, R.: Adaptive simulation security for inner product functional encryption. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 12110, pp. 34–64. Springer, Cham, Switzerland, Edinburgh, UK (May 4–7, 2020). https://doi.org/10.1007/978-3-030-45374-9_2 23
7. Aladow, N.S.: On the ditribution of quadratic residues and nonresidues of a prime P in the series $1, 2, \ldots, P-1$. Matematicheskii Sbornik **18**(1), 61–75 (1896), http://mi.mathnet.ru/eng/msb/v18/i1/p61 39
8. Aranha, D.F., Gouvêa, C.P.L., Markmann, T., Wahby, R.S., Liao, K.: RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic 28
9. Aumüller, M., Bernhardsson, E., Faithfull, A.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms (2018) 29
10. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) Advances in Cryptology – CRYPTO 2007. Lecture Notes in Computer Science, vol. 4622, pp. 535–552. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2007). https://doi.org/10.1007/978-3-540-74143-5_30 7
11. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer, Berlin, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006). https://doi.org/10.1007/11761679_25 8
12. Berlekamp, E.R.: Factoring polynomials over large finite fields. Mathematics of Computation **24**(111), 713–735 (1970), http://www.jstor.org/stable/2004849 39
13. Bernstein, D.J.: Curve25519: New Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 3958, pp. 207–228. Springer, Berlin, Heidelberg, Germany, New York, NY, USA (Apr 24–26, 2006). https://doi.org/10.1007/11745853_14 28
14. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015, Part I. Lecture Notes in Computer Science, vol. 9452, pp. 470–491. Springer, Berlin, Heidelberg, Germany, Auckland, New Zealand (Nov 30 – Dec 3, 2015). https://doi.org/10.1007/978-3-662-48797-6_20 7

15. Bogatov, D., Kellaris, G., Kollios, G., Nissim, K., O'Neill, A.: εpsolute: Efficiently querying databases while providing differential privacy. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021: 28th Conference on Computer and Communications Security. pp. 2262–2276. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). https://doi.org/10.1145/3460120.3484786 59
16. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) Advances in Cryptology – EUROCRYPT 2009. Lecture Notes in Computer Science, vol. 5479, pp. 224–241. Springer, Berlin, Heidelberg, Germany, Cologne, Germany (Apr 26–30, 2009). https://doi.org/10.1007/978-3-642-01001-9_13 7
17. Boldyreva, A., Gui, Z., Warinschi, B.: Understanding leakage in searchable encryption: a quantitative approach. Cryptology ePrint Archive, Paper 2024/1558 (2024). https://doi.org/10.56553/popets-2024-0127, https://eprint.iacr.org/2024/1558 7
18. Boldyreva, A., Tang, T.: Privacy-preserving approximate k-nearest-neighbors search that hides access, query and volume patterns. Proceedings on Privacy Enhancing Technologies **2021**(4), 549–574 (Oct 2021). https://doi.org/10.2478/popets-2021-0084 5, 30
19. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011: 8th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 6597, pp. 253–273. Springer, Berlin, Heidelberg, Germany, Providence, RI, USA (Mar 28–30, 2011). https://doi.org/10.1007/978-3-642-19571-6_16 2
20. Castagnos, G., Laguillaumie, F., Tucker, I.: Practical fully secure unrestricted inner product functional encryption modulo p. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018, Part II. Lecture Notes in Computer Science, vol. 11273, pp. 733–764. Springer, Cham, Switzerland, Brisbane, Queensland, Australia (Dec 2–6, 2018). https://doi.org/10.1007/978-3-030-03329-3_25 7
21. Catalano, D., De Prisco, R. (eds.): SCN 18: 11th International Conference on Security in Communication Networks, Lecture Notes in Computer Science, vol. 11035. Springer, Cham, Switzerland, Amalfi, Italy (Sep 5–7, 2018) 33
22. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 9614, pp. 164–195. Springer, Berlin, Heidelberg, Germany, Taipei, Taiwan (Mar 6–9, 2016). https://doi.org/10.1007/978-3-662-49384-7_7 7
23. Dowerah, U., Dutta, S., Mitrokotsa, A., Mukherjee, S., Pal, T.: Unbounded predicate inner product functional encryption from pairings. Journal of Cryptology **36**(3), 29 (Jul 2023). https://doi.org/10.1007/s00145-023-09458-2 2, 14
24. Fuchsbauer, G., Ghosal, R., Hauke, N., O'Neill, A.: Approximate distance-comparison-preserving symmetric encryption. In: Galdi, C., Jarecki, S. (eds.) SCN 22: 13th International Conference on Security in Communication Networks. Lecture Notes in Computer Science, vol. 13409, pp. 117–144. Springer, Cham, Switzerland, Amalfi, Italy (Sep 12–14, 2022). https://doi.org/10.1007/978-3-031-14791-3_6 7
25. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual Symposium on Foundations of Computer Science. pp. 40–49.

IEEE Computer Society Press, Berkeley, CA, USA (Oct 26–29, 2013). https://doi.org/10.1109/FOCS.2013.13 2
26. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 578–602. Springer, Berlin, Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_32 2, 7
27. Haagh, H., Ji, Y., Li, C., Orlandi, C., Song, Y.: Revealing encryption for partial ordering. In: O'Neill, M. (ed.) Cryptography and Coding - 16th IMA International Conference, IMACC 2017, Oxford, UK, December 12-14, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10655, pp. 3–22. Springer (2017) 2, 7
28. Hadjiabadi, M., Langrehr, R., O'Neill, A., Wang, M.: On the black-box complexity of private-key inner-product functional encryption. In: TCC (2024), To appear. 7
29. Hardy, G., Wright, E., Heath-Brown, D., Silverman, J.: An Introduction to the Theory of Numbers. Oxford mathematics, Oxford University Press, sixth edn. (2008) 19, 39
30. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Khuller, S., Williams, V.V. (eds.) 53rd Annual ACM Symposium on Theory of Computing. pp. 60–73. ACM Press, Virtual Event, Italy (Jun 21–25, 2021). https://doi.org/10.1145/3406325.3451093 2
31. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence **33**(1), 117–128 (2010) 29
32. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Transactions on Big Data **7**(3), 535–547 (2019) 4
33. Joye, M., Passelègue, A.: Function-revealing encryption - definitions and constructions. In: Catalano and De Prisco [21], pp. 527–543. https://doi.org/10.1007/978-3-319-98113-0_28 2, 6, 7
34. Katz, J. (ed.): PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography, Lecture Notes in Computer Science, vol. 9020. Springer, Berlin, Heidelberg, Germany, Gaithersburg, MD, USA (Mar 30 – Apr 1, 2015) 30, 31
35. Khattab, O., Zaharia, M.: Colbert: Efficient and effective passage search via contextualized late interaction over bert. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 39–48. SIGIR '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3397271.3401075, https://doi.org/10.1145/3397271.3401075 4
36. Kim, S., Lewi, K., Mandal, A., Montgomery, H., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. In: Catalano and De Prisco [21], pp. 544–562. https://doi.org/10.1007/978-3-319-98113-0_29 4, 6, 7, 24, 27, 28, 47
37. Kim, S., Kim, J., Seo, J.H.: A new approach to practical function-private inner product encryption. Theoretical Computer Science **783**, 22–40 (2019). https://doi.org/https://doi.org/10.1016/j.tcs.2019.03.016, https://www.sciencedirect.com/science/article/pii/S0304397519301690 6, 7, 27, 28
38. Lai, Q., Liu, F.H., Wang, Z.: New lattice two-stage sampling technique and its applications to functional encryption - stronger security and smaller ciphertexts. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021, Part I. Lecture Notes in Computer Science, vol. 12696, pp. 498–527. Springer,

Cham, Switzerland, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77870-5_18 2
39. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist **2** (2010) 29
40. Lin, J., Nogueira, R.F., Yates, A.: Pretrained Transformers for Text Ranking: BERT and Beyond. Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers (2021) 4
41. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell. **42**(4), 824–836 (2020) 5, 29
42. Newman, D.: Bag of Words. UCI Machine Learning Repository (2008), DOI: https://doi.org/10.24432/C5ZG6P 29
43. Pandey, O., Rouselakis, Y.: Property preserving symmetric encryption. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 375–391. Springer, Berlin, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012). https://doi.org/10.1007/978-3-642-29011-4_23 3, 7, 36
44. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014), http://www.aclweb.org/anthology/D14-1162 29
45. Pollack, P., Treviño, E.: Finding the four squares in lagrange's theorem. Integers: Electronic Journal of Combinatorial Number Theory **18A** (2018). https://doi.org/10.5281/zenodo.10581444 3, 19
46. Rabin, M.O., Shallit, J.O.: Randomized algorithms in number theory. Communications on Pure and Applied Mathematics **39**(S1), S239–S256 (1986). https://doi.org/https://doi.org/10.1002/cpa.3160390713, https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160390713 3, 19
47. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) Advances in Cryptology – EUROCRYPT'97. Lecture Notes in Computer Science, vol. 1233, pp. 256–266. Springer, Berlin, Heidelberg, Germany, Konstanz, Germany (May 11–15, 1997). https://doi.org/10.1007/3-540-69053-0_18 47
48. Tomida, J., Abe, M., Okamoto, T.: Efficient functional encryption for inner-product values with full-hiding security. In: Bishop, M., Nascimento, A.C.A. (eds.) ISC 2016: 19th International Conference on Information Security. Lecture Notes in Computer Science, vol. 9866, pp. 408–425. Springer, Cham, Switzerland, Honolulu, HI, USA (Sep 3–6, 2016). https://doi.org/10.1007/978-3-319-45871-7_24 7
49. Ünal, A.: Impossibility results for lattice-based functional encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 169–199. Springer, Cham, Switzerland, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45721-1_7 3
50. Wee, H.: Functional encryption for quadratic functions from k-lin, revisited. In: Pass, R., Pietrzak, K. (eds.) TCC 2020: 18th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 12550, pp. 210–228. Springer, Cham, Switzerland, Durham, NC, USA (Nov 16–19, 2020). https://doi.org/10.1007/978-3-030-64375-1_8 5, 22
51. Xiong, L., Xiong, C., Li, Y., Tang, K.F., Liu, J., Bennett, P.N., Ahmed, J., Overwijk, A.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: International Conference on Learning Representations. ICLR '21 (2021), https://openreview.net/forum?id=zeFrfgyZln 4

52. Zeng, H., Zamani, H., Vinay, V.: Curriculum learning for dense retrieval distillation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 1979–1983. SIGIR '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3477495.3531791, https://doi.org/10.1145/3477495.3531791 4

A Equivalence of FH-IPBFE and IPBFRE

We show here the omitted steps for the equivalence of FH-IPBFE and IPBFRE in detail.

A.1 From FH-IPBFE to IPBFRE

First, we give a transformation that takes an FH-IPBFE and constructs an IPBFRE using the FH-IPBFE scheme. This transformation uses ideas from [43]. This transformation does not use any specifics about inner-product functionalities. It can be generalized to turn FH-FE for any function class $\mathcal{F}$ into FRE for the function $g(x, y_f) := f(x)$, where y_f is a suitable encoding of f, if the function g is symmetric.

Let $\mathsf{FH\text{-}IPBFE} = (\mathsf{PPGen}, \mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ be an FH-IPBFE scheme with message-space and key-space $\mathbb{Z}_p^d$ for a prime p and dimension $d \in \mathbb{N}$. Define the associated IPBFRE scheme $\mathsf{IPBFRE}[\mathsf{FH\text{-}IPBFE}] = (\mathsf{PPGen}, \mathsf{Setup}, \mathsf{Enc}', \mathsf{Eval}')$ with message-space $\mathbb{Z}_p^d$ where PPGen and Setup remain the same as FH-IPBFE and the rest is given in Fig. 15.

$\underline{\mathsf{Enc}'(\mathsf{msk}, \mathbf{x}):}$

1 $\mathsf{ct}_{\mathbf{x}} \leftarrow\!\!\$\ \mathsf{FH\text{-}IPBFE.Enc}(\mathsf{msk}, \mathbf{x})$
2 $\mathsf{sk}_{\mathbf{x}} \leftarrow\!\!\$\ \mathsf{FH\text{-}IPBFE.KeyGen}(\mathsf{msk}, \mathbf{x})$
3 $\mathsf{ct}'_{\mathbf{x}} \leftarrow (\mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_{\mathbf{x}})$
4 **return** $\mathsf{ct}'_{\mathbf{x}}$

$\underline{\mathsf{Eval}'(\mathsf{ct}'_{\mathbf{x}_1}, \mathsf{ct}'_{\mathbf{x}_2}):}$

5 $(\mathsf{ct}_{\mathbf{x}_1}, \mathsf{sk}_{\mathbf{x}_1}) \leftarrow \mathsf{ct}'_{\mathbf{x}_1}$; $(\mathsf{ct}_{\mathbf{x}_2}, \mathsf{sk}_{\mathbf{x}_2}) \leftarrow \mathsf{ct}'_{\mathbf{x}_2}$
6 $y \leftarrow\!\!\$\ \mathsf{FH\text{-}IPBFE.Dec}(\mathsf{sk}_{\mathbf{x}_1}, \mathsf{ct}_{\mathbf{x}_2})$
7 **return** y

Fig. 15. Transform from FH-IPBFE to IPBFRE.

Proposition 3. *If* FH-IPBFE *is decryption correct, then* IPBFRE[FH-IPBFE] *is evaluation correct.*

Proof. Let $\mathsf{pp} \in \mathbf{Out}(\mathsf{FH\text{-}IPBFE.PPGen}(1^\lambda))$, $\mathsf{msk} \in \mathbf{Out}(\mathsf{FH\text{-}IPBFE.Setup}(\mathsf{pp}, 1^d))$ for $\lambda, d \in \mathbb{N}$. Next, let $\mathsf{sk}_{\mathbf{x}_1} \in \mathbf{Out}(\mathsf{FH\text{-}IPBFE.KeyGen}(\mathsf{msk}, \mathbf{x}_1))$ and $\mathsf{ct}_{\mathbf{x}_2} \in \mathbf{Out}(\mathsf{FH\text{-}IPBFE.Enc}(\mathsf{msk}, \mathbf{x}_2))$. By decryption correctness of FH-IPBFE, we know that $\mathsf{FH\text{-}IPBFE.Dec}(\mathsf{pp}, \mathsf{sk}_{\mathbf{x}_1}, \mathsf{ct}_{\mathbf{x}_2}) = g(\langle \mathbf{x}_1, \mathbf{x}_2 \rangle)$. We have that

$$\mathsf{IPBFRE[FH\text{-}IPBFE].Eval}(\mathsf{ct}'_{\mathbf{x}_1}, \mathsf{ct}'_{\mathbf{x}_2}) = \mathsf{FH\text{-}IPBFE.Dec}(\mathsf{sk}_{\mathbf{x}_1}, \mathsf{ct}_{\mathbf{x}_2}).$$

Thus, evaluation correctness of IPBFRE[FH-IPBFE] is fulfilled.

Theorem 7 (IND-security). *If* FH-IPBFE *is IND-secure, then* IPBFRE[FH-IPBFE] *is IND-secure. Concretely, for every adversary* A *against the IND-security of* IPBFRE[FH-IPBFE]*, there exists an adversary* B *against the IND-security of* FH-IPBFE *with roughly the same runtime as* A *such that for every* $\lambda \in \mathbb{N}$

$$\mathsf{Adv}^{\text{ind}}_{\mathsf{FH\text{-}IPBFE},\mathrm{B}}(\lambda) \;=\; \mathsf{Adv}^{\text{ind}}_{\mathsf{IPBFRE[FH\text{-}IPBFE]},\mathrm{A}}(\lambda)\ .$$

Proof. Let A be an adversary against IND-security of IPBFRE[FH-IPBFE]. We construct an adversary B against IND-security of FH-IPBFE in Figure 16.

Adversary $\mathrm{B} = (\mathrm{B}_1, \mathrm{B}_2)$

$\mathrm{B}_1(1^\lambda, \mathsf{pp})$:
1 $(d, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
2 **return** (d, st)

$\mathrm{B}_2^{\textsc{EncO}'(\cdot,\cdot),\textsc{KeyGenO}'(\cdot,\cdot)}(st)$:
3 $b' \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{EncO}(\cdot,\cdot)}(st)$
4 **return** b'

$\textsc{EncO}(\mathbf{x}_0, \mathbf{x}_1)$:
5 $\mathsf{ct}_{\mathbf{x}} \leftarrow \textsc{EncO}'(\mathbf{x}_0, \mathbf{x}_1)$
6 $\mathsf{sk}_{\mathbf{x}} \leftarrow \textsc{KeyGenO}'(\mathbf{x}_0, \mathbf{x}_1)$
7 $\mathsf{ct} \leftarrow (\mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_{\mathbf{x}})$
8 **return** ct

Fig. 16. Adversary against IND-security of IPBFRE[FH-IPBFE].

It is easy to see that when A plays the $\mathbf{G}^{\text{ind-b}}_{\mathsf{IPBFRE[FH\text{-}IPBFE]},A}(\lambda)$ game, it perfectly simulates the game $\mathbf{G}^{\text{ind-}b}_{\mathsf{FH\text{-}IPBFE},\mathrm{B}}(\lambda)$ for B and outputs the same bit as B. To show that these adversaries have the same advantage, what remains to show is that A causes the game to output $\perp$ exactly when B does.

