Inner-Product Functional Encryption with Fine-Grained Access Control

Michel Abdalla, Dario Catalano, Romain Gay, Bogdan Ursu

Inner-Product Functional Encryption with Fine-Grained Access Control

Michel Abdalla[1,2], Dario Catalano[3], Romain Gay[4], and Bogdan Ursu[5]

[1] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
michel.abdalla@ens.fr
[2] INRIA, Paris, France
[3] Dipartimento di Matematica e Informatica, Università di Catania, Italy
catalano@dmi.unict.it
[4] IBM Zürich, Switzerland
romain.rgay@gmail.com
[5] ETH Zürich, Switzerland
bogdan.ursu@inf.ethz.ch

Abstract. We construct new functional encryption schemes that combine the access control functionality of attribute-based encryption with the possibility of performing linear operations on the encrypted data. While such a primitive could be easily realized from fully fledged functional encryption schemes, what makes our result interesting is the fact that our schemes simultaneously achieve all the following properties. They are public-key, efficient and can be proved secure under standard and well established assumptions (such as LWE or pairings). Furthermore, security is guaranteed in the setting where adversaries are allowed to get functional keys that decrypt the challenge ciphertext. Our first results are two functional encryption schemes for the family of functions that allow users to embed policies (expressed by monotone span programs) in the encrypted data, so that one can generate functional keys to compute weighted sums on the latter. Both schemes are pairing-based and quite generic: they combine the ALS functional encryption scheme for inner products from Crypto 2016 with any attribute-based encryption schemes relying on the dual-system encryption methodology. As an additional bonus, they yield simple and elegant multi-input extensions essentially for free, thereby broadening the set of applications for such schemes. Multi-input is a particularly desirable feature in our setting, since it gives a finer access control over the encrypted data, by allowing users to associate different access policies to different parts of the encrypted data. Our second result builds identity-based functional encryption for inner products from lattices. This is achieved by carefully combining existing IBE schemes from lattices with adapted, LWE-based, variants of ALS. We point out to intrinsic technical bottlenecks to obtain richer forms of access control from lattices. From a conceptual point of view, all our results can be seen as further evidence that more expressive forms of functional encryption can be realized under standard assumptions and with little computational overhead.

1 Introduction

Public-key encryption allows the owner of a secret key sk to decrypt any ciphertext created with respect to a corresponding public key pk. At the same time, without sk, one should not be able to extract any information whatsoever about the encrypted plaintext. This all-or-nothing feature is becoming restrictive nowadays as, in many applications, a much more fine grained access control to data is required. Functional encryption addresses this need by providing an encryption mechanism where decryption keys are associated with functions. Specifically, given a ciphertext $\mathsf{Enc}(m)$ and a secret key sk_f associated to some function f, the holder of sk_f learns $f(m)$ and nothing else.

Security for functional encryption is formalized via a variant of the standard indistinguishability notion. In a nutshell, this notion states that an adversary who is allowed to see secret keys corresponding to functions $f_1, \dots f_n$ should not be able to say which of the challenge messages m_0 or m_1 has been encrypted, as long as $f_i(m_0) = f_i(m_1)$, for all i. This indistinguishability notion has been proposed in [BSW11, O'N10] and shown inadequate for certain, somewhat complex, functionalities. These authors also suggested an alternative, simulation based, security notion that however turns out to be impossible to achieve for general functionalities without introducing additional restrictions. See [BSW11, O'N10] for details.

Since its introduction, functional encryption has attracted a lot of interest. Known results can be broadly categorized as focusing on (1) feasibility results for general functionalities, and on (2) concrete, efficient realizations for restricted functionalities of practical interest. Constructions of the first type are all horrendously inefficient. Also, they either rely on quite unstable assumptions (e.g. indistinguishability obfuscation) or impose severe restrictions on the number of secret keys that can be issued. Constructions of the second type, on the other hand, are known only for the case of linear functions and quadratic functions. Over the last few years, significant research efforts have been devoted to the quest of improving these constructions along different directions. For the case of the inner-product functionality (IPFE) [ABDP15], this meant, for instance, improved security guarantees (e.g. [ALS16, ABDP16, BBL17, CLT18]), function hiding realizations (e.g. [BJK15, DDM16, DOT18]), multi-input extensions (e.g. [AGRW17, ACF+18]), decentralized schemes (e.g. [CDG+18, ABKW19, LT19, ABG19]), unbounded-size vectors (e.g. [TT18, DP19]) and specialized variants (e.g. [BCSW19]). For the case of quadratic functions, current schemes are limited to [BCFG17, Gay20] in the public-key setting. Note that FE for inner products, which is the focus of this work, can be used a building block to obtain FE for quadratic functions. This fact, implicit in [BCFG17], is made explicit in [Gay20] and in the private-key variants [AS17, Lin17].

In spite of these efforts, only a few convincing practical applications of the primitive have been proposed so far. Notable examples include the recent non interactive protocol for hidden-weight coin flips from [CS19], a practical construction of function-hiding inner product FE with applications such as in biometric authentication, nearest-neighbor search on encrypted data in [KLM+18], an application of functional encryption for quadratic functions for performing private inference on encrypted data in [RPB+19].

A possible explanation for this is that, behind its charming theoretical appearance, functional encryption hides a fragile and potentially dangerous nature: each new released secret key inherently leaks information. This becomes particularly painful for the case of inner products, as, when encrypting plaintexts of length, say, n, holding n secret keys allows, in general, to recover the full plaintext completely. While this might seem inherent in the nature of IPFE, one might wonder if additional measures might be put in place to reduce leakage and make the primitive more appealing for applications. Think for instance of the case of a medical database. To preserve privacy while maintaining the possibility of performing simple descriptive statistics (such as the weighted mean) on the data, one might decide to encrypt the database using IPFE. A drawback of this solution, however, is that the confidentiality of the whole database is compromised if a sufficiently large number of different keys is released. This is problematic since this threshold might be easy to reach when many users access the database.

A natural way to limit the inherent information leakage of existing IPFE schemes would be to use FE primitives with more sophisticated functionalities. Ideally, this primitive should allow to embed access policies in the (encrypted) data while allowing to compute weighted sums on the latter. More precisely, each key should allow to obtain the desired inner product only when some appropriate access policy is satisfied. Going back to our medical example, this means that the confidentiality of a *particular* database entry would be compromised only if sufficiently many different keys satisfying the ciphertext policy associated with that entry are released.

Another way to look at the question, is providing additional security guarantees with respect to basic identity or attribute based encryption schemes. These typically control who is authorized to decrypt the data. Still, once the data

is accessed, no additional control is possible: authorized users get the full information, while others get nothing. In this sense, it is natural to consider encryption primitives that, beyond access control, also permit to more carefully tune the information leakage.

Notice that the mechanisms above are easy to realize if one is willing to resort to functional encryption schemes for general functionalities. The trouble with this is that such a solution would be of little (if any) practical interest. Our goal, on the other hand, is to develop a scheme that implements the features above while retaining as much as possible all the nice properties of currently known IPFEs.

This motivates the following question.

Is it possible to develop an efficient, public-key, functional encryption scheme that allows users both to embed access policies in the encrypted data and to generate decryption keys to compute inner products on the data?

A trivial generic approach. Since ABE and IPFE are both well-studied primitives, the first natural question is whether we can easily combine existing schemes to achieve our target notion. In the target scheme, each ciphertext is associated with a predicate P and encrypts a vector $\vec{x}$. Each functional decryption key $\mathsf{sk}_{\vec{y},\mathsf{att}}$ is associated with an attribute att and a vector $\vec{y}$. Decryption recovers $\langle \vec{x}, \vec{y} \rangle$ if $\mathsf{P}(\mathsf{att}) = 1$. If it is not the case, no information about $\vec{x}$ should be revealed.

Now, consider the approach of encrypting a plaintext via an IPFE and then encrypting the resulting ciphertext via the ABE. This is not secure against collusions as, once the outer ciphertext is decrypted, the inner one becomes completely independent from the ABE. To see why, assume we have keys for $\mathsf{sk}_{\vec{y}_0,\mathsf{att}_0}$ and $\mathsf{sk}_{\vec{y}_1,\mathsf{att}_0}$ and a ciphertext ct, encrypting a vector $\vec{x}$ under the predicate P such that $\mathsf{P}(\mathsf{att}_0) = 1$ and $\mathsf{P}(\mathsf{att}_1) = 0$. The trivial solution allows to use $\mathsf{sk}_{\vec{y}_0,\mathsf{att}_0}$ to obtain the original IPFE ciphertext, which can then be used with $\mathsf{sk}_{\vec{y}_1,\mathsf{att}_1}$ to obtain $\langle \vec{x}, \vec{y}_1 \rangle$ (even though we should only have been able to compute $\langle \vec{x}, \vec{y}_0 \rangle$). This means that mix-and-match attacks are possible. In fact, there seems to be no trivial solution to this problem.

Another trivial generic approach. One other approach to limit the leakage is by encrypting various databases under a different IPFE public key for every recipient. Apart from the fact that this leads to a prohibitive blow-up in size, it would not be possible to agregate data between different databases. Our solution has neither of these limitations and ensures that the ciphertext size is independent of the number of potential recipients.

Our contributions. In this paper, we construct schemes for inner-product functional encryption with fine-grained access control. Our realizations are both efficient and provably secure under standard and well-established assumptions.

The key distinguishing feature of our constructions is that they can be proved secure in the, technically more challenging, setting where the adversary is allowed to (get keys to) decrypt the challenge ciphertext. Let us explain this more in detail. Popular specializations of functional encryption (such as identity-based encryption (IBE) [Sha84,BF01] and attribute-based encryption [SW05, GPSW06]) are ones where the message is interpreted as a pair (I, m), where m is the actual message (often called the "payload") and I is a string, referred to as the index (or in the context of ciphertext-policy ABE [BSW07], a predicate), that can be either public or private. For these schemes, confidentiality of the payload is guaranteed as long as no decryption keys associated with attributes that satisfy the predicate are issued. In our case, we still guarantee a meaningful security notion when keys which allow users to decrypt the payload are issued.

Private-index schemes also provide meaningful security guarantees when keys that decrypt are leaked, namely, they still hide the index in that case. However, as opposed to public-index schemes, for which we have constructions for all circuits from standard assumptions [GVW13, BGG+14], such schemes can only handle restrictive policies, that are expressed by orthogonality testing (also referred to as inner-product encryption [KSW08]), or assume a weaker security property, called *weak attribute hiding*, which limits the set of keys that the adversary can get. Namely, this property dictates that the adversary is only allowed to ask secret keys corresponding to functions that cannot be used to decrypt the challenge ciphertext. As observed in [GVW15], a fully attribute-hiding predicate encryption for circuits would bring us tantalizing close to getting indistinguishability obfuscation, which explains why they are much harder to realize in practice.

We consider both public-index schemes where policies are expressive (they can be expressed by monotone span programs, which capture Boolean formulas), and private-index schemes for orthogonality testing (which captures constant depth Boolean formulas). In both settings, we permit a fine-tuned access to the payload, which, from a

technical point of view, involve providing security even when the adversary obtains keys that decrypt the challenge ciphertext (even in the public-index case).

IP-FE WITH FINE-GRAINED ACCESS CONTROL FROM PAIRINGS. Our first main result is the construction of functional encryption schemes for the family of functions that allows users to embed policies on the encrypted data, so that one can generate decryption keys that computes weighted sums on the latter. More precisely, in our schemes, each ciphertext is associated with a predicate P and encrypts a (small norm) vector $\vec{x}$. Each functional decryption key is associated with an attribute att and a (small norm) vector $\vec{y}$. Decryption recovers $\langle \vec{x}, \vec{y} \rangle$ if att satisfies P. If this is not the case, security guarantees that no information about $\vec{x}$ is revealed.