Assume A causes the game to output $\perp$. Then A made Enc queries with $(\mathbf{x}_0^j, \mathbf{x}_1^j)$ and $(\mathbf{x}_0^k, \mathbf{x}_1^k)$ such that $g(\langle \mathbf{x}_0^j, \mathbf{x}_0^k \rangle) \neq g(\langle \mathbf{x}_1^j, \mathbf{x}_1^k \rangle)$. In order to simulate these oracle queries, B made in particular an Enc′ query for $(\mathbf{x}_0^j, \mathbf{x}_1^j)$ and an KeyGen′ query for $(\mathbf{x}_0^k, \mathbf{x}_1^k)$. Thus also B causes its game to abort.

Next, assume B causes the game to abort. Then B made an Enc′ query for $(\mathbf{x}_0^i, \mathbf{x}_1^i)$ and an KeyGen′ query for $(\mathbf{y}_0^j, \mathbf{y}_1^j)$ such that $g(\langle \mathbf{x}_0^i, \mathbf{y}_0^j \rangle) \neq g(\langle \mathbf{x}_1^i, \mathbf{y}_1^j \rangle)$. These queries could only be cause by A making Enc queries for $(\mathbf{x}_0^i, \mathbf{x}_1^i)$ and $(\mathbf{y}_0^j, \mathbf{y}_1^j)$. Thus, A also made its game abort.

With this, we see

$$\Pr[\mathbf{G}^{\text{ind-1}}_{\mathsf{FH\text{-}IPBFE},\mathrm{B}}(\lambda) \Rightarrow 1] \;=\; \Pr[\mathbf{G}^{\text{ind-1}}_{\mathsf{IPBFRE[FH\text{-}IPBFE]},\mathrm{A}}(\lambda) \Rightarrow 1]$$

and

$$\Pr[\mathbf{G}^{\text{ind-0}}_{\mathsf{FH\text{-}IPBFE},\mathrm{B}}(\lambda) \Rightarrow 1] \;=\; \Pr[\mathbf{G}^{\text{ind-0}}_{\mathsf{IPBFRE[FH\text{-}IPBFE]},\mathrm{A}}(\lambda) \Rightarrow 1]\ .$$

Subtracting yields the result.

Theorem 8 (SIM-security). *If* FH-IPBFE *is SIM-secure, then* IPBFRE[FH-IPBFE] *is SIM-secure. Concretely, for every adversary* A *against* IPBFRE[FH-IPBFE]*, there exists an adversary* B *against* FH-IPBFE *such that for every simulator* $\mathrm{S_B}$ *there exists a simulator* S_A *where for all* $\lambda \in \mathbb{N}$

$$\mathsf{Adv}^{\mathrm{sim}}_{\mathsf{IPBFRE[FH\text{-}IPBFE]},\mathrm{A},\mathrm{S_A}}(\lambda) = \mathsf{Adv}^{\mathrm{sim}}_{\mathsf{FH\text{-}IPBFE},\mathrm{B},\mathrm{S_B}}(\lambda).$$

Proof. Let A be an adversary against IPBFRE[FH-IPBFE]. We first construct an adversary B against SIM-security of FH-IPBFE in Figure 17.

Next, by SIM-security of FH-IPBFE, for an adversary B there exists a simulator $\mathrm{S_B} = (\mathrm{S_{B,1}}, \mathrm{S_{B,2}}, \mathrm{S_{B,3}}, \mathrm{S_{B,4}})$. Then, consider the simulator $S_\mathrm{A} = (S_{\mathrm{A},1}, S_{\mathrm{A},2}, S_{\mathrm{A},3})$ for IPBFRE[FH-IPBFE] for an adversary A also given in figure 17.

Adversary $\mathrm{B} = (\mathrm{B}_1, \mathrm{B}_2)$

$\mathrm{B}_1(1^\lambda, \mathsf{pp})$:
1 $(d, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
2 **return** (d, st)

$\mathrm{B}_2^{\textsc{EncO}'(\cdot),\textsc{KeyGenO}'(\cdot)}(st)$:
3 $b \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{EncO}(\cdot)}(st)$
4 **return** b

$\textsc{EncO}(\mathbf{x})$:
5 $\mathsf{ct_x} \leftarrow \textsc{EncO}'(\mathbf{x})$
6 $\mathsf{sk_x} \leftarrow \textsc{KeyGenO}'(\mathbf{x})$
7 $\mathsf{ct} \leftarrow (\mathsf{ct_x}, \mathsf{sk_x})$
8 **return** ct

Simulator $\mathrm{S_A} = (\mathrm{S_{A,1}}, \mathrm{S_{A,2}}, \mathrm{S_{A,3}})$

$S_{\mathrm{A},1}(1^\lambda)$:
1 $(\mathsf{pp}, st_\mathrm{B}) \leftarrow\!\!\$\ \mathrm{S_{B,1}}(1^\lambda)$
2 $st \leftarrow st_\mathrm{B}$
3 **return** (pp, st)

$\mathrm{S_{A,2}}(1^d, st)$:
4 $i \leftarrow 0;\ st_\mathrm{B} \leftarrow st$
5 $st_\mathrm{B} \leftarrow\!\!\$\ \mathrm{S_{B,2}}(1^d, st_\mathrm{B})$
6 $st \leftarrow (st_\mathrm{B}, k)$
7 **return** st

$\mathrm{S_{A,3}}(\mathcal{C}'_{\mathsf{ip}}, st)$:
8 $(st_\mathrm{B}, k) \leftarrow st$
9 $i \leftarrow i + 1$
10 $\mathcal{C}_{\mathsf{ip}} \leftarrow \{((j,k), c_{j,k}) \mid ((j,k), c_{j,k}) \in \mathcal{C}'_{\mathsf{ip}} \wedge (j \le i) \wedge (k < i)\} \cup \{((k,j), c_{j,k}) \mid ((j,k), c_{j,k}) \in \mathcal{C}'_{\mathsf{ip}} \wedge (j \le i) \wedge (j < i)\}$
11 $(\mathsf{ct}_i, st_\mathrm{B}) \leftarrow S_{\mathrm{B},3}(\mathcal{C}_{\mathsf{ip}}, st_\mathrm{B})$
12 $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}_{\mathsf{ip}} \cup \{(j,i), c_{j,i}) \mid ((j,i), c_{j,i}) \in \mathcal{C}'_{\mathsf{ip}} \wedge (i > j)\} \cup \{(i,j), c_{j,i}) \mid ((j,i), c_{j,i}) \in \mathcal{C}'_{\mathsf{ip}} \wedge (i > j)\}$
13 $(\mathsf{sk}_i, st_\mathrm{B}) \leftarrow S_{\mathrm{B},4}(\mathcal{C}_{\mathsf{ip}}, st_\mathrm{B})$
14 $\mathsf{ct} \leftarrow (\mathsf{ct}_i, \mathsf{sk}_i)$
15 $st \leftarrow (st_\mathrm{B}, k)$
16 **return** (ct, st)

Fig. 17. Adversary against SIM-security of FH-IPBFE and simulator for SIM security of IPBFRE[FH-IPBFE].

The simulator S_A gets the set $\mathcal{C}'_{\mathsf{ip}}$ that contains all pairwise inner products of IPBFRE ciphertexts with each other using a global index to keep track of the

queries. The simulator turns this into a set $\mathcal{C}_{\mathsf{ip}}$ that is indexed by query-type (Enc or KeyGen) specific indices, where the first index corresponds to the ciphertext and the second one corresponds to the key. When adding new tuples to $\mathcal{C}_{\mathsf{ip}}$, the simulator must keep track of the indices and only add new values right before running $\mathrm{S}_{\mathrm{B},3}$ or $\mathrm{S}_{\mathrm{B},4}$ with an updated $\mathcal{C}_{\mathsf{ip}}$ as given in simulator lines 9-13.

With this, we see

$$\Pr[\mathbf{G}^{\text{sim-1}}_{\mathsf{IPBFRE}[\mathsf{FH\text{-}IPBFE}],\mathrm{A}}(\lambda) \Rightarrow 1] = \Pr[\mathbf{G}^{\text{sim-1}}_{\mathsf{FH\text{-}IPBFE},\mathrm{B}}(\lambda) \Rightarrow 1]$$

and

$$\Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{IPBFRE}[\mathsf{FH\text{-}IPBFE}],\mathrm{A},\mathrm{S}_\mathrm{A}}(\lambda) \Rightarrow 1] = \Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{FH\text{-}IPBFE},\mathrm{B},\mathrm{S}_\mathrm{B}}(\lambda) \Rightarrow 1] \ .$$

Subtracting yields the result.

A.2 From NZIP-AC-IPBFRE to FH-IPBFE

In this Section we give an optimized version of the transformation from NZIP-AC-IPBFRE to FH-IPBFE. In many IPBFE/IPBFRE constructions the modulus m is a prime. In this case, the following theorem gives us a better solution than Lagrange's four square theorem.

Theorem 9. *There exists a PT algorithm* PrimeSoS *that inputs a prime m and outputs (a_1, a_2) such that $1 + a_1^2 + a_2^2 = 0 \mod m$.*

Proof. We first proof the existence of a solution. The proof is taken from [29, §6.7].

The statement is trivial for $m = 2$. Therefore, assume that m is odd in the following.

The polynomial $X^2 - c$, for $c \in \mathbb{Z}_m$, has at most two distinct solutions in $\mathbb{Z}_m$. Thus, by squaring every $a \in \mathbb{Z}_m$ we hit each quadratic residue at most twice and thus there must be at least $(m+1)/2$ quadratic residue in $\mathbb{Z}_m$.

Using the above argument with the polynomial $-X^2 - c - 1$, there must also be $(m+1)/2$ integers of the form $-a^2 - 1$ in $\mathbb{Z}_m$. By the pigeonhole principle, there must be one c with $-a^2 - 1 = c = b^2 \mod m$ for some $a, b \in \mathbb{Z}_m$ which can be turned immediately into a solution of the above equation.

We next describe PrimeSoS, namely how to compute a_1 and a_2 efficiently. We show how to proceed for primes of the form $m = 4k + 3$. For primes of the form $m = 4k + 1$ one can proceed analogously (but we do not need that here, because we will show below that for those prime a solution with $\ell = 1$ can be found).

The first step is to find a number $c \in \mathbb{Z}_m$ that satisfies the properties required in the proof, i.e. such that $X^2 - c$ and $-X^2 - c - 1$ have both solutions in $\mathbb{Z}_m$. That is, c and $-(c+1)$ should be both quadratic residues. -1 is a quadratic non-residue for primes of the form $m = 4k + 3$, the second requirement is fulfilled iff $c + 1$ is a quadratic non-residue (law of quadratic reciprocity and its first supplement). There are $(m+1)/4$ possible values for c satisfying this [7], we can thus just compute one by trial and error. The values a_1 and a_2 can now be computed by computing the roots of the polynomials $X^2 - c$ and $-X^2 - c - 1$, which can be done efficiently [12].

We now focus on further optimizing efficiency. For primes of the form $m = 4k + 1$, we can do even better and find a solution with $\ell = 2$.

Theorem 10. *There exists a PT algorithm* Prime1Mod4SoS *that inputs a prime m of the form $m = 4k + 1$ for an $k \in \mathbb{Z}$ and outputs a_1 such that $1 + a_1^2 = 0 \mod m$.*

Proof. The existence of such an a_1 follows from the first supplement to the law of quadratic reciprocity, $\left(\frac{-1}{m}\right) = 1$ for primes m of the form $m = 4k+1$. This implies that there exists $a_1 \in \mathbb{Z}_m$ with $a_1^2 = -1 \mod m$, which can be rearranged to the equation in the theorem.

We proceed to describe Prime1Mod4SoS. We can find a_1 from the above theorem in expected time $\mathcal{O}(\log m)$ as follows: By Euler's criterion, $\left(\frac{x}{m}\right) = x^{\frac{m-1}{2}} \mod m$ and thus for every quadratic non-residue x we have $x^{\frac{m-1}{2}} = -1 \mod m$ and by setting $a_1 = x^{\frac{m-1}{4}}$ we have a solution. Since every other (non-zero) number in $\mathbb{Z}_m$ is a quadratic non-residue, we can find one efficiently by trial and error (again, converting the algorithm to worst-case polynomial-time in standard ways).

Our Encoding Functions and Their Properties Given a sum of squares decomposition of the modulus $1 + a_1^2 + \cdots + a_\ell^2 = 0 \mod m$, the following two tag vectors will be used in our transformation:

$$\mathbf{u}_{(a_i)_{1\le i\le \ell}} := \begin{pmatrix} 1 \\ a_1 \\ 0 \\ \vdots \\ a_\ell \\ 0 \end{pmatrix} \in \mathbb{Z}_m^{2\ell+1} \qquad \mathbf{v}_{(a_i)_{1\le i\le \ell}} := \begin{pmatrix} 1 \\ 0 \\ a_1 \\ \vdots \\ 0 \\ a_\ell \end{pmatrix} \in \mathbb{Z}_m^{2\ell+1} .$$

The following gives the crucial properties of the tag vectors, which are easy to prove by inspection.

Lemma 1.

$$\langle \mathbf{v}_{(a_i)_{1\le i\le \ell}}, \mathbf{v}_{(a_i)_{1\le i\le \ell}} \rangle = 0 \qquad \textit{and} \qquad \langle \mathbf{u}_{(a_i)_{1\le i\le \ell}}, \mathbf{u}_{(a_i)_{1\le i\le \ell}} \rangle = 0 .$$

Lemma 2.

$$\langle \mathbf{v}_{(a_i)_{1\le i\le \ell}}, \mathbf{u}_{(a_i)_{1\le i\le \ell}} \rangle = a_1^2 = 1 .$$

The Transformation Recall in Section 4.1 we showed how to construct NZIP-AC-IPBRE generically from IPBFRE for the same inner-product based functionality $f(\mathbf{x}, \mathbf{y}) = g(\langle \mathbf{x}, \mathbf{y} \rangle)$. We now construct an FH-IPBFE generically from an NZIP-AC-IPBFRE. This transformation can easily be generalized to constructing FH-FE for any functionality form NZIP-AC-FRE for the same functionality. The transformation is given in Fig. 18.

$\underline{\mathsf{Setup}'(1^\lambda, 1^d)}$:

Obtain modulus m from pp
if m is prime **then**
  if $m \equiv 1 \mod 4$ **then**
    $(a_i)_{1\le i\le \ell} := (a_1) \leftarrow\!\!\$\ \mathsf{Prime1Mod4SoS}(m)$
  else
    $(a_i)_{1\le i\le \ell} := (a_1, a_2) \leftarrow\!\!\$\ \mathsf{PrimeSoS}(m)$
else
  $(a_i)_{1\le i\le \ell} := (a_1, a_2, a_3, a_4) \leftarrow\!\!\$\ \mathsf{LagrangeSoS}(m-1)$
$\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^d, 1^{2\ell+1})$
return $\mathsf{msk}' := (\mathsf{msk}, (a_i)_{1\le i\le \ell})$

$\underline{\mathsf{KeyGen}(\mathsf{msk}' = (\mathsf{msk}, (a_i)_{1\le i\le \ell}), \mathbf{x})}$:

return $\mathsf{Enc}(\mathsf{msk}, \mathbf{u}_{(a_i)_{1\le i\le \ell}}, \mathbf{x}))$

$\underline{\mathsf{Enc}'(\mathsf{msk}' = (\mathsf{msk}, (a_i)_{1\le i\le \ell}), \mathbf{y})}$:

return $\mathsf{Enc}(\mathsf{msk}, \mathbf{v}_{(a_i)_{1\le i\le \ell}}, \mathbf{y})$

$\underline{\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_\mathbf{x}, \mathsf{ct}_\mathbf{y})}$:

return $\mathsf{Eval}(\mathsf{pp}, \mathsf{sk}_\mathbf{x}, \mathsf{ct}_\mathbf{y})$

Fig. 18. Transform from $\mathsf{NZIP\text{-}AC\text{-}IPBFRE} = (\mathsf{PPGen}, \mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval})$ to $\mathsf{FH\text{-}IPBFE}[\mathsf{NZIP\text{-}AC\text{-}IPBFRE}] = (\mathsf{PPGen}, \mathsf{Setup}', \mathsf{KeyGen}, \mathsf{Enc}', \mathsf{Dec})$.

Proposition 4. *If* NZIP-AC-IPBFRE *is evaluation correct, then* FH-IPBFE[IPBFRE] *is decryption correct.*

Proof. Let $\mathsf{pp} \in \mathbf{Out}(\mathsf{PPGen}(1^\lambda))$, $\mathsf{msk} \in \mathbf{Out}(\mathsf{Setup}(1^\lambda, 1^d))$, $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_m^d$, $\mathsf{sk}_\mathbf{x} \in \mathbf{Out}(\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x}))$ and $\mathsf{ct}_\mathbf{y} \in \mathbf{Out}(\mathsf{Enc}(\mathsf{msk}, \mathbf{y}))$. Then by Lemma 2, the tags of the NZIP-AC-IPBFRE ciphertexts $\mathsf{sk}_\mathbf{x}$ and $\mathsf{ct}_\mathbf{y}$ satisfy the access policy and by evaluation correctness of NZIP-AC-IPBFRE $\mathsf{Eval}(\mathsf{pp}, \mathsf{sk}_\mathbf{x}, \mathsf{ct}_\mathbf{y}) = g(\langle \mathbf{x}, \mathbf{y} \rangle)$. Thus decryption correctness of FH-IPBFE[IPBFRE] follows.

Theorem 11 (IND-security). *If* NZIP-AC-IPBFRE *is IND-secure, then* FH-IPBFE[NZIP-AC-IPBFRE] *is IND-secure. Concretely, for every adversary* A *against IND-security of* FH-IPBFE[NZIP-AC-IPBFRE]*, there exists an adversary* B *against IND-security of* NZIP-AC-IPBFRE *with roughly the same runtime, such that for all* $\lambda \in \mathbb{N}$

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{FH\text{-}IPBFE}[\mathsf{NZIP\text{-}AC\text{-}IPBFRE}],\mathrm{A}}(\lambda) \;\le\; \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE},\mathrm{B}}(\lambda)\ .$$

Proof. Assume there exists an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ against the IND-security of FH-IPBFE[NZIP-AC-IPBFRE]. We then construct an adversary $\mathrm{B} = (\mathrm{B}_1, \mathrm{B}_2)$ against the adaptive IND security of NZIP-AC-IPBFRE as follows:

It is easy to see that when A plays the $\mathbf{G}^{\text{ind-b}}_{\mathsf{FH\text{-}IPBFE}[\mathsf{NZIP\text{-}AC\text{-}IPBFRE}],A}(\lambda)$ game, it perfectly simulates the game $\mathbf{G}^{\text{ind-}b}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE},\mathrm{B}}(\lambda)$ for B and outputs the same bit as B. To show that these adversaries have the same advantage, what remains to show is that A causes the game to output $\perp$ exactly when B does.

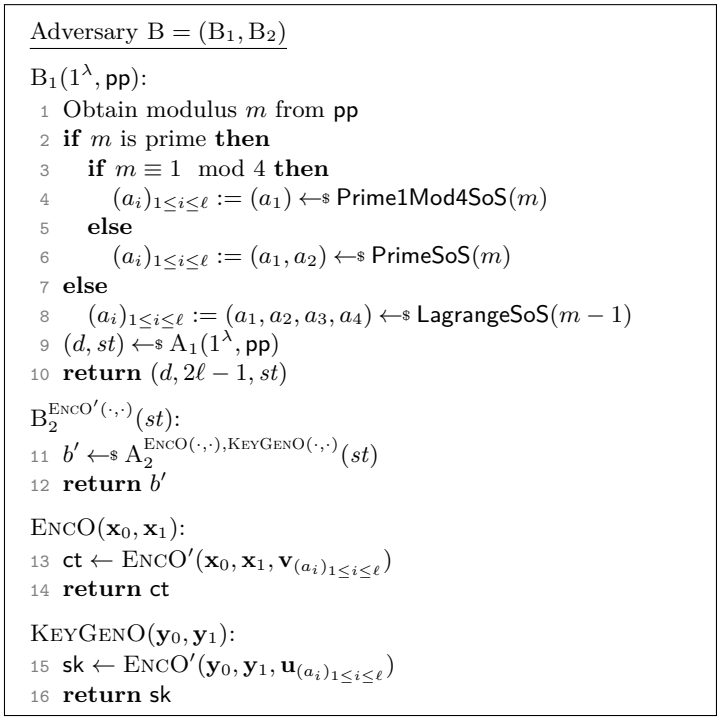


Fig. 19. Adversary against IND-security of NZIP-AC-IPBFRE.

Let $\mathbf{x}_0^j, \mathbf{x}_1^j$ be the input to the j-th Enc oracle query of A_2 and $\mathbf{y}_0^k, \mathbf{y}_1^k$ be the input to the k-th KeyGen oracle query of A_2. The IND-security game for FH-IPBFE[NZIP-AC-IPBFRE] returns $\perp$ if there exists indices j, k such that

$$g(\langle \mathbf{x}_0^j, \mathbf{y}_0^k \rangle) \neq g(\langle \mathbf{x}_1^j, \mathbf{y}_1^k \rangle). \tag{4}$$

To answer the j-th Enc oracle query of A_2, B_2 makes a Enc query for $\mathbf{x}_0^j, \mathbf{x}_1^j, \mathbf{v}_{(a_i)_{1\leq i\leq \ell}}$ and to answer the k-th KeyGen oracle query of A_2, B_2 makes a Enc query for $\mathbf{y}_0^k, \mathbf{y}_1^k, \mathbf{u}_{(a_i)_{1\leq i\leq \ell}}$. From $a(\mathsf{pp}, \mathbf{v}_{(a_i)_{1\leq i\leq \ell}}, \mathbf{u}_{(a_i)_{1\leq i\leq \ell}}) = 1$ (by Lemma 2) and Eq. (4) it follows that B causes the IND-security game for NZIP-AC-IPBFRE to return $\perp$, too.

On the other hand, let $\mathbf{z}_0^j, \mathbf{z}_1^j, \mathbf{w}^j$ be the input of the j-th Enc query of B_2. The IND-security game for NZIP-AC-IPBFRE returns $\perp$ if there exists indices j, k such that $a(\mathsf{pp}, \mathbf{w}^j, \mathbf{w}^k) = 1$ and

$$g(\langle \mathbf{z}_0^j, \mathbf{z}_0^k \rangle) \neq g(\langle \mathbf{z}_1^j, \mathbf{z}_1^k \rangle). \tag{5}$$

The way we defined B_2, $\mathbf{w}^j, \mathbf{w}^k \in \{\mathbf{u}_{(a_i)_{1\leq i\leq \ell}}, \mathbf{v}_{(a_i)_{1\leq i\leq \ell}}\}$ must hold. This shows that the first condition can happen by Lemma 1 only if exactly one of $\mathbf{w}^j, \mathbf{w}^k$ equals $\mathbf{u}_{(a_i)_{1\leq i\leq \ell}}$ (and the other vector equals $\mathbf{v}_{(a_i)_{1\leq i\leq \ell}}$.) Without loss of generality let $\mathbf{w}^j = \mathbf{v}_{(a_i)_{1\leq i\leq \ell}}$ and $\mathbf{w}^k = \mathbf{u}_{(a_i)_{1\leq i\leq \ell}}$. Then the j-th Enc query of B_2 was cause by an $\textsc{Enc}(\mathbf{z}_0^j, \mathbf{z}_1^j)$ for A_2 and the k-th Enc query of B_2 was cause by an $\textsc{KeyGen}(\mathbf{z}_0^k, \mathbf{z}_1^k)$ for A_2. However, these two queries cause by Eq. (5) that the IND-security game for FH-IPBFE[NZIP-AC-IPBFRE] returns $\perp$, too.

With this, we see

$$\Pr[\mathbf{G}^{\text{ind-1}}_{\text{NZIP-AC-IPBFRE},\mathrm{B}}(\lambda) \Rightarrow 1] \;=\; \Pr[\mathbf{G}^{\text{ind-1}}_{\text{FH-IPBFE[NZIP-AC-IPBFRE]},\mathrm{A}}(\lambda) \Rightarrow 1]$$

and

$$\Pr[\mathbf{G}^{\text{ind-0}}_{\text{NZIP-AC-IPBFRE},\mathrm{B}}(\lambda) \Rightarrow 1] \;=\; \Pr[\mathbf{G}^{\text{ind-0}}_{\text{FH-IPBFE[NZIP-AC-IPBFRE]},\mathrm{A}}(\lambda) \Rightarrow 1]\,.$$

Subtracting yields the result.

Theorem 12 (SIM-security). *If* NZIP-AC-IPBFRE *is SIM-secure, then* FH-IPBFE[NZIP-AC-IPBFRE] *is SIM-secure. Concretely, for every adversary* A *against SIM-security of* FH-IPBFE[NZIP-AC-IPBFRE]*, there exists an adversary* B *against SIM-security of* NZIP-AC-IPBFRE *with roughly the same runtime as* A *such that for every simulator* S_A *there exists a simulator* S_B *where for all* $\lambda \in \mathbb{N}$

$$\mathsf{Adv}^{\text{sim}}_{\text{FH-IPBFE[NZIP-AC-IPBFRE]},\mathrm{A},S_\mathrm{A}}(\lambda) \;\leq\; \mathsf{Adv}^{\text{sim}}_{\text{NZIP-AC-IPBFRE},\mathrm{B},S_\mathrm{B}}(\lambda)\,.$$

Proof. First, consider an adversary A against simulation-based security of FH-IPBFE[NZIP-AC-IPBFRE]. If such an adversary exists, then we can use it to construct an adversary B against simulation-based security of NZIP-AC-IPBFRE as given in Fig. 20.

Let $S_\mathrm{A} = (S_{\mathrm{A},1}, S_{\mathrm{A},2}, S_{\mathrm{A},3})$ be the simulator for the SIM security of NZIP-AC-IPBFRE. We first construct a simulator $\mathrm{S}_\mathrm{B} = (\mathrm{S}_{\mathrm{B},1}, \mathrm{S}_{\mathrm{B},2}, \mathrm{S}_{\mathrm{B},3}, \mathrm{S}_{\mathrm{B},4})$ for the SIM security of FH-IPBFE[NZIP-AC-IPBFRE] as described by Fig. 21.

Adversary $\mathrm{B} = (\mathrm{B}_1, \mathrm{B}_2)$

$\mathrm{B}_1(1^\lambda, \mathsf{pp})$:
1 Obtain modulus m from pp
2 **if** m is prime **then**
3 **if** $m \equiv 1 \mod 4$ **then**
4 $(a_i)_{1 \le i \le \ell} := (a_1) \leftarrow\!\!\$\ \mathsf{Prime1Mod4SoS}(m)$
5 **else**
6 $(a_i)_{1 \le i \le \ell} := (a_1, a_2) \leftarrow\!\!\$\ \mathsf{PrimeSoS}(m)$
7 **else**
8 $(a_i)_{1 \le i \le \ell} := (a_1, a_2, a_3, a_4) \leftarrow\!\!\$\ \mathsf{LagrangeSoS}(m-1)$
9 $(d, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
10 **return** $(d, 2\ell - 1, st)$

$\mathrm{B}_2^{\textsc{EncO}'(\cdot)}(st)$:
11 $b' \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{EncO}(\cdot), \textsc{KeyGenO}(\cdot)}(st)$
12 **return** b'

$\textsc{EncO}(\mathbf{x})$:
13 $\mathsf{ct} \leftarrow \textsc{EncO}'(\mathbf{x}, \mathbf{v}_{(a_i)_{1 \le i \le \ell}})$
14 **return** ct

$\textsc{KeyGenO}(\mathbf{y})$:
15 $\mathsf{sk} \leftarrow \textsc{EncO}'(\mathbf{y}, \mathbf{u}_{(a_i)_{1 \le i \le \ell}})$
16 **return** sk

Fig. 20. Adversary against SIM-security of NZIP-AC-IPBFRE.

Simulator $\mathrm{S_B} = (\mathrm{S_{B,1}}, \mathrm{S_{B,2}}, \mathrm{S_{B,3}}, \mathrm{S_{B,4}})$

$\mathrm{S_{B,1}}(1^\lambda)$:
1 $(\mathsf{pp}, st_{\mathrm{S_A}}) \leftarrow\!\!\$\ \mathrm{S_{A,1}}(1^\lambda)$
2 **return** $(\mathsf{pp}, st_{\mathrm{S_B}} \leftarrow (\mathsf{pp}, st_{\mathrm{S_A}}))$

$\mathrm{S_{B,2}}(1^d, st_{\mathrm{S_B}} = (\mathsf{pp}, st_{\mathrm{S_A}}))$:
3 Obtain modulus m from pp
4 **if** m is prime **then**
5 **if** $m \equiv 1 \mod 4$ **then**
6 $(a_i)_{1\le i\le \ell} := (a_1) \leftarrow\!\!\$\ \mathsf{Prime1Mod4SoS}(m)$
7 **else**
8 $(a_i)_{1\le i\le \ell} := (a_1, a_2) \leftarrow\!\!\$\ \mathsf{PrimeSoS}(m)$
9 **else**
10 $(a_i)_{1\le i\le \ell} := (a_1, a_2, a_3, a_4) \leftarrow\!\!\$\ \mathsf{LagrangeSoS}(m-1)$
11 $st_{\mathrm{S_A}} \leftarrow\!\!\$\ \mathrm{S_{A,2}}(1^d, 1^{2\ell-1} st_{\mathrm{S,A}})$
12 $i \leftarrow 0; j \leftarrow 0; k \leftarrow 0; \mathcal{I} \leftarrow \emptyset$
13 $st_{\mathrm{S_B}} \leftarrow (st_{\mathrm{S_A}}, (a_i)_{1\le i\le \ell}, i, j, k, \mathcal{I})$
14 **return** $st_{\mathrm{S_B}}$

$\mathrm{S_{B,3}}(\mathcal{C}'_{\mathsf{ip}}, st_{\mathrm{S_B}} = (st_{\mathrm{S_A}}, (a_i)_{1\le i\le \ell}), i, j, k, \mathcal{I})$:
15 $i \leftarrow i+1; k \leftarrow k+1$
16 $\mathcal{I} \leftarrow \mathcal{I} \cup \{(i, k, \mathtt{Enc})\}$
17 $\mathcal{T} \leftarrow \{(k, \mathbf{v}_{(a_i)_{1\le i\le \ell}}) \mid \exists i' \in \mathbb{N} : (i', k, \mathtt{Enc}) \in \mathcal{I}\}$
$\qquad \cup \{(k, \mathbf{u}_{(a_i)_{1\le i\le \ell}}) \mid \exists j' \in \mathbb{N} : (j', k, \mathtt{KeyGen}) \in \mathcal{I}\}$
18 $\mathcal{C}_{\mathsf{ip}} \leftarrow \{((\min\{k', k''\}, \max\{k', k''\}), c_{i',j'}) \mid \exists i', j' \in \mathbb{N} : (i', k', \mathtt{Enc}) \in \mathcal{I}$
$\qquad \wedge (j', k'', \mathtt{KeyGen}) \in \mathcal{I} \wedge ((i', j'), c_{i',j'}) \in \mathcal{C}'_{\mathsf{ip}}\}$
19 $(\mathsf{ct}, st_{S_{\mathrm{A}}}) \leftarrow S_{\mathrm{A},3}(\mathcal{C}_{\mathsf{ip}}, \mathcal{T}, st_{S,\mathrm{A}})$
20 **return** $(\mathsf{ct}, st_{\mathrm{S_B}} = (st_{\mathrm{S_A}}, (a_i)_{1\le i\le \ell}), i, j, k, \mathcal{I}))$

$\mathrm{S_{B,4}}(\mathcal{C}'_{\mathsf{ip}}, st_{\mathrm{S_B}} = (st_{\mathrm{S_A}}, (a_i)_{1\le i\le \ell}), i, j, k, \mathcal{I})$:
21 $j \leftarrow j+1; k \leftarrow k+1$
22 $\mathcal{I} \leftarrow \mathcal{I} \cup \{(j, k, \mathtt{KeyGen})\}$
23 $\mathcal{T} \leftarrow \{(k, \mathbf{v}_{(a_i)_{1\le i\le \ell}}) \mid \exists i' \in \mathbb{N} : (i', k, \mathtt{Enc}) \in \mathcal{I}\}$
$\qquad \cup \{(k, \mathbf{u}_{(a_i)_{1\le i\le \ell}}) \mid \exists j' \in \mathbb{N} : (j', k, \mathtt{KeyGen}) \in \mathcal{I}\}$
24 $\mathcal{C}_{\mathsf{ip}} \leftarrow \{((\min\{k', k''\}, \max\{k', k''\}), c_{i',j'}) \mid \exists i', j' \in \mathbb{N} : (i', k', \mathtt{Enc}) \in \mathcal{I}$
$\qquad \wedge (j', k'', \mathtt{KeyGen}) \in \mathcal{I} \wedge ((i', j'), c_{i',j'}) \in \mathcal{C}'_{\mathsf{ip}}\}$
25 $(\mathsf{sk}, st_{S_{\mathrm{A}}}) \leftarrow S_{\mathrm{A},3}(\mathcal{C}_{\mathsf{ip}}, \mathcal{T}, st_{S,\mathrm{A}})$
26 **return** $(\mathsf{sk}, st_{\mathrm{S_B}} = (st_{\mathrm{S_A}}, (a_i)_{1\le i\le \ell}), i, j, k, \mathcal{I}))$

Fig. 21. Simulator for SIM-security of FH-IPBFE[NZIP-AC-IPBFRE].

The simulator stores in $\mathcal{I}$ for each query the type (Enc or KeyGen query), a global query index and a type-specific query index. From this, the simulator can construct the set $\mathcal{T}$ of tag vectors for the underlying NZIP-AC-IPBFRE scheme, since all Enc queries are answered with an NZIP-AC-IPBFRE ciphertext with $\mathbf{v}_{(a_i)_{1\leq i\leq \ell}}$ as tag vector and all KeyGen queries are answered with an NZIP-AC-IPBFRE ciphertext with $\mathbf{u}_{(a_i)_{1\leq i\leq \ell}}$ as tag vector. Furthermore, for all pairs of vectors $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ was queried with Enc and $\mathbf{y}$ was queried with KeyGen, the simulator gets in in $\mathcal{C}'_{\mathsf{ip}}$ the function evaluation $g(\langle \mathbf{x}, \mathbf{y}\rangle)$. It stores these in $\mathcal{C}_{\mathsf{ip}}$, but switching from the query-type specific indexing used in the SIM security game for FH-IPBFE to the global indexing used in the SIM security game for NZIP-AC-IPBFRE. For pairs of vectors where both vectors are queried in Enc or both are queried in KeyGen, the access policy is not satisfied by Lemma 1 and therefore the simulator does not to add anything corresponding to these pairs to $\mathcal{C}_{\mathsf{ip}}$.