Our constructions are quite generic and show that it is possible to combine existing pairing-based attribute-based encryption with the IPFE from [ALS16]. Our construction relies on any attribute-based encryption that uses the dual-system encryption methodology [Wat09]. In particular, we provide a modular framework that turns any ABE that supports the class of predicates $\mathcal{P}$ into a functional encryption scheme for the functions described by an attribute $\mathsf{att} \in \mathcal{U}$ and a vector $\vec{y}$, that given as input a vector $\vec{x}$ and a predicate $\mathsf{P} \in \mathcal{P}$, outputs $\langle \vec{x}, \vec{y} \rangle$ if $\mathsf{P}(\mathsf{att}) = 1$ and $\perp$ otherwise. For correctness to hold we require that both $\vec{x}$ and $\vec{y}$ are vectors of polynomially-bounded dimension and norm. We consider both the case where the policy P associated with a ciphertext is public, or at the contrary, remains hidden. As explained previously, leveraging state of the art pairing-based ABE, we obtain an FE for $\mathcal{P}$ described by monotone span programs, and an FE for $\mathcal{P}$ for any constant depth formula, where the formula itself remains hidden.

From a technical point of view, our first realization combines the IPFE from [ALS16] with any predicate encoding for prime-order pairing groups. In a nutshell, predicate encodings [Wee14, Att14] are a one-time secure, private key, statistical variant of ABE that are much simpler to construct and to deal with. The resulting construction achieves simulation security, but only in a selective sense, and unfortunately this happens to be the case even if the underlying building blocks achieve adaptive security. Informally, this comes from the fact that our security model explicitly allows the adversary to (get keys to) decrypt the challenge ciphertext. Technically, this means that, throughout the security proof, only functional decryption keys associated with pairs $(\mathsf{att}, \vec{y})$ for which $\mathcal{P}^*(\mathsf{att}) = 0$ can be turned into semi-functional ones (here $\mathcal{P}^*$ denotes the predicate chosen by the adversary for the challenge ciphertext). Following the dual-system encryption methodology, semi-functional keys refer to keys that cannot decrypt successfully the challenge ciphertext, but can decrypt correctly any other honestly generated ciphertext. Keys for which $\mathcal{P}^*(\mathsf{att}) = 1$ cannot be turned semi-functional as otherwise they would fail to (correctly) decrypt the challenge ciphertext. Such a decryption issue does not arise in typical ABE settings, as their security model explicitly prevents the adversary to decrypt the challenge ciphertext.

Our second construction circumvents this difficulty and obtains adaptive security by generalizing the techniques introduced in [OT12], later improved in [CGW18] in the context of fully-hiding predicate encryption for inner product testing. Indeed, in fully-hiding predicate encryption, the proof also has to explicitly deal with the decryption issue sketched above. To do so, we introduce the notion of function encoding, which is the analogue of predicate encoding for functional encryption. Recall that predicate encodings, introduced in [Att14,Wee14], are a "dumbed-down" version of ABE, and provide a framework to extend the dual system encryption methodology introduced by [Wat09] in the context of adaptively-secure IBE to a broad class of ABE, including inner product testing, or Boolean formulas. In our case, we use the abstraction of function encoding to generalize the information-theoretic argument from [CGW18] to capture a broad class of functional encryption, including inner-product FE with access control expressed by inner-product testing, Boolean formulas, and more.

Similarly to predicate encoding, which has received significant interest (particularly as its more general form referred to as Conditional-Disclosure of Secret, e.g. [GIKM00, GKW15, AARV17, LVW17]), we believe the notion of function encoding could be interesting on its own.

In a nutshell, functional encodings enhance a more sophisticated information theoretic argument than traditional Dual System Encryption, where secret keys are switched to a semi-functional mode that still allows them to decrypt the challenge ciphertext, but yield different information than normally generated secret keys. Indeed, in the security proof, the ciphertext will encode the original message $\vec{x}_0$, but also the message $\vec{x}_1$, where the pair $(\vec{x}_0, \vec{x}_1)$ is chosen by the adversary during the indistinguishability game. Normal keys will decrypt with respect to the message $\vec{x}_0$, whereas the semi-functional keys will decrypt with respect to the message $\vec{x}_1$, thereby successfully proving security.

IDENTITY-BASED INNER-PRODUCT FE FROM LATTICES. Our second main result is the construction of two identity-based inner-product FE (IB-IPFE) from the LWE assumption[6]. Both schemes combine existing LWE-based IBE with the LWE-based inner-product FE from [ALS16]. The first one uses the IBE from [GPV08], where the public key described a trapdoor function for which it is hard to sample short preimage. Given the trapdoor— the master key of the IBE— it is possible to efficiently compute a short preimage of any target image. Each identity id yielding a different image, the corresponding preimage, a matrix of short coefficients $\vec{M}_{\mathsf{id}}$, defines the user secret key for id. As it turns out, to produce functional decryption keys associated with identity id and vector $\vec{y}$, we can simply give a projection $\vec{M}_{\mathsf{id}}\vec{y}$. We prove this remarkably simple scheme adaptively-secure in the random oracle model using the security argument of [GPV08] to handle all functional decryption keys that do not decrypt the challenge ciphertext, whereas we use the proof techniques of [ALS16] to take care of all keys that decrypt the challenge ciphertexts.

Our second constructions relies on the IBE from [ABB10], where the public key can be used to derive an identity-based public key $\mathsf{pk}_{\mathsf{id}}$ for any identity id. The public key $\mathsf{pk}_{\mathsf{id}}$ describes a trapdoor function, for which, as in [GPV08], it is hard to compute short preimages. A fixed target image, which belongs to the range of all the trapdoor functions $\mathsf{pk}_{\mathsf{id}}$ is made public. The user secret key for id is a short preimage of the fixed target image, for the function $\mathsf{pk}_{\mathsf{id}}$. Once again, user secret keys happen to be matrices, which can be projected to obtain functional decryption keys $\mathsf{sk}_{\mathsf{id},\vec{y}}$ and get an IB-IPFE.

As a bonus, our schemes inherit the anonymity property of the underlying IBE, that is, the identity associated with a ciphertext remains hidden as long as no functional decryption key that decrypts is issued.

RICHER ACCESS CONTROL FROM LATTICES. The puncturing technique that is used in the security proof of [ABB10] has been generalized to obtain ABE for all circuits in [BGG+14]. However, there are intrinsic technical limitations in our proof strategy which prevent from extending our scheme to the ABE case. In particular, to use the security argument of the IPFE from [ALS16] as part of our own security proof, we rely on a lazy sampling argument: to obtain a functional decryption key $\mathsf{sk}_{\mathsf{id}^\star,\vec{y}}$ where $\mathsf{id}^\star$ is the identity of the challenge ciphertext, we first sample a matrix with short coefficients $\vec{M}_{\mathsf{id}^\star}$ and set the fixed public target image such that this short matrix is a preimage of the target image by the function described by the public key $\mathsf{pk}_{\mathsf{id}^\star}$. Concretely, the target image is a matrix $\vec{T}$, the public key $\mathsf{pk}_{\mathsf{id}} = \vec{A}_{\mathsf{id}^\star}$ is also a matrix, and we want $\vec{A}_{\mathsf{id}^\star}\vec{M}_{\mathsf{id}^\star} = \vec{T}$, where the matrices have matching dimensions. We can first sample $\vec{T}$, then use the trapdoor to compute $\vec{M}_{\mathsf{id}^\star}$ satisfying the previous equation, but we can also first sample a short $\vec{M}_{\mathsf{id}^\star}$, and then set $\vec{T} = \vec{A}_{\mathsf{id}^\star}\vec{M}_{\mathsf{id}^\star}$. This produces identically distributed matrices, and in the latter case, we can produce $\vec{M}_{\mathsf{id}^\star}$ without knowing the trapdoor, which is necessary in the security proof. The matrix $\vec{M}_{\mathsf{id}^\star}$ will actually correspond to the master secret key of the IPFE of [ALS16]. The key $\mathsf{sk}_{\mathsf{id}^\star,\vec{y}}$ is $\vec{M}_{\mathsf{id}^\star}\vec{y}$, as described above, which corresponds to a functional decryption key for $\vec{y}$ in the scheme from [ALS16]. However, this lazy sampling argument is inherently limited to the case where only one attribute (here, identity) satisfies the predicate (here, identity) of the challenge ciphertext. In the case of ABE, there can be multiple such attributes for a given predicate. We leave combining ABE for circuits with inner-product FE as a challenging open problem.

MULTI-INPUT EXTENSIONS. As a final contribution, we show how to generalize our pairing-based IP-FE scheme to the multi input setting. Our realization is rather generic in the sense that it converts any single input construction of the primitive, satisfying few additional properties, into a multi input scheme supporting the same class of functionalities. Specifically, the required properties are that (1) the underlying IP-FE is pairings-based (2) its encryption and key generation algorithms can take as input large norm vectors and (3) its encryption algorithm enjoys linearly homomorphic properties. Recall that, to guarantee efficient decryption, our parings based constructions require that both the plaintext vectors $\vec{x}$ and the function vector $\vec{y}$ have small norm. What we require now is that, if one is willing to give up efficient decryption, the small norm condition can be relaxed (i.e. decryption returns an encoding of the output rather than the output itself).

On a technical level the transformation follows very closely the generic single-input to multi-input IP-FE transform by Abdalla et al. [ACF+18, AGRW17]. In this sense, we believe that the interesting contribution is the primitive itself. Indeed, information leakage is even more problematic in the multi input setting, as here users can combine their inputs with many different ciphertexts coming from other users. In the case of n users this easily leads to an information leakage that completely destroys security. While countermeasures could be put in place to limit the encryption and key queries that the adversary is allowed to ask, by resorting for instance, to the notion of multi-client IPFE, where

[6] We stress that both schemes support exponentially large input domains, as for existing LWE-based inner-product FE schemes.

ciphertexts are associated with time-stamps, and only ciphertext with matching time-stamps can be combined (e.g. [CDG+18]) we believe that our proposed primitive provides a more general and versatile solution to the problem.

Our construction allows users to compute weighted sums on encrypted vectors each associated with a possibly *different* access structure. In our medical example above, this might be used to add even more granularity to the access control of data. That is, some users may obtain keys that can compute statistics on some, but not all, the encrypted data. For instance, doctors in a hospital may be able to compute on a different set of encrypted data then employees of a health insurance company. Moreover, multi-input allows users to aggregate data coming from different sources.

Related Work. We emphasize that the primitive considered in this paper is natural, and as such, it has also been considered in previous works, either implicitly or explicitly.

In [DP19], Dufour-Sans and Pointcheval describe an identity-based functional encryption scheme for inner products as a byproduct of their realization of unbounded IPFE with succinct keys. Their construction is proven selectively secure in the random-oracle model based on the standard decisional bilinear Diffie-Hellman assumption. Compared to their construction, our pairing-based schemes provide support for significantly richer functionalities and are proven secure in the standard model.

In prior works [AJS18,JLMS19], the authors define a so-called partially-hiding FE allowing for the computation on two inputs (x, y), where the input x is seen as a public attribute and the other one, y, remains hidden. The construction of [AJS18] supports degree-2 computation on the private input y, and degree-1 computation on the public input x. Its security rely on the generic bilinear group model. In [JLMS19], functional secret keys support the computation of degree-2 polynomials on the private input, as in [AJS18], but it supports NC_0 computation on the public input. As an additional benefit, the security of their construction rely on a standard assumption on pairing groups (namely, SXDH). In [JLS19], Jain, Lin, and Sahai provided a partially-hiding FE allowing for degree-2 computation on the private input, and NC_1 computation on the public inputs; relying on the SXDH assumption. All of these schemes are in the secret-key setting. Our scheme has the advantage to be public-key, although our techniques inherently rely on the linearity of the inner-product functionality. All of those works focus on simulation, selective security, and use partially-hiding FE in the context of providing indistinguishability obfuscation.

In [CZY19], Chen, Zang, and Yiu propose a construction of attribute-based functional encryption for inner products. Like ours, their construction is pairing-based, but it is less generic, and relies on three decisional assumptions on bilinear groups of composite order $N = p_1p_2p_3$ (p_1, p_2, p_3 distinct primes), which are less efficient than prime-order groups. Our realizations, on the other hand, build generically from any dual system encryption-based ABE. In terms of security, their construction guarantees indistinguishability against adaptive adversaries in the standard model, but only in the weaker setting discussed above, where keys that decrypt cannot be leaked to the adversary, which does not capture the essence of the notion that we achieve, since it does not offer any additional security guarantees with respect to standard ABE schemes. We recall that all our schemes explicitly allow the adversary to get functional keys to decrypt the challenge ciphertext. Also, while our first scheme is only selectively secure, it achieves this in the stronger simulation setting. Finally, no extensions to the multi-input case are considered in [CZY19].

In [Wee17], Wee builds partially hiding predicate encryption schemes which *simultaneously* generalize existing attribute-based and inner-product predicate encryption schemes. Although his constructions support a larger class of policies than our constructions, the decryptor still has access to the payload message (a KEM key in this case) once the access policy is satisfied or to a uniformly random value otherwise. We see it as an interesting open problem to extend his work to also permit selective computations over the payload message when the access policy is satisfied.

Subsequent work. In [AGW20], Abdalla, Gong, and Wee present functional encryption schemes for attribute-weighted sums, where encryption takes as input N attribute-value pairs (x_i, z_i) where x_i is public and z_i is private; secret keys are associated with arithmetic branching programs f, and decryption returns the weighted sum $\sum_{i=1}^{N} f(x_i)z_i$ while leaking no additional information about the z_i's. The functionalities being introduced in this paper can be seen as a particular case of the new functionality in [AGW20] by setting $N = 1$. We note, however, that our adaptively secure construction in Section 3 also allows for private attributes.

In [GJLS20], Gay, Jain, Lin, and Sahai extend the partially-hiding FE scheme in [JLS19] to the public-key setting. Their new functionality also encompasses the ones being introduced in this paper since it supports the computation of degree-2 polynomials on the private input.

Publication note. An abridged version of this paper appears in the proceedings of ASIACRYPT 2020 [ACGU20]. In addition to including complete proofs of security, this version also describes a lattice-based standard-model construction of identity-based functional encryption and a multi-input extension of our schemes. Moreover, it also provides concrete function encodings that correspond to identity-based encryption, inner-product predicate encryption, fully-hiding inner-product predicate encryption and monotone span programs.

Organization. Section 2 recalls some standard notation together with the syntax and security definitions for functional encryption schemes. Section 3 presents our constructions of inner-product FE with fine-grained access control from pairings. Section 4 describes our first lattice-based construction of identity-based functional encryption in the random-oracle model. Section 5 describes the lattice-based standard-model construction of identity-based functional encryption and Section 6 presents a multi-input extension of our schemes.

2 Preliminaries

Notation. We denote with $\lambda \in \mathbb{N}$ a security parameter. A *probabilistic polynomial time* (PPT) algorithm $\mathcal{A}$ is a randomized algorithm for which there exists a polynomial $p(\cdot)$ such that for every input x the running time of $\mathcal{A}(x)$ is bounded by $p(|x|)$. We say that a function $\varepsilon : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every positive polynomial $p(\lambda)$ there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$: $\varepsilon(\lambda) < 1/p(\lambda)$. If S is a set, $x \leftarrow_{\mathrm{R}} S$ denotes the process of selecting x uniformly at random in S. If $\mathcal{A}$ is a probabilistic algorithm, $y \leftarrow_{\mathrm{R}} \mathcal{A}(\cdot)$ denotes the process of running $\mathcal{A}$ on some appropriate input and assigning its output to y. For a positive integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$. We denote vectors $\vec{x} = (x_i)$ and matrices $\mathbf{A} = (a_{i,j})$ in bold. For a set S (resp. vector $\vec{x}$) $|S|$ (resp. $|\vec{x}|$) denotes its cardinality (resp. number of entries). Also, given two vectors $\vec{x}$ and $\vec{x}'$ we denote by $\vec{x} \| \vec{x}'$ their concatenation. By $\equiv$, we denote the equality of statistical distributions, and for any $\varepsilon > 0$, we denote by $\approx_\varepsilon$ the ε-statistical difference of two distributions. For any $x \in \mathbb{R}$, we denote by $\lfloor x \rfloor$ the largest integer less than or equal to x, while for any $z \in [0, 1]$, we denote by $\lfloor z \rceil$ the closest integer to z. For all $\vec{a}_i \in \mathbb{Z}_p^{n_i}$ for $i \in [n]$, we denote by $(\vec{a}_1, \ldots, \vec{a}_n) \in \mathbb{Z}_p^{\sum_{i \in [n]} n_i}$ a column vector, and by $(\vec{a}_1^\top | \cdots | \vec{a}_n^\top) \in \mathbb{Z}_p^{1 \times \sum_{i \in [n]} n_i}$ a row vector.

2.1 Pairing groups

Let PGGen be a PPT algorithm that on input the security parameter 1^λ, returns a description $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e)$ where for all $s \in \{1, 2, T\}$, $\mathbb{G}_s$ is an additive cyclic group of order p for a 2λ-bit prime p. $\mathbb{G}_1$ and $\mathbb{G}_2$ are generated by P_1 and P_2 respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map. Define $P_T := e(P_1, P_2)$, which is a generator of $\mathbb{G}_T$, of order p. We use implicit representation of group elements. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$, define $[a]_s = a \cdot P_s \in \mathbb{G}_s$ as the implicit representation of a in $\mathbb{G}_s$. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n \times m}$ we define $[\mathbf{A}]_s$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_s$:

$$[\mathbf{A}]_s := \begin{pmatrix} a_{11} \cdot P_s & \ldots & a_{1m} \cdot P_s \\ & & \\ a_{n1} \cdot P_s & \ldots & a_{nm} \cdot P_s \end{pmatrix} \in \mathbb{G}_s^{n \times m}.$$

Given $[a]_1$ and $[b]_2$, one can efficiently compute $[a \cdot b]_T$ using the pairing e. For matrices $\mathbf{A}$ and $\mathbf{B}$ of matching dimensions, define $e([\mathbf{A}]_1, [\mathbf{B}]_2) := [\mathbf{AB}]_T$. For any matrix $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_p^{n \times m}$, any group $s \in \{1, 2, T\}$, we denote by $[\mathbf{A}]_s + [\mathbf{B}]_s = [\mathbf{A} + \mathbf{B}]_s$.

For any prime p, we define the following distributions. The DDH distribution over $\mathbb{Z}_p^2$: $a \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, outputs $\vec{a} := \binom{1}{a}$. The DLIN distribution over $\mathbb{Z}_p^{3 \times 2}$: $a, b \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, outputs $\mathbf{A} := \begin{pmatrix} a & 0 \\ 0 & b \\ 1 & 1 \end{pmatrix}$.

Definition 2.1 (DDH assumption). *For any adversary $\mathcal{A}$, any group $s \in \{1, 2, T\}$ and any security parameter λ, let*

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_s, \mathcal{A}}(\lambda) := |\Pr[1 \leftarrow \mathcal{A}(\mathcal{PG}, [\vec{a}]_s, [\vec{a}r]_s)] - \Pr[1 \leftarrow \mathcal{A}(\mathcal{PG}, [\vec{a}]_s, [\vec{u}]_s)]|,$$

where the probabilities are taken over $\mathcal{PG} \leftarrow_{\mathrm{R}} \mathsf{GGen}(1^\lambda, d)$, $\vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\vec{u} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$, and the random coins of $\mathcal{A}$. We say DDH holds in $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_s, \mathcal{A}}(\lambda)$ is a negligible function of λ.

Definition 2.2 (SXDH assumption). *For any security parameter λ and any pairing group $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow_{\mathrm{R}} \mathsf{PGGen}(1^\lambda)$, we say SXDH holds in $\mathcal{PG}$ if DDH holds in $\mathbb{G}_1$ and $\mathbb{G}_2$.*

2.2 Functional Encryption

Definition 2.3 (Functional Encryption [BSW11, O'N10]). *Let $\mathcal{F}$ be a family of functions, with $f \in \mathcal{F}$ defined as $f : \mathcal{X} \to \mathcal{Y}$. A functional encryption scheme for $\mathcal{F}$ consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda, \mathcal{F})$*: takes as input the security parameter λ and a description of the function family $\mathcal{F}$, and outputs a master public key* mpk *and a master secret key* msk*. The master public key* mpk *is assumed to be part of the input of all the remaining algorithms.*
- $\mathsf{Enc}(x \in \mathcal{X})$*: takes as input the master public key* mpk *and a message $x \in \mathcal{X}$, and it outputs a ciphertext* ct*.*
- $\mathsf{KeyGen}(\mathsf{msk}, f \in \mathcal{F})$*: takes as input the master secret key* msk*, a function $f \in \mathcal{F}$, and it outputs a decryption key* sk_f*.*
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$*: takes as input a decryption key* sk_f *along with a ciphertext* ct*, and it outputs a value $y \in \mathcal{Y}$ or the special symbol $\perp$ if it fails.*

A scheme as defined above is correct if for all security parameter λ, $x \in \mathcal{X}$, and $f \in \mathcal{F}$, we have: $\Pr[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_x) = f(x)] = 1]$ *where the probability is taken over* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, $\mathsf{ct}_x \leftarrow \mathsf{Enc}(x)$.

Partial information. For the rest of this paper, it is convenient to split the output of the function in two parts: $(f(x), \mathsf{part}(x))$, where $\mathsf{part}(x)$ is some partial information on x that is independent from f. For instance, we will consider the case of $x := (\mathsf{P}, \vec{x})$, where P is a predicate, and $\vec{x} \in \mathbb{Z}^d$ is a vector of dimension d; each function is described by a pair $(\mathsf{att}, \vec{y})$ where att is an attribute, and $\vec{y} \in \mathbb{Z}^d$. The output $f(x)$ reveals $\vec{x}^\top \vec{y}$ and P if $\mathsf{P}(\mathsf{att}) = 1$; only P otherwise. Note that the information P is always revealed, no matter the function. Considering this part of the input separately will be helpful later.

Security notions. We first recall the selective indistinguishability variant for the security of functional encryption here.

Definition 2.4 (SEL-IND security). *For every functional encryption $\mathcal{FE}$, every security parameter λ, every stateful adversary $\mathcal{A}$, we define the following experiments for $\beta \in \{0, 1\}$:*

Experiment $\mathbf{SEL\text{-}IND}_\beta^{\mathcal{FE}}(1^\lambda, \mathcal{A})$*:*

$(x_0, x_1) \leftarrow \mathcal{A}(1^\lambda, \mathcal{F})$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$
$\mathsf{ct}^\star \leftarrow \mathsf{Enc}(x_\beta)$
$\beta' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{mpk}, \mathsf{ct}^\star)$
Output: β'

where $\mathsf{OKeyGen}(\cdot)$ *is an oracle that on input $f \in \mathcal{F}$, outputs* $\mathsf{KeyGen}(\mathsf{msk}, f)$*. Additionally, if $\mathcal{A}$ ever calls the oracle* KeyGen *on an input $f \in \mathcal{F}$, the challenge queries x_0, x_1 must satisfy: $f(x_0) = f(x_1)$ and $\mathsf{part}(x_0) = \mathsf{part}(x_1)$.*

A functional encryption scheme $\mathcal{FE}$ is SEL-IND-secure if for every PPT adversary $\mathcal{A}$, the following advantage is a negligible function of λ:

$$\mathsf{Adv}^{\mathsf{SEL\text{-}IND}}_{\mathcal{FE},\mathcal{A}}(\lambda) = \left| \Pr\left[\mathbf{SEL\text{-}IND}_0^{\mathcal{FE}}(1^\lambda, \mathcal{A}) = 1\right] - \Pr\left[\mathbf{SEL\text{-}IND}_1^{\mathcal{FE}}(1^\lambda, \mathcal{A}) = 1\right] \right|$$

Now we give the adaptive, indinstinguishability based variant of security for FE. It is the same as the previous definition, except the challenge (x^0, x^1) can be chosen adaptively, after seeing the public key and querying functional decryption keys.