With this, we see

$$\Pr[\mathbf{G}^{\text{sim-1}}_{\mathsf{FH\text{-}IPBFE}[\mathsf{NZIP\text{-}AC\text{-}IPBFRE}],\mathrm{A}}(\lambda) \Rightarrow 1] = \Pr[\mathbf{G}^{\text{sim-1}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE},\mathrm{B}}(\lambda) \Rightarrow 1]$$

and

$$\Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{FH\text{-}IPBFE}[\mathsf{NZIP\text{-}AC\text{-}IPBFRE}],\mathrm{A},\mathrm{S_A}}(\lambda) \Rightarrow 1] = \Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{NZIP\text{-}AC\text{-}IPBFRE},\mathrm{B},\mathrm{S_B}}(\lambda) \Rightarrow 1] \ .$$

Subtracting yields the result.

B TinyFHIPFE Security

Unlike Shoup's original model [47], we adopt an equivalent variant of the generic bilinear group model also previously used by Kim et al. [36]. In the generic group model, access to group elements of the ciphertexts and function keys is provided via "handles" representing them. Thus to facilitate group operations, the adversary has access to generic bilinear group model oracles $\textsc{Op}, \textsc{Pair}, \textsc{ZT}$ which provide group operation, pairing operation, and zero testing respectively. This is in addition to the encryption and key generation oracles EncO and KeyGenO provided in the simulation security games.

We will prove security by first presenting the security games and then a simulator in the generic bilinear group model. Then, we will argue that an adversary cannot distinguish between the games given that our simulator is used in the sim-0 game given in Fig. 23. To make this argument, we must analyze how the simulator answers zero-test queries and bound the difference in the outputs by using Schwartz-Zippel lemma.

In Fig. 22 and Fig. 23, we present the security games for simulation-based security of TinyFHIPFE in the generic bilinear group model. In the security games, L_1, L_2, L_t are lists of pairs, with operation append. For such a list L, we denote by $(x, y) \in L$ that (x, y) appears in L, and we denote by $y \in L$ that there exists x such that $(x, y) \in L$.
Before describing the security games, we introduce the formal variables used in the security games and the simulator below.

Let q be the total number of encryption and keygen queries made by an adversary A. Then, we define two sets of formal variables as follows.

$$\mathcal{T} = \left\{x_k^i, y_k^i\right\}_{i\in[q],k\in[d]} \cup \left\{s_1^i, s_2^i\right\}_{i\in[q]} \cup \{a_{1,k}, a_{2,k}\}_{k\in[d]} \cup \left\{b_{m,n}^i, {b^*}_{m,n}^i\right\}_{m,n\in[4]} \cup \left\{v_1^i, v_2^i, w_1^i, w_2^i\right\}_{i\in[q]}$$

$$\mathcal{R} = \left\{\phi_k^i\right\}_{i\in[q],k\in[d]} \cup \left\{\gamma_k^i\right\}_{i\in[q],k\in[d]} \cup \left\{\mu_m^i\right\}_{i\in[q],m\in[4]} \cup \left\{\psi_m^i\right\}_{i\in[q],m\in[4]}$$

These variables model the following components of the scheme:

- x_k^i: k^{th} component of the i^{th} query input vector $\mathbf{x}_i$ when made to KeyGen.
- y_k^i: k^{th} component of the i^{th} query input vector $\mathbf{y}_i$ when made to Enc.
- s_1^i, s_2^i: components of the uniform random vector $\mathbf{s}_i$ sampled during $\mathsf{Enc}(\mathbf{y}^i)$ or $\mathsf{KeyGen}(\mathbf{x}^i)$.
- $a_{1,k}, a_{2,k}$: components of the k^{th} column of the secret key uniform random matrix $\mathbf{A}$ sampled during Setup.
- $b_{m,n}, {b^*}_{m,n}$: $(m,n)^{th}$ components of the secret key matrices $\mathbf{B}, \mathbf{B}^*$ sampled during Setup.
- v_1^i, v_2^i: components of the vector $\mathbf{A}\mathbf{y}^i$ calculated inside $\mathsf{Enc}(\mathbf{y}^i)$.

- w_1^i, w_2^i: components of the vector $\mathbf{A}\mathbf{x}^i + \mathbf{A}\mathbf{A}^T\mathbf{s}_i$ calculated inside $\mathsf{KeyGen}(\mathbf{x}^i)$.
- ϕ_k^i: k^{th} component of the vector $\mathbf{A}^T\mathbf{s}_i + \mathbf{x}^i$ calculated inside $\mathsf{KeyGen}(\mathbf{x}^i)$. $\boldsymbol{\phi^i}$ refers to the vector of ϕ_k^i for $k \in [d]$.
- γ_k^i: k^{th} component of the vector $\mathbf{A}^T\mathbf{s}_i + \mathbf{y}^i$ calculated inside $\mathsf{Enc}(\mathbf{y}^i)$. $\boldsymbol{\gamma^i}$ refers to the vector of γ_k^i for $k \in [d]$.
- μ_m^i: m^{th} component of the vector $\mathbf{B} \cdot (\mathbf{s}_i \| \mathbf{A}\mathbf{x}^i + \mathbf{A}\mathbf{A}^T\mathbf{s}_i)$ calculated inside $\mathsf{KeyGen}(\mathbf{x}_i)$.
- ψ_m^i: m^{th} component of the vector $\mathbf{B}^* \cdot (\mathbf{A}\mathbf{y}^i \| \mathbf{s}_i)$ calculated inside $\mathsf{Enc}(\mathbf{y}_i)$.

Note: Given that $\mathbf{B}^* = (\mathbf{B}^{-1})^T$, $b^*_{m,n}$ can be specified as follows:

$$b^*_{m,n} = (-1)^{(m+n)} \det(\mathbf{B}_{m,n})$$

Here, $\mathbf{B}_{m,n}$ is the 3×3 matrix formed by removing the m^{th} row and n^{th} column of the matrix $\mathbf{B}$. Next, the determinant of a 3×3 matrix $\mathbf{C}$ is given by the following formula:

$$\det(\mathbf{C}) = \sum_{\sigma \in S_3} \epsilon(\sigma) \cdot c_{1,\sigma(1)} \cdot c_{2,\sigma(2)} \cdot c_{3,\sigma(3)}$$

Here, S_3 is all permutations of $\{1, 2, 3\}$, $\epsilon(\sigma)$ gives the sign of the corresponding term (1 if even permutation and -1 if odd permutation). Finally, $b^*_{m,n}$ can be written in terms of $\{b_{m,n}\}_{m,n \in [d]}$ as follows:

$$b^*_{m,n} = \sum_{\sigma \in S'_3} sgn(\sigma, m, n) \cdot b_{m_1,\sigma(m_1)} \cdot b_{m_2,\sigma(m_2)} \cdot b_{m_3,\sigma(m_3)}$$

Here, $m_1, m_2, m_3 \neq m$ and $\in [4]$ and S'_3 is all bijections from m_1, m_2, m_3 to $m_4, m_5, m_6 \neq n$ and $\in [4]$. We collapse the sign of the monomial into a function sgn for which the explicit specification will be not be necessary for the proof.

Theorem 13. $\mathsf{TinyFHIPFE}$ *is simulation-secure in the generic bilinear group model. In particular, for every adversary* A *that makes* q_t *total queries to its oracles, there exists a simulator* S *such that*

$$\mathbf{Adv}^{\text{sim-ggm}}_{\mathsf{IPFRE},\mathrm{A},\mathrm{S}}(\lambda) \leq \frac{(4q_t(d+4))^2}{p} \ .$$

Now, we analyze the advantage of the adversary which is given by the following equation:

$$\mathbf{Adv}^{\text{sim-ggm}}_{\mathsf{IPFE},\mathrm{A},\mathrm{S}}(\lambda) = \Pr[\mathbf{G}^{\text{sim-1}}_{\mathsf{IPFE},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{IPFE},\mathrm{A},\mathrm{S}}(\lambda) \Rightarrow 1] \tag{6}$$

First, in Fig. 24, we provide a simulator for the game described above. Due to space constraints in the figure, we describe part of one of the simulator's algorithm S_6 in text below.

Game $\mathbf{G}^{\text{sim-1}}_{\mathsf{IPFE},\mathrm{A}}(\lambda)$

MAIN:
$L_1, L_2, L_t \leftarrow \emptyset$ // Initialize lists
$H \leftarrow \emptyset$
$(p, g_1, g_2, g_t, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_t, \mathsf{e}) \leftarrow\!\!\$\ \mathcal{G}_{\mathsf{asym}}(1^\lambda)$
$h_1, h_2, h_t \leftarrow\!\!\$\ \text{NEWHANDLE}(1^\lambda)$
$L_1.\mathsf{append}(g_1, h_1)$
$L_2.\mathsf{append}(g_2, h_2)$
$L_t.\mathsf{append}(g_t, h_t)$
$\mathsf{pp} \leftarrow (p, h_1, h_2, h_t)$
$(d, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_1^{\text{OP}(\cdot,\cdot,\cdot),\text{PAIR}(\cdot,\cdot),\text{ZT}(\cdot,\cdot)}(\mathsf{pp})$
$\mathbf{B} \leftarrow\!\!\$\ \mathbf{SL}_4(\mathbb{Z}_p)$
$\mathbf{B}^* \leftarrow (\mathbf{B}^{-1})^T$
$\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_p^{2\times d}$
$\mathsf{msk} \leftarrow (\mathbf{A}, \mathbf{B}, \mathbf{B}^*)$
$b \leftarrow\!\!\$\ \mathrm{A}_2^{\text{ENCO}(\cdot),\text{KEYGENO}(\cdot),\text{OP}(\cdot,\cdot,\cdot),\text{PAIR}(\cdot,\cdot),\text{ZT}(\cdot,\cdot)}(st_{\mathrm{A}})$
return b

KEYGENO($\mathbf{x}$):
$\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_p^2$
$\mathsf{ct}_1 \leftarrow \mathbf{As} + \mathbf{x}$
$\mathsf{ct}_2 \leftarrow \mathbf{B} \cdot (\mathbf{s} \| \mathbf{Ax} + \mathbf{AA}^T\mathbf{s})$
ADDLIST($L_1, \mathbf{s}$)
$\mathbf{h}_1 \leftarrow \text{ADDLIST}(L_1, \mathsf{ct}_1)$
$\mathbf{h}_2 \leftarrow \text{ADDLIST}(L_1, \mathsf{ct}_2)$
$\mathsf{ct}_{\mathbf{x}} \leftarrow (\mathbf{h}_1, \mathbf{h}_2)$
return $\mathsf{sk}_{\mathbf{x}}$

ENCO($\mathbf{y}$):
$\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_p^2$
$\mathsf{sk}_1 \leftarrow \mathbf{As} + \mathbf{y}$
$\mathsf{sk}_2 \leftarrow \mathbf{B}^* \cdot (\mathbf{Ay} \| \mathbf{s})$
ADDLIST($L_2, \mathbf{s}$)
$\mathbf{h}_1 \leftarrow \text{ADDLIST}(L_2, \mathsf{ct}_1)$
$\mathbf{h}_2 \leftarrow \text{ADDLIST}(L_2, \mathsf{ct}_2)$
$\mathsf{ct}_{\mathbf{x}} \leftarrow (\mathbf{h}_1, \mathbf{h}_2)$
return $\mathsf{ct}_{\mathbf{x}}$

OP(h_1, h_2, b):
if $b = 1$ **then** $L \leftarrow L_1$
if $b = 2$ **then** $L \leftarrow L_2$
else $L \leftarrow L_t$
Find x_1 s.t. $(x_1, h_1) \in L$
Find x_2 s.t. $(x_2, h_2) \in L$
$x_3 \leftarrow x_1 + x_2$
$h_3 \leftarrow\!\!\$\ \{0,1\}^\lambda$
$L.\mathsf{append}(x_3, h_3)$
return (h_3)

PAIR(h_1, h_2):
Find x_1 s.t. $(x_1, h_1) \in L_1$
Find x_2 s.t. $(x_2, h_2) \in L_2$
$x_3 \leftarrow x_1 \cdot x_2$
$h_3 \leftarrow\!\!\$\ \{0,1\}^\lambda$
$L_t.\mathsf{append}(x_3, h_3)$
return (h_3)

ZT(h, b):
if $b = 1$ **then** $L \leftarrow L_1$
if $b = 2$ **then** $L \leftarrow L_2$
else $L \leftarrow L_t$
  Find x s.t. $(x, h) \in L$
return $(x = 0)$

ADDLIST($L, \mathbf{x}$):
$k \leftarrow |\mathbf{x}|$
for $i \in [k]$ **do**
  $h_i \leftarrow\!\!\$\ \text{NEWHANDLE}(1^\lambda)$
  $L.\mathsf{append}(\mathbf{x}[i], h)$
return $(h_1, \ldots, h_k)$

NEWHANDLE(1^λ):
$h \leftarrow\!\!\$\ \{0,1\}^\lambda \backslash H$
$H \leftarrow H \cup \{h\}$
return h

Fig. 22. Game defining simulation security of TinyFHIPFE in the generic group model.

The polynomial decomposition used by S_6 is as follows:

$$p = \sum_{i,j \in [q]} \left(c_{i,j} \left(\left(\sum_{k \in [d]} \phi_k^i \gamma_k^j - \sum_{m \in [4]} \mu_m^i \ \psi_m^j \right) - z_{i,j} \right) + f^{i,j} \right) \tag{7}$$

Game $\mathbf{G}^{\text{sim-0}}_{\mathsf{IPFE},\mathrm{A}}(\lambda)$

MAIN:
1 $\mathcal{C}_{\mathsf{ip}} \leftarrow \emptyset;\ i \leftarrow 0$
2 $E, K \leftarrow \emptyset$
3 $(\mathsf{pp}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}_1(1^\lambda)$
4 $(d, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_1^{\mathrm{OP}(\cdot,\cdot,\cdot),\mathrm{PAIR}(\cdot,\cdot),\mathrm{ZT}(\cdot,\cdot)}(1^\lambda, \mathsf{pp})$
5 $st_{\mathrm{S}} \leftarrow\!\!\$\ \mathrm{S}_2(1^d, st_{\mathrm{S}})$
6 $b \leftarrow\!\!\$\ \mathrm{A}_2^{\mathrm{ENCO}(\cdot),\mathrm{KEYGENO}(\cdot),\mathrm{OP}(\cdot,\cdot,\cdot),\mathrm{PAIR}(\cdot,\cdot),\mathrm{ZT}(\cdot,\cdot)}(st_{\mathrm{A}})$
7 **return** b

KEYGENO($\mathbf{x}$):
8 $i \leftarrow i+1;\ K \leftarrow K \cup \{i\}$
9 $\mathbf{x}_i \leftarrow \mathbf{x}$
10 **for** $j \le i$ **do**
11 **if** $j \in E$: $c_{i,j} \leftarrow g(\langle \mathbf{x}, \mathbf{y}_j \rangle)$
12 **else** : $c_{i,j} \leftarrow \perp$
13 $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}_{\mathsf{ip}} \cup \{((i,j), c_{i,j})\}$
14 $(\mathsf{sk}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}_3(\mathcal{C}_{\mathsf{ip}}, st_{\mathrm{S}})$
15 **return** sk

ENCO($\mathbf{y}$):
16 $i \leftarrow i+1;\ E \leftarrow E \cup \{i\}$
17 $\mathbf{y}_i \leftarrow \mathbf{y}$
18 **for** $j \le i$ **do**
19 **if** $j \in K$: $c_{i,j} \leftarrow g(\langle \mathbf{x}_j, \mathbf{y} \rangle)$
20 **else** : $c_{i,j} \leftarrow \perp$
21 $\mathcal{C}_{\mathsf{ip}} \leftarrow \mathcal{C}_{\mathsf{ip}} \cup \{((i,j), c_{i,j})\}$
22 $(\mathsf{ct}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}_4(\mathcal{C}_{\mathsf{ip}}, st_{\mathrm{S}})$
23 **return** ct

OP(h_1, h_2, b):
24 **return** $\mathrm{S}_5(h_1, h_2, b)$

PAIR(h_1, h_2):
25 **return** $\mathrm{S}_6(h_1, h_2)$

ZT(h, b):
26 **return** $\mathrm{S}_7(h, b)$

Fig. 23. Game defining simulation security of TinyFHIPFE in the generic group model.

where $z_{i,j}$ is the inner product value $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$, $c_{i,j}$ is the scalar coefficient of the first term of $\mu^i_m\ \psi^j_m$ which is $\mu^i_1 \psi^j_1$ ($c_{i,j}$ could be 0), and $f^{i,j}$ consists of all remaining terms in terms of $\{\phi^i_k, \gamma^j_k, \mu^i_m, \psi^j_m\}$ for $k \in [d], m \in [4]$.