Definition 2.5 (AD-IND security). *For every functional encryption $\mathcal{FE}$, every security parameter λ, every stateful adversary $\mathcal{A}$, we define the following experiments for $\beta \in \{0, 1\}$:*

Experiment $\mathbf{AD\text{-}IND}_{\beta}^{\mathcal{FE}}(1^\lambda, \mathcal{A})$*:*

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$
$(x_0, x_1) \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(1^\lambda, \mathcal{F})$
$\mathsf{ct}^\star \leftarrow \mathsf{Enc}(x_\beta)$
$\beta' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{mpk}, \mathsf{ct}^\star)$
Output: β'

where $\mathsf{OKeyGen}(\cdot)$ *is an oracle that on input* $f \in \mathcal{F}$*, outputs* $\mathsf{KeyGen}(\mathsf{msk}, f)$*. Additionally, if* $\mathcal{A}$ *ever calls the oracle* KeyGen *on an input* $f \in \mathcal{F}$*, the challenge queries* x_0, x_1 *must satisfy:* $f(x_0) = f(x_1)$ *and* $\mathsf{part}(x_0) = \mathsf{part}(x_1)$*.*

A functional encryption scheme $\mathcal{FE}$ *is AD-IND-secure if for every PPT adversary* $\mathcal{A}$*, the following advantage is a negligible function of* λ*:*

$$\mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{FE},\mathcal{A}}(\lambda) = \left| \Pr\left[\mathbf{AD\text{-}IND}_0^{\mathcal{FE}}(1^\lambda, \mathcal{A}) = 1\right] - \Pr\left[\mathbf{AD\text{-}IND}_1^{\mathcal{FE}}(1^\lambda, \mathcal{A}) = 1\right] \right|$$

We now give the simulation-based, selective security. Note that simulation security straightforwardly implies indistinguishable security.

Definition 2.6 (SEL-SIM security). *For any FE scheme* $\mathcal{FE}$ *for functionality* $\mathcal{F}$*, any security parameter* λ*, any PPT stateful adversary* $\mathcal{A}$*, and any PPT simulator* $\mathcal{S} := (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$*, we define the following two experiments.*

$\mathsf{Real}_{\mathcal{A}}^{\mathcal{FE}}(1^\lambda)$*:*
$x^\star \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$
$\mathsf{ct}^\star \leftarrow \mathsf{Enc}(x^\star)$
$\alpha \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{mpk}, \mathsf{ct}^\star)$

$\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathcal{FE}}(1^\lambda)$*:*
$x^\star \leftarrow \mathcal{A}(1^\lambda)$
$(\widetilde{\mathsf{mpk}}, \widetilde{\mathsf{msk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F})$
$\mathsf{ct}^\star \leftarrow \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{msk}}, \mathsf{part}(x^\star))$
$\alpha \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\widetilde{\mathsf{mpk}}, \mathsf{ct}^\star)$

In the real experiment, the key generation oracle OKeyGen*, when given as input* $f \in \mathcal{F}$*, returns* $\mathsf{KeyGen}(\mathsf{msk}, f)$*. In the ideal experiment, the key generation oracle* OKeyGen*, when given as input* $f \in \mathcal{F}$*, computes* $f(x^\star)$*, and returns* $\widetilde{\mathsf{KeyGen}}(\widetilde{\mathsf{msk}}, \mathsf{part}(x^\star), f, f(x^\star))$*, where* $\mathsf{part}(x^\star)$ *denotes the partial information on* $x^\star$*.*

We say an FE scheme is SEL-SIM secure if for all PPT adversaries $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S} := (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$ *such that*

$$\mathsf{Adv}^{\mathsf{SEL\text{-}SIM}}_{\mathcal{FE},\mathcal{A}}(\lambda) := |\Pr[1 \leftarrow \mathsf{Real}_{\mathcal{A}}^{\mathcal{FE}}(1^\lambda)] - \Pr[1 \leftarrow \mathsf{Ideal}_{\mathcal{A},\mathcal{S}}^{\mathcal{FE}}(1^\lambda)]| = \mathsf{negl}(\lambda).$$

3 Inner-Product FE with Fine-grained Access Control

In this section, we present functional encryption schemes for the family of functions that allows users to embed access policies in the encrypted data, and generate functional decryption keys that compute weighted sum on the latter. Namely, each ciphertext is associated with a predicate P, and encrypts a vector $\vec{x} \in [0, B]^d$ for some dimension d and some bound B. Each functional decryption key is associated with an attribute att and a vector $\vec{y} \in [0, B]^d$. Decryption recovers the inner product $\vec{x}^\top \vec{y} \in [0, dB^2]$ together with P if the attribute att satisfies the predicate P. Otherwise, it only recovers the predicate P, but no information about the encrypted vector $\vec{x}$ is revealed.

We show it is possible to combine existing pairing-based ABE together with the inner-product FE from [ALS16]. Our generic construction works on any ABE that relies on the dual system encryption methodology, originally put forth by [Wat09]. Namely, any such ABE that supports the class of predicates $\mathcal{P}$, can be turned into an FE scheme for the family $\mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}} := \mathcal{U} \times [0, B]^d$ of functions described by an attribute $\mathsf{att} \in \mathcal{U}$ and a vector $\vec{y} \in [0, B]^d$, that given as input a predicate $\mathsf{P} \in \mathcal{P}$ where $\mathsf{P} : \mathcal{U} \to \{0, 1\}$ and a vector $\vec{x} \in [0, B]^d$, returns $\vec{x}^\top \vec{y} \in [0, dB^2]$ if $\mathsf{P}(\mathsf{att}) = 1$, 0 otherwise. Note that this can be compactly written as $\mathsf{P}(\mathsf{att}) \cdot \vec{x}^\top \vec{y}$. We will consider the case where the partial information that is leaked about $(\mathsf{P}, \vec{x})$ is P, which corresponds to the case of ABE with public indices, but also the case where the predicate itself is hidden, which corresponding to the case of predicate encryption, also referred

to as ABE with private indices. For correctness, we require the bound B and the dimension d to be polynomially bounded.

We first give a scheme that builds upon any predicate encoding, a one-time secure, private-key, statistical variant of ABE, introduced in [Wee14, Att14], later refined in [Att16, AC16, AC17, ABS17] for prime-order pairing groups. Building a predicate encoding is much easier than directly building an attribute based encryption, since the heavy machinery that is being used to prove security of the resulting ABE is taken care of by these modular frameworks. We follow this line of work by giving a definition of predicate encoding which is essentially that of [CGW15]. For simplicity, we leave the question of using more general predicate encodings, such as those from [AC17], which capture a larger class of ABE, as future work. Our modular construction is general enough to capture identity-based encryption, inner-product predicate encryption, and monotone span programs, whose concrete predicate encodings are given in Appendix B.

3.1 FE with simulation, selective security

First, we recall the definition of predicate encodings.

Definition 3.1 (predicate encoding). *Let $\mathcal{P}$ be a family of predicates and p be a prime. A predicate encoding for $(\mathcal{P}, \mathbb{Z}_p)$ is given by the following polynomial-time deterministic algorithms:*

- $\mathsf{Param}(\mathcal{P})$*: takes as input the family of predicates $\mathcal{P}$, and returns the parameters $(n, |\mathsf{ct}|, |\mathsf{sk}|) \in \mathbb{N}^3$.*
- $\mathsf{EncCt}(\mathsf{P})$*: takes as input a predicate $\mathsf{P} \in \mathcal{P}$, and returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{n \times |\mathsf{ct}|}$.*
- $\mathsf{EncKey}(\mathsf{att})$*: takes as input an attribute $\mathsf{att} \in \mathcal{U}$, and returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{(n+1) \times |\mathsf{sk}|}$.*
- $\mathsf{Decode}(\mathsf{P}, \mathsf{att})$*: takes as input a predicate $\mathsf{P} \in \mathcal{P}$, an attribute $\mathsf{att} \in \mathcal{U}$, and returns a vector $\vec{d} \in \mathbb{Z}_p^{|\mathsf{ct}|+|\mathsf{sk}|}$.*

We require the following properties.

Correctness. *If $\mathsf{P} \in \mathcal{P}$ and $\mathsf{att} \in \mathcal{U}$ such that $\mathsf{P}(\mathsf{att}) = 1$, $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}) \in \mathbb{Z}_p^{n \times |\mathsf{ct}|}$, $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}) \in \mathbb{Z}_p^{(n+1) \times |\mathsf{sk}|}$, $\vec{d} := \mathsf{Decode}(\mathsf{P}, \mathsf{att})$, then $\left(\frac{\mathbf{0}}{\mathbf{C}} \middle| \mathbf{K}\right) \vec{d} = (1, 0, \ldots, 0) \in \mathbb{Z}_p^{n+1}$, where $\mathbf{0} \in \mathbb{Z}_p^{1 \times |\mathsf{ct}|}$.*

Security. *If $\mathsf{P} \in \mathcal{P}$ and $\mathsf{att} \in \mathcal{U}$ such that $\mathsf{P}(\mathsf{att}) = 0$, then the following are identically distributed:*

$$(\alpha | v_1 | \cdots | v_n) \left(\frac{\mathbf{0}}{\mathbf{C}} \middle| \mathbf{K}\right) \text{ and } (0 | v_1 | \cdots | v_n) \left(\frac{\mathbf{0}}{\mathbf{C}} \middle| \mathbf{K}\right),$$

where $\alpha, v_1, \ldots, v_n \leftarrow_{\mathrm{R}} \mathbb{Z}_p$.

Example: Identity-Based Encryption.

- $\mathsf{Param}(\mathsf{IBE})$: takes as input the family of predicates $\mathcal{I}$, where each predicate is described by an identity $\mathsf{id} \in \mathcal{I}$, and returns 1 when given as an input an identity id' such that $\mathsf{id}' = \mathsf{id}$, returns 0 otherwise. It returns the parameters $(n = 2, |\mathsf{ct}| = 1, |\mathsf{sk}| = 1) \in \mathbb{N}^3$.
- $\mathsf{EncCt}(\mathsf{id})$: given $\mathsf{id} \in \mathcal{I}$, returns a matrix $\mathbf{C} = (1, \mathsf{id}) \in \mathbb{Z}_p^{2 \times 1}$ such that $(v_1 | v_2)\mathbf{C} = v_1 + \mathsf{id} v_2 \in \mathbb{Z}_p$.
- $\mathsf{EncKey}(\mathsf{id})$: given $\mathsf{id} \in \mathcal{I}$, returns a matrix $\mathbf{K} = (1, 1, \mathsf{id}) \in \mathbb{Z}_p^{3 \times 1}$ such that $(\alpha | v_1 | v_2)\mathbf{K} = \alpha + v_1 + \mathsf{id} v_2 \in \mathbb{Z}_p$.
- $\mathsf{Decode}(\mathsf{id}, \mathsf{id}')$: if $\mathsf{id} = \mathsf{id}'$, it returns the vector $\vec{d} := \binom{-1}{1} \in \mathbb{Z}_p^2$.

Our simulation, selectively secure FE is described in Fig. 1.

Correctness. Observe that for all predicates $\mathsf{P} \in \mathcal{P}$, the vector $[(\mathbf{W}_1^\top \vec{c}_1 | \ldots | \mathbf{W}_n^\top \vec{c}_1)]_1 \in \mathbb{G}_1^{2 \times n}$ can be computed from mpk and the randomness $s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ used by the encryption algorithm to compute $[\vec{c}_1]_1 := [\vec{a}s]_1$. Then, the encryption algorithm multiplies the resulting vector by the matrix $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}) \in \mathbb{Z}_p^{n \times |\mathsf{ct}|}$ to obtain $[\mathbf{C}_2]_1 \in \mathbb{G}_1^{2 \times |\mathsf{ct}|}$. Similarly, for all attributes $\mathsf{att} \in \mathcal{U}$, the vector $[(\mathbf{U}\vec{y} | \mathbf{W}_1 \vec{k}_1 | \ldots | \mathbf{W}_n \vec{k}_1)]_2 \in \mathbb{G}_2^{2 \times (n+1)}$ can be computed from mpk, msk, and

the randomness $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ used by the key generation algorithm to compute $[\vec{k}_1]_2 := [\vec{b}r]_2$. Then, the key generation algorithm multiplies the resulting vector by the matrix $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}$ to obtain $[\mathbf{K}_2]_1 \in \mathbb{G}_2^{2\times|\mathsf{sk}|}$.