1. If $f^{i,j}$ consists of any terms, then return "false".
2. Else return "true".

Lemma 3. *If an adversary interacting with the two games* $\mathbf{G}^{sim\text{-}1}_{\mathsf{IPFE},\mathrm{A}}(\lambda)$ *and* $\mathbf{G}^{sim\text{-}0}_{\mathsf{IPFE},\mathrm{A},\mathrm{S}}(\lambda)$ *is only provided access to* ENCO, KEYGENOOP, PAIR *oracles (and*

Simulator $\mathrm{S}^{\text{sim-1}}_{\mathsf{IPFE},\mathrm{A}}(\lambda)$

S_1:
1 $\mathcal{C}_{\mathsf{ip}} \leftarrow \emptyset$; $i \leftarrow 0$
2 $L_1, L_2, L_t \leftarrow \emptyset$
3 $(p, g_1, g_2, g_t, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_t, \mathsf{e}) \leftarrow\!\!\$\ \mathcal{G}_{\mathsf{asym}}(1^\lambda)$
4 $g_1, g_2, g_t \leftarrow\!\!\$\ \textsc{NewHandle}(1^\lambda)$
5 $\mathsf{pp} \leftarrow (g_1, g_2, g_t, p)$
6 $st_{\mathrm{S}} \leftarrow (L_1, L_2, L_t, \mathcal{C}_{\mathsf{ip}})$
7 Return $(\mathsf{pp}, st_{\mathrm{S}})$

$\mathrm{S}_2(1^d, st_{\mathrm{S}})$:
8 Return st_{S}

$\mathrm{S}_3(\mathcal{C}_{\mathsf{ip}}, st_{\mathrm{S}})$: // Key Generation
9 $i \leftarrow i + 1$
10 $\textsc{AddList}(L_1, \boldsymbol{\phi}^{\boldsymbol{i}})$
11 $\textsc{AddList}(L_1, \boldsymbol{\psi}^{\boldsymbol{i}})$
12 $\mathsf{sk}_{\mathbf{x}} \leftarrow (\boldsymbol{\phi}^{\boldsymbol{i}}, \boldsymbol{\mu}^{\boldsymbol{i}})$
13 **return** $\mathsf{sk}_{\mathbf{x}}$

$\mathrm{S}_4(\mathcal{C}_{\mathsf{ip}}, st_{\mathrm{S}})$: // Encryption
14 $i \leftarrow i + 1$
15 $\textsc{AddList}(L_2, \boldsymbol{\gamma}^{\boldsymbol{i}})$
16 $\textsc{AddList}(L_2, \boldsymbol{\psi}^{\boldsymbol{i}})$
17 $\mathsf{ct}_{\mathbf{y}} \leftarrow (\boldsymbol{\gamma}^{\boldsymbol{i}}, \boldsymbol{\psi}^{\boldsymbol{i}})$
18 **return** $\mathsf{ct}_{\mathbf{y}}$

$\mathrm{S}_5(h_1, h_2, b)$: // Group Operation
19 **if** $b = 1$ **then** $L \leftarrow L_1$
20 **if** $b = 2$ **then** $L \leftarrow L_2$
21 **else** $L \leftarrow L_t$
22 Find x_1 s.t. $(x_1, h_1) \in L$
23 Find x_2 s.t. $(x_2, h_2) \in L$
24 $x_3 \leftarrow x_1 + x_2$
25 $h_3 \leftarrow\!\!\$\ \{0,1\}^\lambda$
26 $L.\mathsf{append}(x_3, h_3)$
27 **return** (h_3)

$\mathrm{S}_6(h_1, h_2)$: // Pairing Operation
28 Find x_1 s.t. $(x_1, h_1) \in L_1$
29 Find x_2 s.t. $(x_2, h_2) \in L_2$
30 $x_3 \leftarrow x_1 \cdot x_2$
31 $h_3 \leftarrow\!\!\$\ \{0,1\}^\lambda$
32 $L_t.\mathsf{append}(x_3, h_3)$
33 **return** (h_3)

$\mathrm{S}_7(h, b)$: // Zero Test
34 **if** $b = 1$ **or** $b = 2$ **then**
35 Find x s.t. $(x, h) \in L_b$
36 **if** $x = 0$ **then return** true
37 **else return** false
38 **if** $b = 0$ **then**
39 Find x s.t. $(x, h) \in L_t$
40 **if** $x = 0$ **then return** true
41 **else return** // according to equation 7

$\textsc{AddList}(L, \mathbf{x})$: // $\mathbf{x}$ is a vector of polynomials
42 $k \leftarrow |\mathbf{x}|$
43 **for** $i \in [k]$ **do**
44 $h_i \leftarrow\!\!\$\ \textsc{NewHandle}(1^\lambda)$
45 $L.\mathsf{append}(\mathbf{x}[i], h)$
46 **return** $(h_1, \ldots, h_k)$

$\textsc{NewHandle}(1^\lambda)$:
47 $h \leftarrow\!\!\$\ \{0,1\}^\lambda \backslash H$
48 $H \leftarrow H \cup \{h\}$
49 **return** h

Fig. 24. Simulator for simulation security of TinyFHIPFE in the generic group model.

not ZT*), then it cannot distinguish between the two games. Formally, consider the modified games* $\mathbf{G}'^{sim\text{-}1}_{\mathsf{IPFE},\mathrm{A}}(\lambda)$ *and* $\mathbf{G}'^{sim\text{-}0}_{\mathsf{IPFE},\mathrm{A},\mathrm{S}}(\lambda)$ *where the oracle* ZT *is omitted. Then, for any PT adversary* A *and any PT simulator* S*, we have that*

$$\mathbf{Adv}^{\text{sim-ggm}}_{\mathsf{IPFE},\mathrm{A},\mathrm{S}}(\lambda) = \Pr[\mathbf{G}'^{sim\text{-}1}_{\mathsf{IPFE},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}'^{sim\text{-}0}_{\mathsf{IPFE},\mathrm{A},\mathrm{S}}(\lambda) \Rightarrow 1] = 0$$

Proof. By inspection of the oracles, we notice that the return value is a vector of appropriate size consisting of handles. Each of these handles is sampled independently and uniformly randomly in both games. Thus, given information-

theoretically indistinguishable inputs, the adversary has exactly the same probability of returning 1 in each game. Thus, the probability of the game outputting 1 is also identical between the two games, and the advantage of the adversary is equal to 0.

Now, it suffices to consider the difference in The rest of the proof refers to the zero test oracle and analyzes return values in each game. In $\mathbf{G}^{\text{sim-1}}_{\mathsf{IPFE},\mathrm{A}}(\lambda)$, the zero test returns true when the value stored in zero. Thus, we analyze the polynomial decomposition in equation (7) that corresponds to $\mathbf{G}^{\text{sim-0}}_{\mathsf{IPFE},\mathrm{A}}(\lambda)$.

First, if the polynomial p is the identically zero polynomial, the simulator correctly responds with "zero". Next, the handles received by $\mathcal{A}$ correspond to ciphertexts which are stored in the lists L_1, L_2 with the following components from the set $\mathcal{R}$ for each $i \in [q]$:

$$\phi^i_1, \ldots, \phi^i_d,\ \gamma^i_1, \ldots, \gamma^i_d,\ \mu^i_1, \mu^i_2, \mu^i_3, \mu^i_4,\ \psi^i_1, \psi^i_2, \psi^i_3, \psi^i_4$$

Lemma 4. *For all $i \in [q]$, each of $v^i_1, v^i_2, w^i_1, w^i_2$ is not the identically zero polynomial.*

Proof. First, consider v^i_1, v^i_2 which represent components of the vector $\mathbf{A}\mathbf{y}_i$. In terms of formal variables, $v^i_1 = \sum_{k\in[d]} a_{1,k}\ y^i_k$ and $v^i_2 = \sum_{k\in[d]} a_{2,k}\ y^i_k$. Here, $a_{1,k}, a_{2,k}$ are formal variables for all $k \in [d]$. Thus, v^i_1, v^i_2 are not identically zero.

Next, we look at w^i_1, w^i_2 which represent components of the vector $\mathbf{A}\mathbf{x}_i + \mathbf{A}\mathbf{A}^T\mathbf{s}_i$. In terms of formal variables,

$$w^i_1 = \sum_{k\in[d]} a_{1,k}\ x^i_k + \left(\sum_{k\in[d]} (a_{1,k})^2\right) s^i_1 + \left(\sum_{k\in[d]} (a_{1,k}a_{2,k})\right) s^i_2$$

Here all of $s^i_1, s^i_2, a_{1,k}, a_{2,k}$ are indeterminate formal variables, thus each of the summation terms are not identically zero. Hence, the polynomial is not identically zero.

Next, if the handle h belongs to one of the source groups, the polynomial p must be some linear combination of variables in $\mathcal{R}$. In particular, each of $\phi^i_k, \gamma^i_k, \mu^i_m, \psi^i_m$ for all $i \in [q], k \in [d], m \in [4]$ consist of monomials which are a product of either two indeterminates or an indeterminate and one of $v^i_1, v^i_2, w^i_1, w^i_2$ which are not identically zero as given by Lemma 1. Thus, any linear combination of these polynomials cannot be the identically zero polynomial. The simulator correctly answers the zero-test query when the input handle is in the source group.

On the other hand, analysis of an input handle in the target group is more involved. We consider all the possible pairing for the handles that $\mathcal{A}$ has access to.

In the target group, we know that $f^{i,j}$ consists of a linear combination of all possible pairings. First, we expand each of the pairing to show that they are not

the identically zero polynomial. Next, we show that any leftover combination in $f^{i,j}$ can also not be identically zero.

$$\begin{aligned}
\phi_k^i \cdot \gamma_\ell^j &= \left(a_{1,k}\ s_1^i + a_{2,k}\ s_2^i + x_k^i\right) \cdot \left(a_{1,\ell}\ s_1^j + a_{2,\ell}\ s_2^j + y_\ell^j\right) \\
\phi_k^i \cdot \psi_n^j &= \left(a_{1,k}\ s_1^i + a_{2,k}\ s_2^i + x_k^i\right) \cdot \\
&\qquad \left({b^*}_{n,1}\ v_2^j + {b^*}_{n,2}\ v_2^j + {b^*}_{n,3}\ s_1^j + {b^*}_{n,4}\ s_2^j\right) \\
\mu_m^i \cdot \gamma_\ell^j &= \left(b_{m,1}\ s_1^i + b_{m,2}\ s_2^i + b_{m,3}\ w_1^i + b_{m,4}\ w_2^i\right) \cdot \\
&\qquad \left(a_{1,\ell}\ s_1^j + a_{2,\ell}\ s_2^j + y_\ell^j\right) \\
\mu_m^i \cdot \psi_n^j &= \left(b_{m,1}\ s_1^i + b_{m,2}\ s_2^i + b_{m,3}\ w_1^i + b_{m,4}\ w_2^i\right) \cdot \\
&\qquad \left({b^*}_{n,1}\ v_1^j + {b^*}_{n,2}\ v_2^j + {b^*}_{n,3}\ s_1^j + {b^*}_{n,4}\ s_2^j\right)
\end{aligned}$$

For further evaluation, we include the degree of all monomials in each handle and subsequent pairings with respect to the formal variables $\{s, a, b\}$ which can be derived from the polynomial expressions above.

- ϕ_k^i: degree 2
- μ_m^i: degree 2 and degree 4
- γ_k^i: degree 2
- ψ_m^i: degree 4
- $\phi_k^i \cdot \gamma_\ell^j$: degree 4
- $\phi_k^i \cdot \psi_n^j$: degree 6
- $\mu_m^i \cdot \gamma_\ell^j$: degree 4 and 6
- $\mu_m^i \cdot \psi_n^j$: degree 6 and 8

Case I: $f^{i,j}$ consists of terms from a pairing of the form $\mu_m^i \psi_n^j$ without loss of generality in the indices i and j.

First, notice that the degree of monomials coming from this pairing have degree either 6 or 8 in the formal variables, in particular all monomials have degree 4 in variables $\{b_{m,n}\}_{m,n\in[4]}$ since each monomial consists of the product $b_{m,n} \cdot b^*_{m',n'}$ for some indices $m, n, m', n' \in [4]$. However, in Case II we will see that other monomials of degree 6 or 8 do not have degree 4 in $\{b_{m,n}\}_{m,n\in[4]}$ and thus cannot be combined with monomials in this pairing to form the identically zero polynomial. Thus, it suffices to partition the cases into this one where there are pairings of the $\mu_m^i \psi_n^j$ (and potentially other pairings) in $f^{i,j}$ and case II with no pairings of the $\mu_m^i \psi_n^j$.
Next, we analyze the monomials within the pairing and argue that they cannot be combined to form the identically zero polynomial either. Crucial to this analysis is

the fact that $f^{i,j}$ does not include the pairings $\psi_1^i \mu_1^j$ and $\psi_1^j \mu_1^i$ since the first part of the polynomial p includes all instances of them by factoring out its coefficient $c_{i,j}$. This fact leads to the argument that $f^{i,j}$ cannot include the entire sum $\gamma^{i,j}$ which would "cancel out" b and b^* variables. Below, we show a detailed proof of this argument.

Here, consider the expansion of $\mu_m^i \psi_n^j$. For simplicity let $y^i = (s_1^i, s_2^i, w_1^i, w_2^i)$ and $z^j = (v_1^j, v_2^j, s_1^j, s_2^j)$. We have:

$$\begin{aligned} \mu_m^i \ \psi_n^j &= \left(\sum_{k_1 \in [4]} b_{m,k_1} y_{k_1}^i \right) \left(\sum_{k_2 \in [4]} b_{n,k_2}^* z_{k_2}^j \right) \\ &= \sum_{k_1,k_2 \in [4]} \left(b_{n,k_2} b_{m,k_1}^* y_{k_1}^i z_{k_2}^j \right) \end{aligned}$$

As given before, b_{m,k_1}^* can be written as follows:

$$b_{m,k_1}^* = \sum_{\sigma \in S_3'} sgn(\sigma, m, k_1) \cdot b_{m_1,\sigma(m_1)} b_{m_2,\sigma(m_2)} b_{m_3,\sigma(m_3)}$$

Here, S_3' consists of bijections from $m_1, m_2, m_3 \in [4]$ and $m_1, m_2, m_3 \neq m$ to $m_4, m_5, m_6 \in [4]$ and $m_4, m_5, m_6 \neq k_1$ and $sgn(\sigma)$ gives the sign of the corresponding term.

Sub-case (a): $f^{i,j}$ consists of terms from a pairing $\mu_m^i \psi_n^j$ where $m \neq n$.

In the expansion above, each monomial inside the sums consists of four b variables, one from μ_n^i and the other 3 from the product given by the expansion of b^* variable in ψ_n^j. Note that the monomial does not consist of $b_{m,k'}$ for any $k' \in [4]$. However, for every $m' \in [4], m' \neq m$, the analogous monomial must consist of $b_{m,\sigma(m)}$ in the inside product. Thus, these monomials cannot cancel out any monomials from $\psi_{m'}^i \mu_n^j$ where $m' \neq n$.
Next, when $m \neq n$, $b_{n,\sigma(n)}$ is part of b_{m,k_1}^* and thus the monomial consists of two variables from the same row n, namely b_{n,k_2} and $b_{n,\sigma(n)}$. On the other hand, if $m = n$, each monomial consists of exactly one variable from each row of the matrix. Thus, monomials from $\mu_m^i \psi_n^j$ when $m \neq n$ cannot be canceled out by monomials in the case $m = n$. Thus, in this case $f^{i,j}$ cannot be identically zero.

Sub-case (b): $f^{i,j}$ does not consist of terms from a pairing $\mu_m^i \psi_n^j$ where $m \neq n$.

First, recall that $\mu_m^i \psi_n^j$ cannot be in $f^{i,j}$ by construction, thus we consider $\mu_m^i \psi_n^j$ where $m \in \{2, 3, 4\}$. Among all the monomials in the sum, there exists a monomial that contains both b_{m,k_2} and b_{1,k_2}. However, there is no monomial that contains both b_{m',k_2} and b_{1,k_2} for any $m' \in [4], m' \neq m$. Thus, each pairing contains a monomial with both b_{m,k_2} and b_{1,k_2} in the product which cannot be canceled by any other pairing. Thus, $f^{i,j}$ cannot be identically zero in this case either.

Notice that the above argument does not apply when $k_1 = k_2$ since $\sigma(r) \neq k_1$. In that case, each monomial consists of $b_{m,k}$ but cannot contain $b_{m',k}$ for any $m' \in [4], m' \neq m$. This applies to all $m \in [4]$ which means that monomials from pairings $m' \neq m$ contain $b_{m',k}$ but not $b_{m,k}$. Thus, these two types of monomials cannot cancel each other out and thus $f^{i,j}$ is not identically zero.

Case II: $f^{i,j}$ does not consist of terms from a pairing of the form $\mu^i_m \psi^j_n$.

Here, it suffices to analyze the monomials that have the same degree since monomials that have a different degree in the formal variables cannot cancel each other out in any linear combinations to form the identically zero polynomial.

First, consider all monomials of degree 4. There are two sources for these monomials: $\phi^i_k \cdot \gamma^j_\ell$ and $\mu^i_m \cdot \gamma^j_n$. Below in Table 4 for each pairing, we give an example of monomial of degree 4 in the polynomial expansion. The other monomials differ only in indices and we provide the total number of such monomials.

monomial origin	$\deg(s)$	$\deg(a)$	$\deg(b)$	example
$\phi^i_k \cdot \gamma^j_\ell$	2	2	0	$a_{1,k} s^i_1 a_{1,\ell} s^j_1$
$\mu^i_m \cdot \gamma^j_n$	2	1	1	$b_{m,1} s^i_1 a_{1,\ell} s^j_1$
$\mu^i_m \cdot \gamma^j_n$	1	2	1	$b_{m,1} w^i_1 a_{1,\ell} s^j_1$

Table 4. Degree 4 monomials.