Let $\mathsf{P} \in \mathcal{P}$ and $\mathsf{att} \in \mathcal{U}$ such that $\mathsf{P}(\mathsf{att}) = 1$, $\vec{x}, \vec{y} \in [0, B]^d$, $(\mathsf{P}, [\vec{c}_1]_1, [\mathbf{C}_2]_1, [\vec{c}_3]_1) \leftarrow_{\mathrm{R}} \mathsf{Enc}(\mathsf{mpk}, \mathsf{P}, \vec{x})$, and $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2) \leftarrow_{\mathrm{R}} \mathsf{KeyGen}(\mathsf{msk}, \mathsf{att}, \vec{y})$. The values computed by the decryption algorithm are such that $[\vec{d}_1^\top]_T := [(\vec{c}_1^\top \mathbf{W}_1 \vec{k}_1 | \dots | \vec{c}_1^\top \mathbf{W}_n \vec{k}_1)\mathbf{C}]_T \in \mathbb{G}_T^{1\times|\mathsf{ct}|}$, where $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}) \in \mathbb{Z}_p^{n\times|\mathsf{ct}|}$, and $[\vec{d}_2^\top]_T := [(\vec{c}_1^\top \mathbf{U}\vec{y} | \vec{c}_1^\top \mathbf{W}_1 \vec{k}_1 | \dots | \vec{c}_1^\top \mathbf{W}_n \vec{k}_1)\mathbf{K}]_T \in \mathbb{G}_T^{1\times|\mathsf{sk}|}$, where $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}$. Thus, by correctness of the predicate encoding $(\mathsf{Param}, \mathsf{EncCt}, \mathsf{EncKey}, \mathsf{Decode})$, we have $[\gamma]_T := [\vec{c}_1^\top \mathbf{U}\vec{y}]_T \in \mathbb{G}_T$. To see why, please note that, since $\vec{d}_1^\top = (\vec{c}_1^\top \mathbf{W}_1 \vec{k}_1 | \dots | \vec{c}_1^\top \mathbf{W}_n \vec{k}_1)\mathbf{C} = (\vec{c}_1^\top \mathbf{U}\vec{y} | \vec{c}_1^\top \mathbf{W}_1 \vec{k}_1 | \dots | \vec{c}_1^\top \mathbf{W}_n \vec{k}_1) \left(\frac{\mathbf{0}}{\mathbf{C}}\right)$, $\gamma = (\vec{d}_1^\top | \vec{d}_2^\top)\vec{d} = (\vec{c}_1^\top \mathbf{U}\vec{y} | \vec{c}_1^\top \mathbf{W}_1 \vec{k}_1 | \dots | \vec{c}_1^\top \mathbf{W}_n \vec{k}_1) \left(\frac{\mathbf{0}}{\mathbf{C}} \middle| \mathbf{K}\right) \vec{d} = (\vec{c}_1^\top \mathbf{U}\vec{y} | \vec{c}_1^\top \mathbf{W}_1 \vec{k}_1 | \dots | \vec{c}_1^\top \mathbf{W}_n \vec{k}_1) \cdot (1|0| \dots |0|)^\top = \vec{c}_1^\top \mathbf{U}\vec{y}$. Therefore, $[\mathsf{out}]_T = [\vec{x}^\top \vec{y}]_T$. Finally, assuming the value $B^2 d$ is polynomial in the security parameter, the decryption can efficiently recover the discrete logarithm out from $[\mathsf{out}]_T$.

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}})}$:
$\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \mathsf{PGGen}(1^\lambda)$, $\vec{a}, \vec{b} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times d}$, $(n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow \mathsf{Param}(\mathcal{P})$, for all $i \in [n]$, $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 2}$, $\mathsf{mpk} := \left([\vec{a}]_1, [\vec{b}]_2, [\mathbf{U}^\top \vec{a}]_1, \{[\mathbf{W}_i^\top \vec{a}]_1, [\mathbf{W}_i \vec{b}]_2\}_{i\in[n]}\right)$, $\mathsf{msk} := \mathbf{U}$. Return $(\mathsf{mpk}, \mathsf{msk})$.

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{P}, \vec{x})}$:
$s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{c}_1]_1 := [\vec{a}s]_1$, $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}) \in \mathbb{Z}_p^{n\times|\mathsf{ct}|}$, $[\mathbf{C}_2]_1 := [(\mathbf{W}_1^\top \vec{c}_1 | \dots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1$, $[\vec{c}_3]_1 := [\vec{x} + \mathbf{U}^\top \vec{c}_1]_1$. Return $(\mathsf{P}, [\vec{c}_1]_1, [\mathbf{C}_2]_1, [\vec{c}_3]_1) \in \mathcal{P} \times \mathbb{G}_1^2 \times \mathbb{G}_1^{2\times|\mathsf{ct}|} \times \mathbb{G}_1^d$

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{att}, \vec{y})}$:
$r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{k}_1]_2 := [\vec{b}r]_2$, $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}$, $[\mathbf{K}_2]_2 := [(\mathbf{U}\vec{y} | \mathbf{W}_1 \vec{k}_1 | \dots | \mathbf{W}_n \vec{k}_1)\mathbf{K}]_2$, Return $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2) \in \mathcal{U} \times [0, B]^d \times \mathbb{G}_2 \times \mathbb{G}_2^{2\times|\mathsf{sk}|}$

$\underline{\mathsf{Dec}\Big((\mathsf{P}, [\vec{c}_1]_1, [\mathbf{C}_2]_1, [\vec{c}_3]_1), (\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2)\Big)}$:
$[\vec{d}_1]_T := e([\mathbf{C}_2]_1^\top, [\vec{k}_1]_2) \in \mathbb{G}_T^{|\mathsf{ct}|}$, $[\vec{d}_2^\top]_T := e([\vec{c}_1]_1^\top, [\mathbf{K}_2]_2) \in \mathbb{G}_T^{1\times|\mathsf{sk}|}$, $\vec{d} := \mathsf{Decode}(\mathsf{P}, \mathsf{att})$, $[\gamma]_T := [(\vec{d}_1^\top | \vec{d}_2^\top)\vec{d}]_T \in \mathbb{G}_T$, $[\mathsf{out}]_T := e([\vec{c}_3]_1^\top, [\vec{y}]_2) - [\gamma]_T$. Return out.

Fig. 1. A selectively-secure FE from pairings, for the function family $\mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}}$.

Theorem 3.2 (SEL-SIM security). *If the underlying predicate encoding is secure, then the FE scheme from Fig. 1 is SEL-SIM secure. Namely, for any PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that:*

$$\mathsf{Adv}^{\mathsf{SEL\text{-}IND}}_{\mathcal{FE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1,\mathcal{B}_1}(\lambda) + 2Q \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2,\mathcal{B}_2}(\lambda) + \tfrac{1}{p},$$

where Q denotes the number of queries to $\mathsf{OKeyGen}$.

Proof. The proof goes over a series of hybrid games, defined in Fig. 4. Let $\mathcal{A}$ be a PPT adversary. For any such game G, we denote by $\mathsf{Adv}_{\mathsf{G}}(\mathcal{A})$ the probability $\Pr[1 \leftarrow_{\mathrm{R}} \mathsf{G}(\mathcal{A})]$, that is, the probability that the game outputs 1 when interacting with $\mathcal{A}$. The probability is taken over the random coins of $\mathcal{A}$ and the game G itself. For an overview of the ciphertext and key distributions in the proof, see Figs. 2 and 3.

Game G_0: is the same as $\mathsf{Real}^{\mathcal{FE}}_{\mathcal{A}}(1^\lambda)$ from Definition 2.6.

Game G_1: in this game, the challenge ciphertext is switched to the semi-functional distribution (see Fig. 2). Namely, the vector $[\vec{c}_1]_1$ contained in the challenge ciphertext is switched to uniformly random over $\mathbb{G}_1^2$, using the DDH assumption. The game is described fully in Fig. 4 and is indistinguishable from G_0 by Lemma 3.3.

Ciphertext	$[\vec{c}_1]_1$	$[\mathbf{C}_2]_1$	$[\vec{c}_3]_1$	Hybrid
Normal	$[\vec{a}s]_1, s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$	$[(\mathbf{W}_1^\top \vec{c}_1 \vert \ldots \vert \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1$	$[\vec{x}^\star + \mathbf{U}^\top \vec{c}_1]_1$	G_0
SF	$[\vec{c}_1]_1 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2$	$[(\mathbf{W}_1^\top \vec{c}_1 \vert \ldots \vert \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1$	$[\vec{x}^\star + \mathbf{U}^\top \vec{c}_1]_1$	G_1
Simulated	$\vec{c}_1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \mathsf{span}(\vec{a})$	$[(\mathbf{W}_1^\top \vec{c}_1 \vert \cdots \vert \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1$	$[\mathbf{U}^\top \vec{c}_1]_1$	$\mathsf{Ideal}^{\mathcal{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$

Fig. 2. Overview of ciphertext distributions appearing in the proof of Theorem 3.2, with changes between hybrids highlighted with a gray background. SF stands for semi-functional. Here, $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}^\star)$.

Type of j^{th} Key	Remark	$[\vec{k}_1]_2$	$[\mathbf{K}_2]_2$	Hybrid
Normal	$r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$	$[\vec{b}r]_2$	$[(\mathbf{U}\vec{y} \vert \mathbf{W}_1\vec{k}_1 \vert \cdots \vert \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$	G_0
Pseudo	if $\mathsf{P}^\star(\mathsf{att}) = 0$	$[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2$	$[(\mathbf{U}\vec{y} \vert \mathbf{W}_1\vec{k}_1 \vert \cdots \vert \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$	$\mathsf{H}_{j-1.2}$
Pseudo SF	if $\mathsf{P}^\star(\mathsf{att}) = 0$	$[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2$	$[(\widetilde{\mathbf{U}}\vec{y} \vert \mathbf{W}_1\vec{k}_1 \vert \cdots \vert \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$	$\mathsf{H}_{j-1.7}$
SF	if $\mathsf{P}^\star(\mathsf{att}) = 0$	$[\vec{b}r]_2$	$[(\widetilde{\mathbf{U}}\vec{y} \vert \mathbf{W}_1\vec{k}_1 \vert \cdots \vert \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$	H_{j+1}
Simulated	if $\mathsf{P}^\star(\mathsf{att})$=0	$[\vec{b}r]_2$	$[(\widetilde{\mathbf{U}}\vec{y} \vert \mathbf{W}_1\vec{k}_1 \vert \cdots \vert \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$	$\mathsf{Ideal}^{\mathcal{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$
Simulated	if $\mathsf{P}^\star(\mathsf{att})$=1	$[\vec{b}r]_2$	$[(-\vec{y}^\top\vec{x}^\star \cdot \vec{a}^\perp + \mathbf{U}\vec{y} \vert \mathbf{W}_1\vec{k}_1 \vert \cdots \vert \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$	$\mathsf{Ideal}^{\mathcal{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$

Fig. 3. Overview of key distributions appearing in the proof of Theorem 3.2, with changes between hybrids highlighted with a gray background. SF stands for semi-functional. Throughout the figure, $\mathbf{K} = \mathsf{EncKey}(\mathsf{att})$.

Lemma 3.3. *There exists a* PPT *adversary* $\mathcal{B}_1$*, such that:*

$$|\mathsf{Adv}_{\mathsf{G}_1}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{G}_0}(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1,\mathcal{B}_1}(\lambda).$$

Proof. The PPT adversary $\mathcal{B}_1$ receives the DDH challenge $([\vec{a}]_1, [\vec{z}]_1)$ where $\vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $[\vec{z}]_1 := [\vec{a}s]_1$ with $s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ or $[\vec{z}]_1 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2$, then samples $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 2}$, $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times d}$, $\vec{b} \leftarrow_{\mathrm{R}} \mathsf{DDH}$ and simulates the experiment for $\mathcal{A}$ in the following way:

Simulation of the master public key: Since $\mathcal{B}_1$ samples $\mathbf{U}$ and $\mathbf{W}_i$ himself, he can use the encoding $[\vec{a}]_1$ to compute $[\mathbf{U}^\top\vec{a}]_1$ and $\{[\mathbf{W}_i^\top\vec{a}]_1\}_{i\in[n]}$). Then $\mathcal{B}_1$, computes $\left([\mathbf{W}_i\vec{b}]_2\}_{i\in[n]}\right)$ and outputs $\mathsf{mpk} := \Big([\vec{a}]_1, [\vec{b}]_2, [\mathbf{U}^\top\vec{a}]_1, \{[\mathbf{W}_i^\top\vec{a}]_1, [\mathbf{W}_i\vec{b}]_2\}_{i\in[n]}\Big)$.

Simulation of the encryption challenge: Adversary $\mathcal{B}_1$ sets $[\vec{c}_1]_1 := [\vec{z}]_1$, $\mathbf{C} := \mathsf{EncCt}(\mathsf{P})$, $[\mathbf{C}_2]_1 := [(\mathbf{W}_1^\top\vec{z} \vert \ldots \vert \mathbf{W}_n^\top\vec{z})\mathbf{C}]_1$, $[\vec{c}_3]_1 := [\vec{x}^\star + \mathbf{U}^\top\vec{z}]_1$, and returns $(\mathsf{P}, [\vec{c}_1]_1, [\mathbf{C}_2]_1, [\vec{c}_3]_1)$. When $\mathcal{B}_1$ gets a DDH challenge of the form $[\vec{z}]_1 := [\vec{a}s]_1$ with $s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, it simulates G_1, whereas it simulates G_2 when $[\vec{z}]_1$ is uniformly random over $\mathbb{G}_1$.

Simulation of the functional keys: $\mathcal{B}_1$ generates the keys straightforwardly as described in G_0, using the matrix $\mathbf{U}$, $\{\mathbf{W}_i\}_{i\in[n]}$, and $\vec{b}$.

Game G_2: in this game, all the functional decryption keys associated with an attribute att such that $\mathsf{P}^\star(\mathsf{att}) = 0$ are switched to semi-functional (see Fig. 3). That is, for these keys, the matrix $\widetilde{\mathbf{U}}$ (defined in Fig. 4) is used in place of the master secret key $\mathbf{U}$. Note that the matrix $\widetilde{\mathbf{U}}$, as opposed to the master secret key $\mathbf{U}$, can be computed (information theoretically) from mpk only. These semi-functional keys decrypt successfully normal ciphertexts (which can be produced from mpk), but fail to decrypt semi-functional ciphertexts.

To switch keys from normal to semi-functional, we use a hybrid argument across keys, where each key is first switched to a high entropy distribution, typically referred to as pseudo mode in the dual system methodology [Wat09], where the vector $[\vec{k}_1]_2$ contained in the key is switched to uniformly random over $\mathbb{G}_2^2$, using the DDH assumption. At this point, the proof relies on the security of the predicate encoding to switch the key a semi-functional distribution. After this statistical transition, the vector $[\vec{k}_1]_2$ is switched back to its original distribution, and the proof proceeds to the next key. Details of the transition from game G_1 to game G_2 are given in Lemma 3.4.

G_0, $\boxed{\mathsf{G}_1, \boxed{\mathsf{G}_2}}$:

$(\mathsf{P}^\star, \vec{x}^\star) \leftarrow \mathcal{A}(1^\lambda)$

$\mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda)$, $\vec{a}, \vec{b} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times d}$, $(n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow \mathsf{Param}(\mathcal{P})$, for all $i \in [n]$, $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 2}$, $\mathsf{mpk} := \left([\vec{a}]_1, [\vec{b}]_2, [\mathbf{U}^\top \vec{a}]_1, \{[\mathbf{W}_i^\top \vec{a}]_1, [\mathbf{W}_i \vec{b}]_2\}_{i\in[n]}\right)$

$\vec{u}_0 := \frac{\mathbf{U}^\top \vec{a}}{\|\vec{a}\|_2^2} \in \mathbb{Z}_p^d$, $\widetilde{\mathbf{U}} := \vec{a}\vec{u}_0^\top \in \mathbb{Z}_p^{2\times d}$

$\mathsf{ct}^\star \leftarrow \mathsf{OEnc}(\mathsf{P}^\star, \vec{x}^\star)$

$b \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{mpk}, \mathsf{ct}^\star)$

$\mathsf{OEnc}(\mathsf{P}^\star, \vec{x}^\star)$:

$s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{c}_1]_1 := [\vec{a}s]_1$, $\boxed{[\vec{c}_1]_1 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2}$, $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}^\star)$, $[\mathbf{C}_2]_1 := [(\mathbf{W}_1^\top \vec{c}_1 | \ldots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1$, $[\vec{c}_3]_1 := [\vec{x}^\star + \mathbf{U}^\top \vec{c}_1]_1$. Return $(\mathsf{P}^\star, [\vec{c}_1]_1, [\mathbf{C}_2]_1, [\vec{c}_3]_1)$

$\mathsf{OKeyGen}(\mathsf{att}, \vec{y})$:

$r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{k}_1]_2 := [\vec{b}r]_2$, $\mathbf{K} := \mathsf{EncKey}(\mathsf{att})$, $[\mathbf{K}_2]_2 := [(\mathbf{U}\vec{y} | \mathbf{W}_1 \vec{k}_1 | \cdots | \mathbf{W}_n \vec{k}_1)\mathbf{K}]_2$,

If $\mathsf{P}^\star(\mathsf{att}) = 0$, then $[\mathbf{K}_2]_2 := [(\widetilde{\mathbf{U}}\vec{y} | \mathbf{W}_1 \vec{k}_1 | \cdots | \mathbf{W}_n \vec{k}_1)\mathbf{K}]_2$. Return $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2)$

Fig. 4. Hybrid games for the proof of Theorem 3.2.

Even though the hybrid argument used here is standard in the context of dual system encryption, the crucial difference is that only the keys associated with att such that $\mathsf{P}^\star(\mathsf{att}) = 0$ can be switched to semi-functional. The other keys should actually decrypt the challenge ciphertext properly. This is the reason the experiment needs to know in advance the value $\mathsf{P}^\star$, so as to determine which key can be switched. For the keys that cannot be switched, we use a security argument similar to that used in [ALS16] instead.

Lemma 3.4 (From Game G_1 to game G_2). *There exists a PPT adversary $\mathcal{B}_2$ such that:*

$$|\mathsf{Adv}_{\mathsf{G}_1}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{G}_2}(\mathcal{A})| \leq 2Q \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2, \mathcal{B}_2}(\lambda) + \tfrac{3Q}{p},$$

where Q denotes the number of queries to $\mathsf{OKeyGen}$.

Proof. The proof goes over a series of hybrid games, defined in Fig. 5. Let $\mathcal{A}$ be a PPT adversary. For any such game H, we denote by $\mathsf{Adv}_{\mathsf{H}}(\mathcal{A})$ the probability $\Pr[1 \leftarrow_{\mathrm{R}} \mathsf{H}(\mathcal{A})]$, that is, the probability that the experiment outputs 1 when interacting with $\mathcal{A}$.

Game H_{j-1}: this game is the same as G_1, except that the first $j-1$ queries to $\mathsf{OKeyGen}$ are answered as in G_2. Thus, the game H_0 is the same as G_1 and H_Q is the same as G_2, where Q denotes the number of queries to $\mathsf{OKeyGen}$. We show that for all $j \in [1, Q]$, the games H_{j-1} and H_j are computationally indistinguishable, using the intermediate hybrid games $\mathsf{H}_{j-1.1}$ and $\mathsf{H}_{j-1.2}$, defined in Fig. 5.

Game $\mathsf{H}_{j-1.1}$: this game is the same as H_{j-1}, except that the j-th functional decryption key is switched to a high entropy distribution, where the vector $[\vec{k}_1]_2$ is switched to uniformly random over $\mathbb{G}_2^2$. This is referred to as the pseudo distribution in Fig. 3. Recall that the key is associated to attributes att. The functional key is changed only if $\mathsf{P}^\star(\mathsf{att}) = 0$, where $\mathsf{P}^\star$ is the predicate of the challenge ciphertext. If the j-th functional decryption key is associated with an attribute att such that $\mathsf{P}^\star(\mathsf{att}) = 1$, then we make no changes and move on to $\mathsf{H}_{j.1}$. Let's consider now the case $\mathsf{P}^\star(\mathsf{att}) = 0$. We build an adversary $\mathcal{B}_{j.1}$ such that:

$$|\mathsf{Adv}_{\mathsf{H}_{j-1}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}_{j-1.1}}(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2, \mathcal{B}_{j.1}}(\lambda).$$

Upon receiving the DDH challenge $([\vec{b}]_2, [\vec{z}]_2)$ where $\vec{b} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $[\vec{z}]_2 := [\vec{b}r]_2$ with $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ or $[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathbb{G}_2^2$, $\mathcal{B}_{j.1}$ samples $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 2}$ for all $i \in [n]$, $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times d}$, upon which it can simulate the experiment for $\mathcal{A}$ straightforwardly. When $[\vec{z}]_2 := [\vec{b}r]_2$, $\mathcal{B}_{j.1}$ simulates H_{j-1}, whereas it simulates $\mathsf{H}_{j-1.1}$ when $[\vec{z}]_2$ is uniformly random over $\mathbb{G}_2^2$.

Game $\mathsf{H}_{j-1.2}$: In this game, we change the distribution of the vector $[\vec{k}_1]$ used in the j-th functional decryption key from uniformly random over $\mathbb{G}_2^2$ to uniformly random over $\mathbb{G}_2^2 \setminus \mathsf{span}([\vec{b}]_2)$. Since the size of $\mathsf{span}([\vec{b}]_2)$ is at most p, while the size of $\mathbb{G}_2^2$ is p^2, this change only induces a statistical change of $1/p$. Namely, we obtain:

$$|\mathsf{Adv}_{\mathsf{H}_{j-1.1}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}_{j-1.2}}(\mathcal{A})| \leq \tfrac{1}{p}.$$

Game $\mathsf{H}_{j-1.3}$: Consider first orthogonal vectors $\vec{a}^\perp, \vec{b}^\perp$, such that $\vec{a}^\perp \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \{\mathbf{0}\}, \vec{b}^\perp \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \{\mathbf{0}\}$ with $\vec{a}^\top \vec{a}^\perp = 0$ and $\vec{b}^\top \vec{b}^\perp = 0$. In this game, we change for every $i \in [n]$, the distribution of matrices $\mathbf{W}_i$ to $\mathbf{W}_i + v_i \cdot \vec{a}^\perp (\vec{b}^\perp)^\top$, with $v_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p$. We use the fact that for all $\vec{a}^\perp, \vec{b}^\perp \in \mathbb{Z}_p^2$, the following distributions are identical:

$$\{\mathbf{W}_i\}_{i\in[n]} \text{ and } \{\mathbf{W}_i + \boxed{v_i \cdot \vec{a}^\perp (\vec{b}^\perp)^\top}\}_{i\in[n]},$$

where for all $i \in [n]$, $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 2}$, $v_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p$. This is because each $\mathbf{W}_i$ is uniformly random over $\mathbb{Z}_p^{2\times 2}$. The leftmost distribution corresponds to $\mathsf{H}_{j-1.2}$, whereas the rightmost distribution corresponds to $\mathsf{H}_{j-1.3}$. Namely, in the latter case, we can write the challenge ciphertext as: $(\mathsf{P}^\star, [\vec{c}_1]_1, [\mathbf{C}_2]_1, [\vec{c}_3]_1)$, with $[\vec{c}_1]_1 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2$, $[\vec{c}_3]_1 = [\vec{x}^\star + \mathbf{U}^\top \vec{c}_1]_1$, and:

$$\begin{aligned}[\mathbf{C}_2]_1 &:= [(\mathbf{W}_1^\top \vec{c}_1 + \boxed{v_1 \cdot \vec{b}^\perp (\vec{a}^\perp)^\top \vec{c}_1} | \cdots | \mathbf{W}_n^\top \vec{c}_1 + \boxed{v_n \cdot \vec{b}^\perp (\vec{a}^\perp)^\top \vec{c}_1})\mathbf{C}]_1 \\ &= [(\mathbf{W}_1^\top \vec{c}_1 | \cdots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1 + \boxed{\left[\left(\vec{b}^\perp (\vec{a}^\perp)^\top \vec{c}_1\right) \cdot (v_1 | \cdots | v_n)\mathbf{C}\right]_1}\end{aligned}$$

where $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}^\star)$.