Similarly, consider monomials of degree 6. There are two sources for these monomials: $\phi^i_k \cdot \psi^j_m$ and $\mu^i_m \cdot \gamma^j_n$. Below in Table 5 for each pairing, we give an example of monomial of degree 6 in the polynomial expansion. The other monomials differ only in indices.

monomial origin	$\deg(s)$	$\deg(a)$	$\deg(b)$	example
$\phi^i_k \cdot \psi^j_m$	1	2	3	$a_{1,k} s^i_1 b^*_{m,1} v^j_1$
$\phi^i_k \cdot \psi^j_m$	2	1	3	$a_{1,k} s^i_1 b^*_{m,1} s^j_1$
$\mu^i_m \cdot \gamma^j_\ell$	2	3	1	$b_{m,3} w^i_1 a_{1,\ell} s^j_1$

Table 5. Degree 6 monomials.

In all the cases above, we can see that even though the monomials are of the same degree, the specific set of formal variables forming the polynomials are different for each type of monomial. Thus, there can be no relationship between these monomials and they cannot be combined to get the identically zero polynomial.

Finally, in Case II without pairings of the form $\psi_m^i \mu_n^j$, the polynomial $f^{i,j}$ cannot be the identically zero polynomial.

On the other hand, when $f^{i,j}$ is empty, we can simplify p as follows:

$$p = \sum_{i,j \in [q]} c_{i,j} \left(\left(\sum_{k \in [d]} \phi_k^i \gamma_k^j - \sum_{m \in [4]} \mu_m^i \ \psi_m^j \right) - z_{i,j} \right)$$

Then using the expansion of each polynomial above and the relationship between $\mathbf{B}$ and $\mathbf{B}^*$:

$$p = \sum_{i,j \in [q]} c_{i,j} \left(\left(\sum_{k \in [d]} x_k^i y_k^j \right) - z_{i,j} \right)$$

Now, the simulator knows that $z_{i,j}$ is the inner product of query vectors i and j, thus:

$$p = \sum_{i,j \in [q]} c_{i,j} \left(z_{i,j} - z_{i,j} \right) = 0$$

Thus, the simulator answers correctly for an honest evaluation.

Lemma 5. *Schwartz-Zippel Lemma: For a non-zero polynomial $P \in R[x_1, \ldots, x_n]$ of total degree $d \geq 0$ over an integral domain R. Let S be a finite subset of R and let $r_1, \ldots, r_n$ be selected at random independently and uniformly from S. Then*

$$\Pr[P(r_1, \ldots, r_n) = 0] \leq \frac{d}{|S|} \ .$$

Let q_t be the total number of queries made by the adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ to all of its oracles. We can split q into number of queries made to various oracles as follows:

1. Encryption Oracle $\textsc{EncO}(\cdot)$: q_{enc}
2. Key Generation Oracle $\textsc{EncO}(\cdot)$: q_{kg}
3. Group Operation Oracle $\textsc{Op}(\cdot, \cdot, \cdot)$: q_{op}
   - Group Operation Oracle in the first source group $\textsc{Op}(\cdot, \cdot, 1)$: $q_{op,1}$
   - Group Operation Oracle in the second source group $\textsc{Op}(\cdot, \cdot, 2)$: $q_{op,2}$
   - Group Operation Oracle in the target group $\textsc{Op}(\cdot, \cdot, 0)$: $q_{op,t}$
4. Bilinear Pairing Oracle $\textsc{Pair}(\cdot, \cdot)$: q_{pair}
5. Zero Test Oracle $\textsc{Op}(\cdot, \cdot, \cdot)$: q_{zt}
   - Zero Test Oracle in the first source group$\mathrm{ZT}(\cdot, 1)$: $q_{zt,1}$
   - Zero Test Oracle in the second source group$\mathrm{ZT}(\cdot, 2)$: $q_{zt,2}$
   - Zero Test Oracle in the target group$\mathrm{ZT}(\cdot, 0)$: $q_{zt,t}$

First, each Group Operation and Bilinear Pairing Oracle query returns one handle while an Encryption and Key Generation Oracle query returns $d+4$ handles. Now in each source group, we have the maximum possible degree of a polynomial is 4. Thus, by the Schwartz-Zippel Lemma, for each Zero Test Oracle query A makes, it has probability at most $\frac{4}{p}$ of the oracle returning true in sim-1 where it would not in the sim-0 game.
Similarly, in order to test linear combinations of polynomials, A takes each handle h and for every other handle h_i also in the same source group, runs the following queries: $\mathrm{ZT}(\mathrm{OP}(h, h_i, 1))$ or $\mathrm{ZT}(\mathrm{OP}(h, h_i, 2))$. Again, by Schwartz-Zippel Lemma, there is probability $\frac{4}{p}$ of the oracle returning true in sim-1 game and false in sim-0.
Then, we take union bound on the probabilities above to bound the probability that at least one Zero Test Oracle query returns true as follows:

$$\frac{4}{p}\left(q_{zt,1}+q_{zt,2}+\binom{q_{kg}(d+4)+q_{op,1}}{2}+\binom{q_{enc}(d+4)+q_{op,2}}{2}\right).$$

Next, in the target group, we have the maximum possible degree of a polynomial is 8. First, for each Zero Test Oracle query A makes, it has probability at most $\frac{8}{p}$ of the oracle returning true in sim-1 where it would not in the sim-0 game.
In order to test linear combinations of polynomials, A again takes each handle h and for every other handle h_i also in the target group, runs the following queries: $\mathrm{ZT}(\mathrm{OP}(h, h_i, 0))$. Again, by Schwartz-Zippel Lemma, there is probability $\frac{8}{p}$ of the oracle returning true in sim-1 game and false in sim-0.
Then, we take union bound on these probabilities:

$$\frac{8}{p}\left(q_{zt,t}+\binom{q_{op,t}+q_{pair}}{2}\right).$$

Now, let's add the above probabilities together to calculate the probability of distinguishing games sim-1 and sim-0 as follows:

$$\begin{aligned}
&\frac{4}{p}\left(q_{zt,1}+q_{zt,2}+\binom{q_{kg}(d+4)+q_{op,1}}{2}+\binom{q_{enc}(d+4)+q_{op,2}}{2}\right)\\
&+\frac{8}{p}\left(q_{zt,t}+\binom{q_{op,t}+q_{pair}}{2}\right)\\
&=\frac{1}{p}\left(4q_{zt,1}+4q_{zt,2}+4\binom{q_{kg}(d+4)+q_{op,1}}{2}+4\binom{q_{enc}(d+4)+q_{op,2}}{2}\right.\\
&\left.+8q_{zt,t}+8\binom{q_{op,t}+q_{pair}}{2}\right)\\
&\leq\frac{1}{p}\left(8q_{zt}+4((q_{op}+q_{pair}+(d+4)(q_{enc}+q_{kg}))^2)\right)\\
&\leq\frac{(4q_t(d+4))^2}{p}\quad\left(\leq\frac{\mathcal{O}(q_t^2)}{p}\right).
\end{aligned}$$

This concludes the proof. ∎

C Our Outsourced Database Protocol for ANN Search

In this Section, we first define ANN data structure and ANN outsourced database protocol. We adopt the notion of outsourced database (ODB) system given by [15]; however, we modify the formalism to cover ANN and dynamic databases.

C.1 ANN Data Structures

A database $\mathcal{D}$ is abstracted as a collection of n vectors $\mathbf{x}$ from $\mathbb{Z}_p^d$, each associated with an unique identifier $\mathsf{ID} \in \mathbb{N}$. We say the database has a lookup operation that on input of an identifier ID, returns the associated vector $\mathbf{x}$. We denote this operation as $\mathbf{x} \leftarrow \mathcal{D}[\mathsf{ID}]$. We require the ANN data structure to use a distance metric dist, which is black-box computable from inner-product (*e.g.* Euclidean).

A query is the defined as a vector $\mathbf{q}$ from $\mathbb{Z}_p^d$, which is associated with an unique identifier $\mathsf{ID}_{\mathbf{q}} \in \mathbb{N}$. An ANN data structure $\mathcal{DS}_{\mathsf{ANN}}$ consists of three algorithms, namely $\mathcal{DS}_{\mathsf{ANN}} = (\mathsf{Initialize}, \mathsf{Insert}, \mathsf{Search})$. In this work, we consider ANN algorithms which only require inner product comparison in order to insert and search the database. This can be formalized by providing the algorithms oracle access to an inner product comparison operation named $\textsc{CompO}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{ID}_3)$ which on input three identifiers outputs whether $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle > \langle \mathbf{x}_1, \mathbf{x}_3 \rangle$ where $\mathbf{x}_i \leftarrow \mathcal{D}[\mathsf{ID}_i]$. Thus, we rewrite the ANN data structure as follows:

$\mathsf{Initialize}(p, d, \mathsf{params})$: On input a prime p, dimension d, and a set of parameters params, the $\mathsf{Initialize}$ algorithm initializes the data structure $\mathcal{DS}_{\mathsf{ANN}}$.

$\mathsf{Insert}^{\textsc{CompO}(\cdot,\cdot,\cdot)}(\mathsf{ID})$: On input a record identifier ID, the insert algorithm with access to an inner-product comparison oracle $\textsc{CompO}$, inserts the record identifier in the data structure $\mathcal{DS}_{\mathsf{ANN}}$. This algorithm does not have a return value.

$\mathsf{Search}^{\textsc{CompO}(\cdot,\cdot,\cdot)}(\mathsf{ID}_{\mathbf{q}})$: On input a query identifier $\mathsf{ID}_{\mathbf{q}}$, the search algorithm with access to an inner-product comparison oracle $\textsc{CompO}$, outputs an ID from the data structure $\mathcal{DS}_{\mathsf{ANN}}$.

Correctness. We require search correctness of the ANN data structure. The ANN data structure is said to be $(1-\delta)$-correct, if for all databases $\mathcal{D}$ of size $n \in \mathbb{N}$ we have the following:

$$\Pr[\mathsf{ID} \leftarrow \mathsf{Search}^{\textsc{CompO}(\cdot,\cdot,\cdot)}(\mathsf{ID}_{\mathbf{q}}) : \mathsf{dist}(\mathbf{q}, \mathbf{x}_i) \leq \mathsf{dist}(\mathbf{q}, \mathbf{x}_j)\ \forall j \in [n]] \geq 1 - \delta$$

where $\mathcal{DS}_{\mathsf{ANN}} \leftarrow \mathsf{Initialize}(p, d, \mathsf{params})$; $\mathsf{Insert}^{\textsc{CompO}(\cdot,\cdot,\cdot)}(\mathsf{ID}_i) \forall i \in [n]$ is run before the search and the probability is over the coins of the insertion algorithm.

C.2 ANN Outsourced Database Protocols

An *ANN outsourced database protocol* $\mathsf{ANN\text{-}ODB}$ for a database $\mathcal{D} = \{(\mathbf{x}_i, \mathsf{ID}_i)_{i \in [n]}\}$, where each $\mathbf{x}_i \in \mathbb{Z}_p^d$ and $\mathsf{ID}_i \in \mathbb{N}$ consists of five algorithms $\mathsf{ANN\text{-}ODB} = (\mathsf{Setup}, \mathsf{PreQuery}, \mathsf{Query}, \mathsf{PreUpdate}, \mathsf{Update})$:

- $\mathsf{Setup}(1^\lambda, 1^n, \mathcal{D})$: On input a unary encoding of the security parameter λ, the size of the database n, and a database $\mathcal{D} = \{(\mathbf{x}_i, \mathsf{ID}_i)_{i\in[n]}\}$, the setup algorithm outputs an ANN data structure $\mathcal{DS}_{\mathsf{ANN}}$ and a database of ciphertexts $\mathcal{D}_S = \{(\mathsf{ct}_i, \mathsf{ID}_i)\}_{i\in[n]}$.
- $\mathsf{PreQuery}(\mathbf{q}, \mathsf{ID}_\mathbf{q})$: On input a query vector $\mathbf{q}$ and its identifier $\mathsf{ID}_\mathbf{q}$, the preprocess query algorithm algorithm outputs a ciphertext of the query vector $\mathsf{ct}_\mathbf{q}$.
- $\mathsf{Query}(\mathcal{DS}_{\mathsf{ANN}}, (\mathsf{ct}_\mathbf{q}, \mathsf{ID}_\mathbf{q}))$: On input an ANN data structure $\mathcal{DS}_{\mathsf{ANN}}$ and a tuple of query ciphertext and identifier $(\mathsf{ct}_\mathbf{q}, \mathsf{ID}_\mathbf{q})$, the query algorithm outputs an identifier ID representing the nearest neighbor of the query vector.
- $\mathsf{PreUpdate}(\mathbf{u}, \mathsf{ID}_\mathbf{u})$: On input an update vector $\mathbf{u}$ and its identifier $\mathsf{ID}_\mathbf{u}$, the preprocess update algorithm outputs a ciphertext of the update vector $\mathsf{ct}_\mathbf{u}$.
- $\mathsf{Update}(\mathcal{DS}_{\mathsf{ANN}}, (\mathsf{ct}_\mathbf{u}, \mathsf{ID}_\mathbf{u}))$: On input an ANN data structure $\mathcal{DS}_{\mathsf{ANN}}$ and a tuple of update ciphertext and identifier $(\mathsf{ct}_\mathbf{u}, \mathsf{ID}_\mathbf{u})$, the update algorithm has no output; the data structure is updated internally.

Correctness. We define Query correctness and Update correctness respectively.

Query **correctness:** For all $\lambda, n, \mathsf{ID}_\mathbf{q} \in \mathbb{N}$, $\mathbf{q} \in \mathbb{Z}_p^d$, and for any database $\mathcal{D}$, we say that the ANN-ODB satisfies query correctness if the following two outputs are identical:

$$\mathsf{Query}(\mathcal{DS}_{\mathsf{ANN}}, (\mathsf{ct}_\mathbf{q}, \mathsf{ID}_\mathbf{q})),\ \mathcal{DS}_{\mathsf{ANN}}.\mathsf{Search}^{\textsc{Compo}(\cdot,\cdot,\cdot)}(\mathsf{ID}_\mathbf{q})$$

where $(\mathcal{DS}_{\mathsf{ANN}}, \mathcal{DS}_S) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^n, \mathcal{D})$ and $\mathsf{ct}_\mathbf{q} \leftarrow\!\!\$\ \mathsf{PreQuery}(\mathbf{q}, \mathsf{ID}_\mathbf{q})$.

Update **correctness:** For all $\lambda, n, \mathsf{ID}_\mathbf{u} \in \mathbb{N}$, $\mathbf{u} \in \mathbb{Z}_p^d$, and for any database $\mathcal{D}$, we say that the ANN-ODB satisfies query correctness if the $\mathcal{DS}_{\mathsf{ANN}}$ is identical after running either of the following:

$$\mathsf{Update}(\mathcal{DS}_{\mathsf{ANN}}, (\mathsf{ct}_\mathbf{u}, \mathsf{ID}_\mathbf{u})),\ \mathcal{DS}_{\mathsf{ANN}}.\mathsf{Insert}^{\textsc{Compo}(\cdot,\cdot,\cdot)}(\mathsf{ID}_\mathbf{u})$$

where $(\mathcal{DS}_{\mathsf{ANN}}, \mathcal{DS}_S) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^n, \mathcal{D})$ and $\mathsf{ct}_\mathbf{u} \leftarrow\!\!\$\ \mathsf{PreQuery}(\mathbf{u}, \mathsf{ID}_\mathbf{u})$.

Security. We give a simulation-based security definition for ANN-ODB. First, we define a leakage profile LP. Let ANN-ODB = (Setup, PreQuery, Query, PreUpdate, Update) be a dynamic ODB system for a database $\mathcal{D}$ with data and query vectors from $\mathbb{Z}_p^d$. A leakage profile $\mathsf{LP} = (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Query}}, \mathcal{L}_{\mathsf{Update}})$ is a tuple of algorithms such that:

Setup Leakage $\mathcal{L}_{\mathsf{Setup}}$: On input a database of records $\mathcal{D} = \{(\mathbf{x}_i, \mathsf{ID}_i)\}_{i\in[n]}$ and an insertion order insert-order, it outputs a string ℓ_{Setup}.

Query Leakage $\mathcal{L}_{\mathsf{Query}}$: On input a database $\mathcal{D} = \{(\mathbf{x}_i, \mathsf{ID}_i)\}_{i\in[n]}$ and a set of previous queries $\mathcal{Q} = \{(\mathbf{q}_i, \mathsf{ID}_i)_{i\in[m]}\}$, it outputs a string ℓ_{Query}.

Update Leakage $\mathcal{L}_{\mathsf{Update}}$: On input a database $\mathcal{D} = \{(\mathbf{x}_i, \mathsf{ID}_i)\}_{i\in[n]}$, a set of previous queries $\mathcal{Q} = \{(\mathbf{q}_i, \mathsf{ID}_i)_{i\in[m]}\}$, and a record $(\mathbf{x}, \mathsf{ID})$, it outputs a string ℓ_{Update}.