For all keys of index $i \neq j$, the corresponding $\vec{k}_1$ is in the span of $\vec{b}$ (i.e. $\vec{k}_1 \in \mathsf{span}(\vec{b})$), thus, we have that $(\vec{b}^\perp)^\top \vec{k}_1 = 0$, and the fine-dotted factors from the j-th key below vanish in all keys, except for the j-th one. The j-th key is of the following form:

$$\begin{aligned}[\mathbf{K}_2]_2 &:= [(\mathbf{U}\vec{y} | \mathbf{W}_1 \vec{k}_1 + \boxed{v_1 \cdot \vec{a}^\perp (\vec{b}^\perp)^\top \vec{k}_1} | \cdots | \mathbf{W}_n \vec{k}_1 + \boxed{v_n \cdot \vec{a}^\perp (\vec{b}^\perp)^\top \vec{k}_1})\mathbf{K}]_2 \\ &= [(\mathbf{U}\vec{y} | \mathbf{W}_1 \vec{k}_1 | \cdots | \mathbf{W}_n \vec{k}_1)\mathbf{K}]_2 + \boxed{[\left(\vec{a}^\perp (\vec{b}^\perp)^\top \vec{k}_1\right)(\vec{0} | v_1 | \cdots | v_n)\mathbf{K}]_2},\end{aligned}$$

Finally, because of the orthogonality constraints imposed on $\vec{a}^\perp$ and $\vec{b}^\perp$, the extra terms (highlighted in fine-dotted boxes) also do not appear in mpk. We thus have:

$$\mathsf{Adv}_{\mathsf{H}_{j-1.2}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{H}_{j-1.3}}(\mathcal{A}).$$

Game $\mathsf{H}_{j-1.4}$: In this game, the value α used to generate the j-th functional decryption key (see Fig. 5) is switched to uniformly random over $\mathbb{Z}_p$. Using the security of the underlying predicate encoding, we can argue that the following are identically distributed:

$$\Big((v_1 | \cdots | v_n)\mathbf{C}, (\alpha | v_1 | \cdots | v_n)\mathbf{K}\Big) \text{ and } \Big((v_1 | \cdots | v_n)\mathbf{C}, (0 | v_1 | \cdots | v_n)\mathbf{K}\Big).$$

The leftmost value corresponds to $\mathsf{H}_{j-1.3}$, whereas the rightmost value corresponds to $\mathsf{H}_{j-1.4}$. Thus, we have:

$$\mathsf{Adv}_{\mathsf{H}_{j-1.3}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{H}_{j-1.4}}(\mathcal{A}).$$

Game $\mathsf{H}_{j-1.5}$: In this game, the vector $\vec{v}$ used to generate the j-th functional decryption key (see Fig. 5) is switched to $\widetilde{\mathbf{U}}\vec{y}$ instead of $\mathbf{U}\vec{y}$. We show that $\mathsf{H}_{j-1.4}$ and $\mathsf{H}_{j-1.5}$ are identically distributed. Using the basis $(\vec{a}, \vec{a}^\perp)$ of $\mathbb{Z}_p^2$, we can write $\mathbf{U} = \vec{a}\vec{u}_0^\top + \vec{a}^\perp \vec{u}_1^\top$, where $\vec{u}_0 = \frac{\mathbf{U}^\top \vec{a}}{\|\vec{a}\|_2^2}$, and $\vec{u}_1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$.

Thus, we can write the j-th key as $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2)$, with $[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathbb{G}_2^2 \setminus \mathsf{span}([\vec{b}]_2)$, and:

$$\begin{aligned}[\mathbf{K}_2]_2 &:= \Big[(\widetilde{\mathbf{U}}\vec{y} + \vec{a}^{\perp}\vec{u}_1^{\top}\vec{y}\,|\mathbf{W}_1\vec{k}_1|\cdots|\mathbf{W}_n\vec{k}_1)\mathbf{K} + \vec{a}^{\perp}(\vec{b}^{\perp})^{\top}\vec{k}_1(\alpha|v_1|\cdots|v_n)\mathbf{K}\Big]_2 \\ &= \Big[(\widetilde{\mathbf{U}}\vec{y}|\mathbf{W}_1\vec{k}_1|\cdots|\mathbf{W}_n\vec{k}_1)\mathbf{K} \\ &+ \vec{a}^{\perp}(\vec{b}^{\perp})^{\top}\vec{k}_1(0|v_1|\cdots|v_n)\mathbf{K} + \vec{a}^{\perp}\Big(\alpha\cdot(\vec{b}^{\perp})^{\top}\vec{k}_1 + \vec{u}_1^{\top}\vec{y}\,\Big|0\Big|\cdots\Big|0\Big)\mathbf{K}\Big]_2,\end{aligned}$$

where $\mathbf{K} := \mathsf{EncKey}(\mathsf{att})$.

We conclude using the fact that for all vectors $\vec{a}^{\perp}, \vec{b}^{\perp}, \vec{k}_1 \in \mathbb{Z}_p^2$ such that $\vec{k}_1^{\top}\vec{b}^{\perp} \neq 0$, the following are identically distributed:

$$\vec{k}_1^{\top}\vec{b}^{\perp}\cdot\alpha + \vec{u}_1^{\top}\vec{y} \text{ and } \vec{k}_1^{\top}\vec{b}^{\perp}\cdot\alpha,$$

where $\alpha \leftarrow_{\mathrm{R}} \mathbb{Z}_p$. The leftmost distribution corresponds to $\mathsf{H}_{j-1.4}$, whereas the rightmost distribution corresponds to $\mathsf{H}_{j-1.5}$. Thus, we have:

$$\mathsf{Adv}_{\mathsf{H}_{j-1.4}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{H}_{j-1.5}}(\mathcal{A}).$$

Game $\mathsf{H}_{j-1.6}$: In this game, the value α used to generate the j-th functional decryption key (see Fig. 5) is switched back to 0. This transition is the reverse than the transition from $\mathsf{H}_{j-1.3}$ to $\mathsf{H}_{j-1.4}$. Similarly, we use the security of the underlying predicate encoding to argue that the following are identically distributed:

$$\Big((v_1|\cdots|v_n)\mathbf{C}, (\alpha|v_1|\cdots|v_n)\mathbf{K}\Big) \text{ and } \Big((v_1|\cdots|v_n)\mathbf{C}, (0|v_1|\cdots|v_n)\mathbf{K}\Big).$$

The leftmost value corresponds to $\mathsf{H}_{j-1.6}$, whereas the rightmost value corresponds to $\mathsf{H}_{j-1.5}$. Thus, we have:

$$\mathsf{Adv}_{\mathsf{H}_{j-1.5}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{H}_{j-1.6}}(\mathcal{A}).$$

Game $\mathsf{H}_{j-1.7}$: Switching back from $\mathbf{W}_i + v_i\cdot\vec{a}^{\perp}(\vec{b}^{\perp})^{\top}$ to $\mathbf{W}_i$ for all $i \in [n]$, we can write the matrix $[\mathbf{C}_2]_1$ in the challenge ciphertext as:

$$\boxed{[\mathbf{C}_2]_1 = [(\mathbf{W}_1^{\top}\vec{c}_1|\cdots|\mathbf{W}_n^{\top}\vec{c}_1)\mathbf{C}]_1}$$

and the matrix $[\mathbf{K}_2]_2$ in the j-th key as:

$$\boxed{[\mathbf{K}_2]_2 := [(\widetilde{\mathbf{U}}\vec{y}|\mathbf{W}_1\vec{k}_1|\cdots|\mathbf{W}_n\vec{k}_1)\mathbf{K}]_2}$$

Similarly to the transition between $\mathsf{H}_{j-1.2}$ and $\mathsf{H}_{j-1.3}$, it holds that:

$$\mathsf{Adv}_{\mathsf{H}_{j-1.6}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{H}_{j-1.7}}(\mathcal{A}).$$

Game $\mathsf{H}_{j-1.8}$: we switch the distribution of the vector $[\vec{k}_1]_2$ contained in the j-th from uniform over $\mathbb{G}_2^2 \setminus \mathsf{span}([\vec{b}]_2)$ back to uniform over $\mathbb{G}_2^2$. This is the reverse transition than from $\mathsf{H}_{j-1.1}$ to $\mathsf{H}_{j-1.2}$. We have:

$$|\mathsf{Adv}_{\mathsf{H}_{j-1.7}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}_{j-1.8}}(\mathcal{A})| \leq \tfrac{1}{p}.$$

Game H_j: we switch the distribution of the vector $[\vec{k}_1]_2$ contained in the j-th key back to $[\vec{k}_1]_2 := [\vec{b}r]_2$ for $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$. This transition is similar to the transition between game H_{j-1} and $\mathsf{H}_{j-1.1}$. Thus, we obtain a PPT adversary $\mathcal{B}_{j.2}$ such that:

$$|\mathsf{Adv}_{\mathsf{H}_{j-1.8}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}_j}(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2,\mathcal{B}_{j.2}}(\lambda).$$

Summing up for all $j \in [Q]$, we obtain the lemma.

H_j, $\mathsf{H}_{j.1}$, $\mathsf{H}_{j.2}$, $\mathsf{H}_{j.3}$, $\mathsf{H}_{j.4}$, $\mathsf{H}_{j.5}$, $\mathsf{H}_{j.6}$, $\mathsf{H}_{j.7}$, $\mathsf{H}_{j.8}$, $\mathsf{H}_{j.9}$:

$(\mathsf{P}^\star, \vec{x}^\star) \leftarrow_{\mathrm{R}} \mathcal{A}(1^\lambda)$

$\mathcal{PG} \leftarrow_{\mathrm{R}} \mathsf{PGGen}(1^\lambda)$, $\vec{a}, \vec{b} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times d}$, $(n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow_{\mathrm{R}} \mathsf{Param}(\mathcal{P})$, for all $i \in [n]$, $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 2}$

$\vec{a}^\perp, \vec{b}^\perp \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \{\mathbf{0}\}$ s.t. $\vec{a}^\top \vec{a}^\perp = \vec{b}^\top \vec{b}^\perp = 0, \forall i \in [n] : v_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p$

$\mathsf{mpk} := \left([\vec{a}]_1, [\vec{b}]_2, [\mathbf{U}^\top \vec{a}]_1, \{[\mathbf{W}_i^\top \vec{a}]_1, [\mathbf{W}_i \vec{b}]_2\}_{i\in[n]}\right)$

$\vec{u}_0 := \frac{\mathbf{U}^\top \vec{a}}{\|\vec{a}\|_2^2} \in \mathbb{Z}_p^d$, $\widetilde{\mathbf{U}} := \vec{a}\vec{u}_0^\top \in \mathbb{Z}_p^{2\times d}$

$\mathsf{ct}^\star \leftarrow_{\mathrm{R}} \mathsf{OEnc}(\mathsf{P}^\star, \vec{x}^\star)$

$b \leftarrow_{\mathrm{R}} \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{mpk}, \mathsf{ct}^\star)$

$\mathsf{OEnc}(\mathsf{P}^\star, \vec{x}^\star)$:

$[\vec{c}_1]_1 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2$, $\mathbf{C} := \mathsf{EncCt}(\mathsf{P})$, $[\mathbf{C}_2]_1 := [(\mathbf{W}_1^\top \vec{c}_1 | \cdots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1$

$[\mathbf{C}_2]_1 = [(\mathbf{W}_1^\top \vec{c}_1 | \cdots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1 + \left[\left(\vec{b}^\perp (\vec{a}^\perp)^\top \vec{c}_1\right) \cdot (v_1 | \cdots | v_n)\mathbf{C}\right]_1$

$[\mathbf{C}_2]_1 = [(\mathbf{W}_1^\top \vec{c}_1 | \cdots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1$

$[\vec{c}_3]_1 := [\vec{x}^\star + \mathbf{U}^\top \vec{c}_1]_1$.

Return $(\mathsf{P}^\star, [\vec{c}_1]_1, [\mathbf{C}_2]_1, [\vec{c}_3]_1)$

$\mathsf{OKeyGen}(\mathsf{att}, \vec{y})$:

On the the ρ-th query:

$r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{k}_1]_2 := [\vec{b}r]_1$, $\mathbf{K} := \mathsf{EncKey}(\mathsf{att})$.