The advantage of an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ interacting with the games given in Fig. 25 for $\lambda \in \mathbb{N}$ with respect to a simulator $\mathrm{S} = (\mathsf{SimSetup}, \mathsf{SimQuery}, \mathsf{SimUpdate})$ and a leakage profile $\mathsf{LP} = (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Query}}, \mathcal{L}_{\mathsf{Update}})$ is as follows:

$$\mathsf{Adv}^{\mathrm{sim}}_{\mathsf{ANN\text{-}ODB},\mathrm{A},\mathrm{S},\mathsf{LP}}(\lambda) = \Pr[\mathbf{G}^{\text{sim-1}}_{\mathsf{ANN\text{-}ODB},\mathrm{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{G}^{\text{sim-0}}_{\mathsf{ANN\text{-}ODB},\mathrm{A},\mathrm{S},\mathsf{LP}}(\lambda) \Rightarrow 1] \ .$$

We say that an ANN-ODB scheme is *SIM-secure* under a leakage profile $\mathsf{LP} = (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Query}}, \mathcal{L}_{\mathsf{Update}})$ if for every PT adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ and for all $\lambda \in \mathbb{N}$, there exists a PT simulator $\mathrm{S} = (\mathsf{SimSetup}, \mathsf{SimQuery}, \mathsf{SimUpdate})$ such that $\mathsf{Adv}^{\mathrm{sim}}_{\mathsf{ANN\text{-}ODB},\mathrm{A},\mathrm{S},\mathsf{LP}}(\lambda)$ is negligible.

Game $\mathbf{G}^{\text{sim-1}}_{\mathsf{ANN\text{-}ODB},\mathrm{A}}(\lambda)$

MAIN:

$b \leftarrow \bot$
$(\mathcal{D}, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, 1^n)$
$(\mathcal{DS}_{\mathsf{ANN}}, \mathcal{D}_S) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^n, \mathcal{D})$
$(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st) \leftarrow\!\!\$\ \mathrm{A}_2(\mathcal{DS}, \mathcal{D}_S, st)$
while $\mathsf{op}_{\mathsf{type}} \neq \mathsf{terminate}$:
  if $\mathsf{op}_{\mathsf{type}} = \mathsf{Query}$:
    $(\mathbf{q}, \mathsf{ID}) \leftarrow \mathsf{op}$
    $(\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}) \leftarrow \mathsf{PreQuery}(\mathbf{q}, \mathsf{ID})$
    $(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st) \leftarrow\!\!\$\ \mathrm{A}_2(\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}, st)$
  if $\mathsf{op}_{\mathsf{type}} = \mathsf{Update}$:
    $(\mathbf{u}, \mathsf{ID}) \leftarrow \mathsf{op}$
    $(\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}) \leftarrow \mathsf{PreUpdate}(\mathbf{u}, \mathsf{ID})$
    $(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st) \leftarrow\!\!\$\ \mathrm{A}_2(\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}, st)$
if $\mathsf{op}_{\mathsf{type}} = \mathsf{terminate}$: $b \leftarrow \mathsf{op}$
return b

Game $\mathbf{G}^{\text{sim-0}}_{\mathsf{ANN\text{-}ODB},\mathrm{A},\mathrm{S}}(\lambda)$

MAIN:

$b \leftarrow \bot$
$\mathcal{Q} \leftarrow \emptyset$
$(\mathcal{D}, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, 1^n)$
$\ell_{\mathsf{Setup}} \leftarrow \mathcal{L}_{\mathsf{Setup}}(\mathcal{D})$
$(\mathcal{DS}_{\mathsf{ANN}}, \mathcal{D}_S, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}.\mathsf{SimSetup}(\ell_{\mathsf{Setup}})$
$(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_2(\mathcal{DS}, \mathcal{D}_S, st_{\mathrm{A}})$
while $\mathsf{op}_{\mathsf{type}} \neq \mathsf{terminate}$:
  if $\mathsf{op}_{\mathsf{type}} = \mathsf{Query}$:
    $(\mathbf{q}, \mathsf{ID}) \leftarrow \mathsf{op}$
    $\ell_{\mathsf{Query}} \leftarrow \mathcal{L}_{\mathsf{Query}}(\mathcal{D}, \mathcal{Q}, (\mathbf{q}, \mathsf{ID}))$
    $(\mathsf{ct}_{\mathbf{q}}, st_{\mathrm{S}}) \leftarrow \mathrm{S}.\mathsf{SimQuery}(\ell_{\mathsf{Query}}, st_{\mathrm{S}})$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathbf{q}, \mathsf{ID})\}$
    $(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_2(\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}, st_{\mathrm{A}})$
  if $\mathsf{op}_{\mathsf{type}} = \mathsf{Update}$:
    $(\mathbf{u}, \mathsf{ID}) \leftarrow \mathsf{op}$
    $\ell_{\mathsf{Update}} \leftarrow \mathcal{L}_{\mathsf{Update}}(\mathcal{D}, \mathcal{Q}, (\mathbf{u}, \mathsf{ID}))$
    $(\mathsf{ct}_{\mathbf{u}}, st_{\mathrm{S}}) \leftarrow \mathrm{S}.\mathsf{SimUpdate}(\ell_{\mathsf{Update}}, st_{\mathrm{S}})$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{u}, \mathsf{ID})\}$
    $(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st) \leftarrow\!\!\$\ \mathrm{A}_2(\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}, st_{\mathrm{A}})$
if $\mathsf{op}_{\mathsf{type}} = \mathsf{terminate}$: $b \leftarrow \mathsf{op}$
return b

Fig. 25. Games defining SIM-security for ANN-ODB.

C.3 AC-IPFRE[FH-IPFE] Security

Theorem 14. *Our* AC-IPFRE[FH-IPFE] *scheme satisfies SIM-security, assuming that the underlying* FH-IPFE *satisfies SIM-security.*

Proof. Let A be an adversary against AC-IPFRE[FH-IPFE]. We first construct an adversary B against SIM-security of FH-IPFE in Fig. 26.

Next, by SIM-security of FH-IPBFE, for an adversary B there exists a simulator $\mathrm{S_B} = (\mathrm{S_{B,1}}, \mathrm{S_{B,2}}, \mathrm{S_{B,3}}, \mathrm{S_{B,4}})$. Then, consider the simulator $S_\mathrm{A} = (S_{\mathrm{A},1}, S_{\mathrm{A},2}, S_{\mathrm{A},3})$ for AC-IPFRE[FH-IPFE] for an adversary A also given in Fig. 26.

The simulator $S_{\mathrm{A},3}$ gets the collection $\mathcal{C}'_{\mathsf{ip}}$ that contains 'accessible' inner products of AC-IPFRE[FH-IPFE] ciphertexts with each other using a global index to keep track of the queries and $\mathcal{T}$ that contains the tags of each ciphertext. The simulator turns this into $\mathrm{M_{ip}}$ that still uses a global index but keeps track of (Enc or KeyGen) queries. Since a ciphertext with tag C uses both a ciphertext and function key, the simulator uses an offset δ to keep track of the indices. When adding new tuples to $\mathcal{C}_{\mathsf{ip}}$, the simulator must keep track of ciphertexts of tag C separately and only add new values right before running $\mathrm{S_{B,3}}$ or $\mathrm{S_{B,4}}$ with an updated $\mathcal{C}_{\mathsf{ip}}$ as given in simulator lines 15-30.

With this, we see

$$\Pr[\mathbf{G}^{\text{sim-1}}_{\text{AC-IPFRE[FH-IPFE]},\mathrm{A}}(\lambda) \Rightarrow 1] = \Pr[\mathbf{G}^{\text{sim-1}}_{\text{FH-IPFE},\mathrm{B}}(\lambda) \Rightarrow 1]$$

and

$$\Pr[\mathbf{G}^{\text{sim-0}}_{\text{AC-IPFRE[FH-IPFE]},\mathrm{A},\mathrm{S_A}}(\lambda) \Rightarrow 1] = \Pr[\mathbf{G}^{\text{sim-0}}_{\text{FH-IPFE},\mathrm{B},\mathrm{S_B}}(\lambda) \Rightarrow 1] \ .$$

Subtracting yields the result.

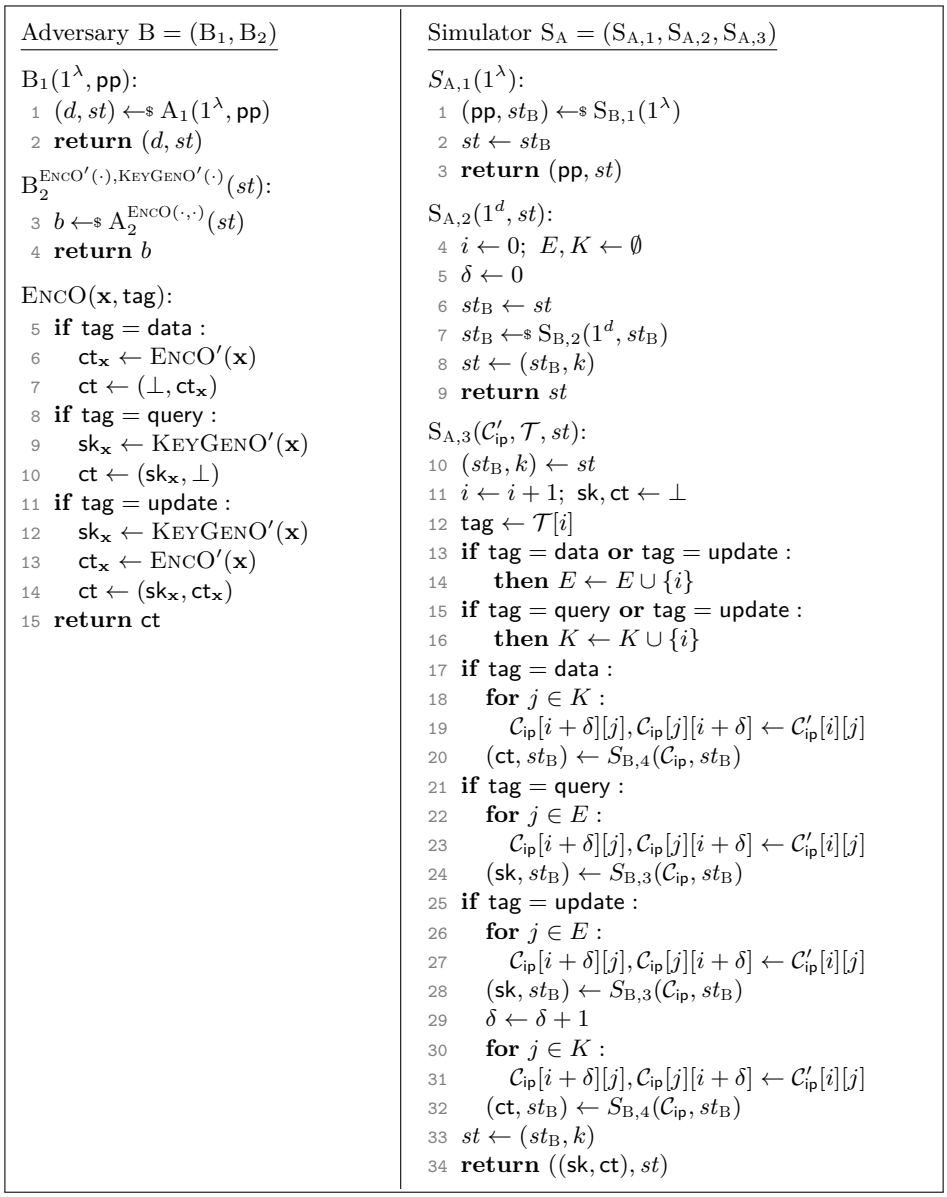
C.4 ANN-ODB Security Proof

Theorem 15. *Our ANN ODB protocol* ANN-ODB[AC-IPFRE] *is SIM-secure under the leakage profile* $\mathsf{LP}_{\mathsf{ANN}}\text{-ODB} = (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Query}}, \mathcal{L}_{\mathsf{Update}})$ *given in Fig. 13, assuming that* AC-IPFRE *is a SIM-secure ACIPFRE for access policy* a_{ann}.

Consider the games given in Fig. 27.

Now, we consider an adversary $\mathrm{B} = (\mathrm{B}_1, \mathrm{B}_2)$ interacting with the games given in Fig. 3 using an adversary $\mathrm{A} = (\mathrm{A}_1, \mathrm{A}_2)$ against the games given in Fig. 27. This adversary is given in Fig. 28.
We also consider a simulator $\mathrm{S} = (\mathsf{SimPPGen}, \mathsf{SimSetup}, \mathsf{SimQuery}, \mathsf{SimUpdate})$ interacting with the game given in Fig. 27 (right) that uses an AC-IPFRE simulator $\mathrm{S}_{\text{AC-IPFRE}} = (\mathsf{SimPPGen}, \mathsf{SimSetup}, \mathsf{SimEnc})$ for the access policy a_{ann}. This simulator is given in Fig. 29.

We can see that when the adversary B is the sim-1 AC-IPFRE game, it perfectly simulates the sim-1 ANN-ODB[AC-IPFRE] game for the adversary A. On the other hand, when the adversary B is in the sim-0 AC-IPFRE game with a simulator $\mathrm{S}_{\text{AC-IPFRE}}$, it simulates the sim-0 ANN-ODB[AC-IPFRE] game for the adversary A with the simulator S given in Fig. 29 (right). Thus, ANN-ODB[AC-IPFRE] is persistent simulation secure under the given leakage profile.



Adversary $\mathrm{B} = (\mathrm{B}_1, \mathrm{B}_2)$

$\mathrm{B}_1(1^\lambda, \mathsf{pp})$:
1 $(d, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, \mathsf{pp})$
2 **return** (d, st)

$\mathrm{B}_2^{\textsc{EncO}'(\cdot),\textsc{KeyGenO}'(\cdot)}(st)$:
3 $b \leftarrow\!\!\$\ \mathrm{A}_2^{\textsc{EncO}(\cdot,\cdot)}(st)$
4 **return** b

$\textsc{EncO}(\mathbf{x}, \mathsf{tag})$:
5 **if** $\mathsf{tag} = \mathsf{data}$:
6 $\quad \mathsf{ct}_{\mathbf{x}} \leftarrow \textsc{EncO}'(\mathbf{x})$
7 $\quad \mathsf{ct} \leftarrow (\perp, \mathsf{ct}_{\mathbf{x}})$
8 **if** $\mathsf{tag} = \mathsf{query}$:
9 $\quad \mathsf{sk}_{\mathbf{x}} \leftarrow \textsc{KeyGenO}'(\mathbf{x})$
10 $\quad \mathsf{ct} \leftarrow (\mathsf{sk}_{\mathbf{x}}, \perp)$
11 **if** $\mathsf{tag} = \mathsf{update}$:
12 $\quad \mathsf{sk}_{\mathbf{x}} \leftarrow \textsc{KeyGenO}'(\mathbf{x})$
13 $\quad \mathsf{ct}_{\mathbf{x}} \leftarrow \textsc{EncO}'(\mathbf{x})$
14 $\quad \mathsf{ct} \leftarrow (\mathsf{sk}_{\mathbf{x}}, \mathsf{ct}_{\mathbf{x}})$
15 **return** ct

Simulator $\mathrm{S}_\mathrm{A} = (\mathrm{S}_{\mathrm{A},1}, \mathrm{S}_{\mathrm{A},2}, \mathrm{S}_{\mathrm{A},3})$

$S_{\mathrm{A},1}(1^\lambda)$:
1 $(\mathsf{pp}, st_\mathrm{B}) \leftarrow\!\!\$\ \mathrm{S}_{\mathrm{B},1}(1^\lambda)$
2 $st \leftarrow st_\mathrm{B}$
3 **return** (pp, st)

$\mathrm{S}_{\mathrm{A},2}(1^d, st)$:
4 $i \leftarrow 0;\ E, K \leftarrow \emptyset$
5 $\delta \leftarrow 0$
6 $st_\mathrm{B} \leftarrow st$
7 $st_\mathrm{B} \leftarrow\!\!\$\ \mathrm{S}_{\mathrm{B},2}(1^d, st_\mathrm{B})$
8 $st \leftarrow (st_\mathrm{B}, k)$
9 **return** st

$\mathrm{S}_{\mathrm{A},3}(\mathcal{C}'_{\mathsf{ip}}, \mathcal{T}, st)$:
10 $(st_\mathrm{B}, k) \leftarrow st$
11 $i \leftarrow i + 1;\ \mathsf{sk}, \mathsf{ct} \leftarrow \perp$
12 $\mathsf{tag} \leftarrow \mathcal{T}[i]$
13 **if** $\mathsf{tag} = \mathsf{data}$ **or** $\mathsf{tag} = \mathsf{update}$:
14 $\quad$ **then** $E \leftarrow E \cup \{i\}$
15 **if** $\mathsf{tag} = \mathsf{query}$ **or** $\mathsf{tag} = \mathsf{update}$:
16 $\quad$ **then** $K \leftarrow K \cup \{i\}$
17 **if** $\mathsf{tag} = \mathsf{data}$:
18 $\quad$ **for** $j \in K$:
19 $\quad\quad \mathcal{C}_{\mathsf{ip}}[i+\delta][j], \mathcal{C}_{\mathsf{ip}}[j][i+\delta] \leftarrow \mathcal{C}'_{\mathsf{ip}}[i][j]$
20 $\quad (\mathsf{ct}, st_\mathrm{B}) \leftarrow S_{\mathrm{B},4}(\mathcal{C}_{\mathsf{ip}}, st_\mathrm{B})$
21 **if** $\mathsf{tag} = \mathsf{query}$:
22 $\quad$ **for** $j \in E$:
23 $\quad\quad \mathcal{C}_{\mathsf{ip}}[i+\delta][j], \mathcal{C}_{\mathsf{ip}}[j][i+\delta] \leftarrow \mathcal{C}'_{\mathsf{ip}}[i][j]$
24 $\quad (\mathsf{sk}, st_\mathrm{B}) \leftarrow S_{\mathrm{B},3}(\mathcal{C}_{\mathsf{ip}}, st_\mathrm{B})$
25 **if** $\mathsf{tag} = \mathsf{update}$:
26 $\quad$ **for** $j \in E$:
27 $\quad\quad \mathcal{C}_{\mathsf{ip}}[i+\delta][j], \mathcal{C}_{\mathsf{ip}}[j][i+\delta] \leftarrow \mathcal{C}'_{\mathsf{ip}}[i][j]$
28 $\quad (\mathsf{sk}, st_\mathrm{B}) \leftarrow S_{\mathrm{B},3}(\mathcal{C}_{\mathsf{ip}}, st_\mathrm{B})$
29 $\quad \delta \leftarrow \delta + 1$
30 $\quad$ **for** $j \in K$:
31 $\quad\quad \mathcal{C}_{\mathsf{ip}}[i+\delta][j], \mathcal{C}_{\mathsf{ip}}[j][i+\delta] \leftarrow \mathcal{C}'_{\mathsf{ip}}[i][j]$
32 $\quad (\mathsf{ct}, st_\mathrm{B}) \leftarrow S_{\mathrm{B},4}(\mathcal{C}_{\mathsf{ip}}, st_\mathrm{B})$
33 $st \leftarrow (st_\mathrm{B}, k)$
34 **return** $((\mathsf{sk}, \mathsf{ct}), st)$

Fig. 26. Adversary against SIM-security of FH-IPFE and simulator for SIM security of AC-IPFRE[FH-IPFE].

Game $\mathbf{G}^{\text{sim-1}}_{\mathsf{ANN\text{-}ODB[AC\text{-}IPFRE]},\mathrm{A}}(\lambda)$

MAIN:

$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{AC\text{-}IPFRE.PPGen}(1^\lambda)$
$(\mathcal{D}_{\mathsf{init}}, st) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, 1^n, 1^d, \mathsf{pp})$
$\mathsf{msk} \leftarrow \mathsf{AC\text{-}IPFRE.Setup}(1^\lambda, 1^d)$
$\mathcal{D} \leftarrow \mathsf{Shuffle}(\mathcal{D}_{\mathsf{init}})$
for $i \in [n]$:
  $\mathcal{DS}_{\mathsf{ANN}}.\mathsf{Insert}^{\textsc{Compo}(\cdot,\cdot,\cdot)}(\mathsf{ID}_i)$
  $\mathsf{ct}_i \leftarrow\!\!\$\ \mathsf{AC\text{-}IPFRE.Enc}(\mathsf{msk}, \mathsf{data}, \mathcal{D}[\mathsf{ID}_i])$
  $\mathcal{D}_S[\mathsf{ID}_i] \leftarrow \mathsf{ct}_i$
$(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st) \leftarrow\!\!\$\ \mathrm{A}_2\left(\mathcal{DS}_{\mathsf{ANN}}, (\mathsf{ct}_i)_{i\in[n]}, st\right)$
while $\mathsf{op}_{\mathsf{type}} \neq \mathsf{terminate}$:
  if $\mathsf{op}_{\mathsf{type}} = \mathsf{Query}$:
    $(\mathbf{q}, \mathsf{ID}) \leftarrow \mathsf{op}$
    $\mathsf{ct}_{\mathbf{q}} \leftarrow\!\!\$\ \mathsf{AC\text{-}IPFRE.Enc}(\mathsf{msk}, \mathsf{query}, \mathbf{q})$
    $(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st) \leftarrow\!\!\$\ \mathrm{A}_3((\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}), st)$
  if $\mathsf{op}_{\mathsf{type}} = \mathsf{Update}$:
    $(\mathbf{u}, \mathsf{ID}) \leftarrow \mathsf{op}$
    $\mathsf{ct}_{\mathbf{u}} \leftarrow\!\!\$\ \mathsf{AC\text{-}IPFRE.Enc}(\mathsf{msk}, \mathsf{update}, \mathbf{u})$
    $(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st) \leftarrow\!\!\$\ \mathrm{A}_4((\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}), st)$
if $\mathsf{op}_{\mathsf{type}} = \mathsf{terminate} : b \leftarrow \mathsf{op}$
return b

$\mathsf{Shuffle}(\mathcal{D}_{\mathsf{old}})$:

$\mathcal{D}_{\mathsf{new}} \leftarrow \emptyset$
$n \leftarrow \mathsf{length}(\mathcal{D}_{\mathsf{old}})$
$\sigma \leftarrow\!\!\$\ \mathsf{S}_n$
for $i \in [n]$:
  $\mathcal{D}_{\mathsf{new}}[\mathsf{ID}_i] \leftarrow \mathcal{D}_{\mathsf{old}}[\mathsf{ID}_{\sigma(i)}]$
return $\mathcal{D}_{\mathsf{new}}$

$\textsc{Compo}(\mathsf{ID}_i, \mathsf{ID}_j, \mathsf{ID}_k)$:

$\mathbf{x}_i \leftarrow \mathcal{D}[\mathsf{ID}_i]; \mathbf{x}_j \leftarrow \mathcal{D}[\mathsf{ID}_j]; \mathbf{x}_k \leftarrow \mathcal{D}[\mathsf{ID}_k]$
if $\langle \mathbf{x}_i, \mathbf{x}_j \rangle > \langle \mathbf{x}_i, \mathbf{x}_k \rangle$:
  return 1
else :
  return 0

Game $\mathbf{G}^{\text{sim-0}}_{\mathsf{ANN\text{-}ODB[AC\text{-}IPFRE]},\mathrm{A},\mathrm{S}}(\lambda)$

MAIN:

$(\mathsf{pp}, st_{\mathrm{S}}) \leftarrow\!\!\$\ \mathrm{S}.\mathsf{SimPPGen}(1^\lambda)$
$(\mathcal{D}_{\mathsf{init}}, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_1(1^\lambda, 1^n, 1^d, \mathsf{pp})$
$\mathcal{D} \leftarrow \mathsf{Shuffle}(\mathcal{D}_{\mathsf{init}})$
$\ell_{\mathsf{Setup}} \leftarrow \mathcal{L}_{\mathsf{Setup}}(\mathcal{D}, \perp)$
$(\mathcal{DS}_{\mathsf{ANN}}, \mathcal{D}_S) \leftarrow\!\!\$\ \mathrm{S}.\mathsf{SimSetup}(\ell_{\mathsf{Setup}}, st_{\mathrm{S}})$
$(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_2\ (\mathcal{DS}_{\mathsf{ANN}}, \mathcal{D}_S, st_{\mathrm{A}})$
while $\mathsf{op}_{\mathsf{type}} \neq \mathsf{terminate}$:
  if $\mathsf{op}_{\mathsf{type}} = \mathsf{Query}$:
    $(\mathbf{q}, \mathsf{ID}) \leftarrow \mathsf{op};\ \mathcal{Q}[\mathsf{ID}] \leftarrow \mathbf{q}$
    $\ell_{\mathsf{Query}} \leftarrow \mathcal{L}_{\mathsf{Query}}((\mathbf{q}, \mathsf{ID}), \mathcal{D})$
    $(\mathsf{ct}_{\mathbf{q}}, st_{\mathrm{S}}) \leftarrow \mathrm{S}.\mathsf{SimQuery}(\ell_{\mathsf{Query}}, st_{\mathrm{S}})$
    $(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_3((\mathsf{ct}_{\mathbf{q}}, \mathsf{ID}), st_{\mathrm{A}})$
  if $\mathsf{op}_{\mathsf{type}} = \mathsf{Update}$:
    $(\mathbf{u}, \mathsf{ID}) \leftarrow \mathsf{op};\ \mathcal{D}[\mathsf{ID}] \leftarrow \mathbf{u}$
    $\ell_{\mathsf{Update}} \leftarrow \mathcal{L}_{\mathsf{Update}}((\mathbf{u}, \mathsf{ID}), \mathcal{D}, \mathcal{Q})$
    $(\mathsf{ct}_{\mathbf{u}}, st_{\mathrm{S}}) \leftarrow \mathrm{S}.\mathsf{SimUpdate}(\ell_{\mathsf{Update}}, st_{\mathrm{S}})$
    $(\mathsf{op}, \mathsf{op}_{\mathsf{type}}, st_{\mathrm{A}}) \leftarrow\!\!\$\ \mathrm{A}_4((\mathsf{ct}_{\mathbf{u}}, \mathsf{ID}), st_{\mathrm{A}})$
if $\mathsf{op}_{\mathsf{type}} = \mathsf{terminate} : b \leftarrow \mathsf{op}$
return b

$\mathsf{Shuffle}(\mathcal{D}_{\mathsf{old}})$:

$\mathcal{D}_{\mathsf{new}} \leftarrow \emptyset$
$n \leftarrow \mathsf{length}(\mathcal{D}_{\mathsf{old}})$
$\sigma \leftarrow\!\!\$\ \mathsf{S}_n$
for $i \in [n]$:
  $\mathcal{D}_{\mathsf{new}}[\mathsf{ID}_i] \leftarrow \mathcal{D}_{\mathsf{old}}[\mathsf{ID}_{\sigma(i)}]$
return $\mathcal{D}_{\mathsf{new}}$

Fig. 27. ANN-ODB[FH-IPFE] Security.

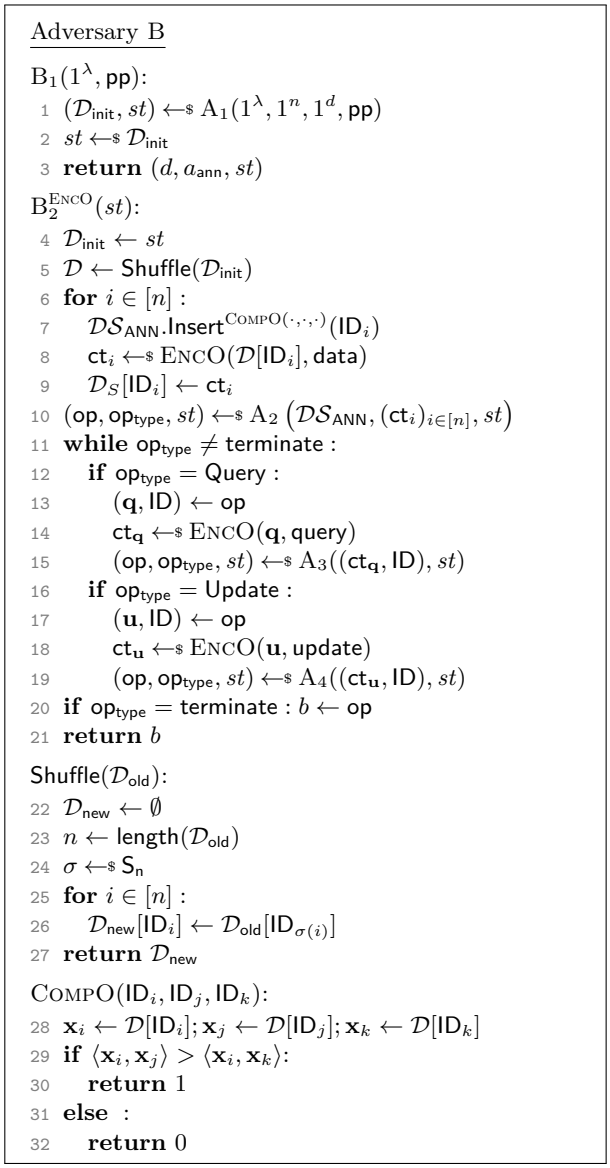


Fig. 28. Adversary for ANN-ODB[AC-IPFRE] simulation security.

Simulator S

$\mathrm{S}.\mathsf{SimPPGen}(1^\lambda)$:
1 $(\mathsf{pp}, st) \leftarrow\!\!\$\ \mathrm{S}_{\mathsf{AC\text{-}IPFRE}}.\mathsf{SimPPGen}(1^\lambda)$
2 **return** (pp, st)

$\mathrm{S}.\mathsf{SimSetup}(\ell_{\mathsf{Setup}}, st)$:
3 $\mathsf{DS}_{\mathsf{ANN}}.\mathsf{Initialize}(d, p, \mathsf{params})$
4 $(\mathsf{ip\text{-}comp}, \mathsf{insert\text{-}order}) \leftarrow \ell_{\mathsf{Setup}}$
5 $\mathcal{C}_{\mathsf{ip}}, \mathcal{T} \leftarrow \emptyset;\ i \leftarrow 0$
6 $st \leftarrow\!\!\$\ \mathrm{S}_{\mathsf{AC\text{-}IPFRE}}.\mathsf{SimSetup}(1^d, \perp, st)$
7 **for** $j \in$ insert-order :
8 $\quad \mathcal{DS}_{\mathsf{ANN}}.\mathsf{Insert}^{\textsc{Compo}(\cdot,\cdot,\cdot)}(\mathsf{ID}_j)$
9 $\quad$ **for** $k \in$ insert-order :
10 $\quad\quad \mathcal{C}_{\mathsf{ip}}[j][k], \mathcal{C}_{\mathsf{ip}}[k][j] \leftarrow \perp$
11 $\quad i \leftarrow i + 1;\ \mathcal{T}[i] \leftarrow \mathsf{data}$
12 $\quad (\mathsf{ct}_i, st) \leftarrow\!\!\$\ \mathrm{S}_{\mathsf{AC\text{-}IPFRE}}.\mathsf{SimEnc}(\mathcal{C}_{\mathsf{ip}}, \mathcal{T}, st)$
13 $\mathcal{D}_S \leftarrow \{(\mathsf{ct}_j, \mathsf{ID}_j)\}_{j\in[n]};\ \mathcal{D}_{\mathcal{Q}} \leftarrow \emptyset$
14 $st_{\mathrm{S}} \leftarrow (st, \mathcal{C}_{\mathsf{ip}}, \mathcal{T})$
15 **return** $(\mathcal{DS}_{\mathsf{ANN}}, \mathcal{D}_S, st_{\mathrm{S}})$

$\mathrm{S}.\mathsf{SimQuery}(\ell_{\mathsf{Query}}, st_{\mathrm{S}})$:
16 $\delta \leftarrow 0$
17 $(st, \mathcal{C}_{\mathsf{ip}}, \mathcal{T}) \leftarrow st_{\mathrm{S}}$
18 $i \leftarrow i + 1;\ \mathcal{T}[i] \leftarrow \mathsf{query}$
19 $\mathsf{ip\text{-}val} \leftarrow \ell_{\mathsf{Query}}$
20 **for** $j \in [i]$:
21 $\quad$ **if** $\mathcal{T}[j] = \mathsf{query}$:
22 $\quad\quad \mathcal{C}_{\mathsf{ip}}[i][j], \mathcal{C}_{\mathsf{ip}}[j][i] \leftarrow \perp;\ \delta \leftarrow \delta + 1$
23 $\quad$ **else** : $\mathcal{C}_{\mathsf{ip}}[i][j], \mathcal{C}_{\mathsf{ip}}[j][i] \leftarrow \mathsf{ip\text{-}val}[j + \delta]$
24 $(\mathsf{ct}_{\mathbf{q}}, st) \leftarrow\!\!\$\ \mathrm{S}_{\mathsf{AC\text{-}IPFRE}}.\mathsf{SimEnc}(\mathcal{C}_{\mathsf{ip}}, \mathcal{T}, st)$
25 $st_{\mathrm{S}} \leftarrow (st, \mathcal{C}_{\mathsf{ip}}, \mathcal{T})$
26 **return** $(\mathsf{ct}_{\mathbf{q}}, st_{\mathrm{S}})$

$\mathrm{S}.\mathsf{SimUpdate}(\ell_{\mathsf{Update}}, st_{\mathrm{S}})$:
27 $(st, \mathcal{C}_{\mathsf{ip}}, \mathcal{T}) \leftarrow st_{\mathrm{S}}$
28 $i \leftarrow i + 1;\ \mathcal{T}[i] \leftarrow \mathsf{update}$
29 $\mathsf{ip\text{-}val} \leftarrow \ell_{\mathsf{Update}}$
30 **for** $j \in [i]$:
31 $\quad \mathcal{C}_{\mathsf{ip}}[i][j], \mathcal{C}_{\mathsf{ip}}[j][i] \leftarrow \mathsf{ip\text{-}val}[j]$
32 $(\mathsf{ct}_{\mathbf{u}}, st) \leftarrow\!\!\$\ \mathrm{S}_{\mathsf{AC\text{-}IPFRE}}.\mathsf{SimEnc}(\mathcal{C}_{\mathsf{ip}}, \mathcal{T}, st)$
33 **return** $(\mathsf{ct}_{\mathbf{u}}, st_{\mathrm{S}})$

$\textsc{Compo}(\mathsf{ID}_i, \mathsf{ID}_j, \mathsf{ID}_k)$:
34 **return** $\mathsf{ip\text{-}comp}[\mathsf{ID}_i][\mathsf{ID}_j][\mathsf{ID}_k]$

Fig. 29. Simulator for ANN-ODB[AC-IPFRE] simulation security.