If $\rho = j+1$ and $\mathsf{P}^\star(\mathsf{att}) = 0$, then $[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathbb{G}_2^2$.

If $\rho = j+1$ and $\mathsf{P}^\star(\mathsf{att}) = 0$, then $[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathbb{G}_2^2 \setminus \mathsf{span}([\vec{b}]_2)$.

If $\rho \le j$ and $\mathsf{P}^\star(\mathsf{att}) = 0$, then: $[\mathbf{K}_2]_2 := [(\widetilde{\mathbf{U}}\vec{y} | \mathbf{W}_1\vec{k}_1 | \cdots | \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$

If $\rho > j$ or $\mathsf{P}^\star(\mathsf{att}) = 1$, then: $[\mathbf{K}_2]_2 := [(\mathbf{U}\vec{y} | \mathbf{W}_1\vec{k}_1 | \cdots | \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$

If $\rho = j+1$ and $\mathsf{P}^\star(\mathsf{att}) = 0$, then:

$\vec{u} = \mathbf{U}\vec{y}$, $\alpha = 0$, $\alpha \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\vec{u} = \widetilde{\mathbf{U}}\vec{y}$,

$[\mathbf{K}_2]_2 := [\vec{u} | \mathbf{W}_1\vec{k}_1 | \cdots | \mathbf{W}_n\vec{k}_1)\mathbf{K} + (\vec{a}^\perp (\vec{b}^\perp)^\top \vec{k}_1)(\alpha | v_1 | \ldots | v_n)\mathbf{K}]_2$

if $\rho = j+1$ and $\mathsf{P}^\star(\mathsf{att}) = 0$, then: $[\mathbf{K}_2]_2 := [(\widetilde{\mathbf{U}}\vec{y} | \mathbf{W}_1\vec{k}_1 | \cdots | \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2$

Return $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2)$

Fig. 5. Hybrid games for the proof of Lemma 3.4, where H_j is defined for all $j \in [Q]$, and $\mathsf{H}_{j.i}$ are defined for all $j \in [Q-1]$, $i \in \{1 \ldots 9\}$. The six type of boxes denote which figure components belong to the hybrid.

Game $\mathsf{Ideal}^{\mathcal{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$**:** we show this game is statistically close to G_2. The simulator $\mathcal{S} := (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$ is described in Fig. 6. First, we use the fact that for all $\vec{a} \in \mathbb{Z}_p^2$, the following distributions are within $1/p$ statistical distance:

$$\vec{c}_1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \text{ and } \vec{c}_1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \mathsf{span}(\vec{a}).$$

The leftmost distribution corresponds to G_2, whereas the rightmost distribution corresponds to $\mathsf{Ideal}^{\mathcal{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$. Then, we use the fact that for all $\vec{x}^\star \in \mathbb{Z}^d$, the following distributions are identical:

$$(\vec{a}, \vec{c}_1, \widetilde{\mathbf{U}}, \mathbf{U}) \text{ and } (\vec{a}, \vec{c}_1, \widetilde{\mathbf{U}}, \mathbf{U} - \vec{a}^\perp (\vec{x}^\star)^\top),$$

where $\vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\vec{c}_1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \mathsf{span}(\vec{a})$, $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times d}$, $\vec{u}_0 := \frac{\mathbf{U}^\top \vec{a}}{\|\vec{a}\|_2^2}$, $\widetilde{\mathbf{U}} := \vec{a}\vec{u}_0^\top$, and $\vec{a}^\perp \in \mathbb{Z}_p^2$ such that $\vec{a}^\top \vec{a}^\perp = 0$ and $\vec{c}_1^\top \vec{a}^\perp = 1$. This is because $\mathbf{U}$ is a uniformly random matrix, so adding an offset $-\vec{a}^\perp (\vec{x}^\star)^\top$ does not change

its distribution. This extra offset doesn't appear in $\widetilde{\mathbf{U}}$ since $\vec{a}^\top \vec{a}^\perp = 0$. The leftmost distribution corresponds to G_2, whereas the rightmost distribution corresponds to $\mathsf{Ideal}^{\mathcal{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$.
Putting everything together, we obtain:

$$|\mathsf{Adv}_{\mathsf{G}_2}(\mathcal{A}) - \Pr[1 \leftarrow_{\mathrm{R}} \mathsf{Ideal}^{\mathcal{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)]| \leq \tfrac{1}{p}.$$

$\underline{\widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}}):}$
$\mathcal{PG} \leftarrow \mathsf{GGen}(1^\lambda)$, $\vec{a}, \vec{b} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\vec{c}_1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \mathsf{span}(\vec{a})$, $\vec{a}^\perp \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ such that $\vec{c}_1^\top \vec{a}^\perp = 1$ and $\vec{a}^\top \vec{a}^\perp = 0$,
$\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times d}$, $\vec{u}_0 := \frac{\mathbf{U}^\top \vec{a}}{\|\vec{a}\|_2^2}$, $\widetilde{\mathbf{U}} := \vec{a}\vec{u}_0^\top$, $(n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow \mathsf{Param}(\mathcal{P})$, for all $i \in [n]$, $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 2}$
Return $\widetilde{\mathsf{pk}} := \left([\vec{a}]_1, [\vec{b}]_2, [\mathbf{U}^\top \vec{a}]_1, \{[\mathbf{W}_i^\top \vec{a}]_1, [\mathbf{W}_i \vec{b}]_2\}_{i\in[n]}\right)$, $\widetilde{\mathsf{msk}} := \left(\widetilde{\mathbf{U}}, \mathbf{U}, \vec{a}^\perp\right)$

$\underline{\widetilde{\mathsf{Enc}}(\widetilde{\mathsf{msk}}, \mathsf{P}^\star):}$
$\mathbf{C} := \mathsf{EncCt}(\mathsf{P})$, $[\mathbf{C}_2]_1 := [(\mathbf{W}_1^\top \vec{c}_1 | \cdots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1$, $[\vec{c}_3]_1 := [\mathbf{U}^\top \vec{c}_1]_1$. Return $(\mathsf{P}^\star, [\vec{c}_1]_1, [\mathbf{C}_2]_1, [\vec{c}_3]_1)$

$\underline{\widetilde{\mathsf{KeyGen}}(\widetilde{\mathsf{msk}}, \mathsf{P}^\star, \vec{y}, \mathsf{att}, \mathsf{P}^\star(\mathsf{att}) \cdot \vec{y}^\top \vec{x}^\star):}$
$r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{k}_1]_2 := [\vec{b}r]_2$, $\mathbf{K} := \mathsf{EncKey}(\mathsf{att})$.
If $\mathsf{P}^\star(\mathsf{att}) = 0$, then $[\mathbf{K}_2]_2 := [(\widetilde{\mathbf{U}}\vec{y} | \mathbf{W}_1 \vec{k}_1 | \cdots | \mathbf{W}_n \vec{k}_1)\mathbf{K}]_2$.
If $\mathsf{P}^\star(\mathsf{att}) = 1$, then $[\mathbf{K}_2]_2 := [(-\vec{y}^\top \vec{x}^\star \cdot \vec{a}^\perp + \mathbf{U}\vec{y} | \mathbf{W}_1 \vec{k}_1 | \cdots | \mathbf{W}_n \vec{k}_1)\mathbf{K}]_2$.
Return $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2)$

Fig. 6. PPT simulator for the security proof of the FE scheme from Fig. 4.

3.2 FE with adaptive, indistinguishability based security

In this section, we build FE schemes for the family of functions $\mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}}$, where $\mathcal{P}$ corresponds to identity-based encryption, inner-product predicate encryption, or even monotone span programs. Similarly to the selective construction in Section 3.1, we give a modular construction that builds upon a simple, information-theoretic, one-time secure object, that generalizes the notion of predicate encoding to functions, hence called function encoding. Namely, a function encoding is a private-key version of functional encryption that only satisfies a one-time security notion.

Recall that our construction from Section 3.1 fails to achieve adaptive security, even if the underlying building blocks are adaptively secure. The reason is that, throughout the security proof, only the functional decryption keys associated with a pair $(\mathsf{att}, \vec{y})$ such that $\mathsf{P}^\star(\mathsf{att}) = 0$ can be turned to semi-functional, where $\mathsf{P}^\star$ is the predicate chosen by the adversary for the challenge ciphertext. In fact, the other keys cannot be turned semi-functional, since they must decrypt correctly the challenge ciphertext, and not just ciphertexts that can be generated from the public key. This challenge does not arise in the typical dual system encryption methodology used for ABE, since none of the queried keys can decrypt.

A similar situation arose in the context of fully-hiding predicate encryption for inner products, where ciphertexts are associated with a vector $\widetilde{\vec{x}} \in \mathbb{Z}_p^n$, functional decryption keys are associated with $\widetilde{\vec{y}} \in \mathbb{Z}_p^n$, and decryption successfully recovers the plaintext if $\widetilde{\vec{x}}^\top \widetilde{\vec{y}} = 0$, whereas no information about that plaintext is revealed otherwise. As opposed to regular inner-product encryption, the vector $\widetilde{\vec{x}}$ is also hidden, the only bit of information that leaks is whether $\widetilde{\vec{x}}^\top \widetilde{\vec{y}} = 0$ or not. In this context, the adversary can query functional decryption keys that decrypt the challenge ciphertext. This is still a meaningful security notion since $\widetilde{\vec{x}}$ remains hidden even when such keys are queried.

We show that the techniques introduced by [OT12], later improved in [CGW18] for adaptively secure fully-hiding predicate encryption for inner products are also relevant to obtain adaptively secure inner-product FE with fine-grained access control (even when the predicate is not hidden). In fact, using function encodings, a new notion we introduce that subsumes the notion of predicate encoding introduced in [Att14, Wee14] in the context of adaptively-secure ABE, we generalize the approach of [OT12, CGW18] to a large class of functional encryption schemes, whereas their scheme corresponds to the special case of inner-product encryption. Namely, we compile any function encoding for the function family $\mathcal{F}$ into an adaptively secure FE for the same class of functions from the SXDH assumption in asymmetric pairings. In Appendix C, we give concrete function encodings that correspond to identity-based encryption, inner-product predicate encryption, fully-hiding inner-product predicate encryption and monotone span programs.

Definition 3.5 (function encoding). *Let $\mathcal{F}$ be a family of functions where each function $f \in \mathcal{F}$ is of the form $f : \mathcal{X} \to \mathbb{Z}_p$, and p be a prime. A function encoding for $(\mathcal{F}, \mathbb{Z}_p)$ is given by the following polynomial-time deterministic algorithms:*

- $\mathsf{Param}(\mathcal{F})$*: takes as input the family of functions $\mathcal{F}$, and returns the parameters $(n, |\mathsf{ct}|, |\mathsf{sk}|) \in \mathbb{N}^3$.*
- $\mathsf{EncCt}(x)$*: takes as input $x \in \mathcal{X}$, and returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{(n+1)\times|\mathsf{ct}|}$.*
- $\mathsf{EncKey}(f)$*: takes as input a function $f \in \mathcal{F}$, and returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}$.*
- $\mathsf{Decode}(f, \mathsf{part}(x))$*: takes as input the partial information $\mathsf{part}(x)$ of $x \in \mathcal{X}$ and $f \in \mathcal{F}$. It returns a vector $\vec{d} \in \mathbb{Z}_p^{|\mathsf{ct}|+|\mathsf{sk}|}$. (See Section 2.2 for a discussion on the partial information).*

We require the following properties.

Correctness. *For all $x \in \mathcal{X}$ and $f \in \mathcal{F}$, $\mathbf{C} := \mathsf{EncCt}(x) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{ct}|}$, $\mathbf{K} := \mathsf{EncKey}(f) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}$, $\vec{d} := \mathsf{Decode}(f, \mathsf{part}(x))$, we have: $(\mathbf{C}|\mathbf{K})\vec{d} = (f(x), 0, \ldots, 0) \in \mathbb{Z}_p^{n+1}$.*

Security. *For any $x^0, x^1 \in \mathcal{X}$ and $f \in \mathcal{F}$ such that $f(x^0) = f(x^1)$ and $\mathsf{part}(x^0) = \mathsf{part}(x^1)$, the following are identically distributed:*

$$\vec{v}^\top(\mathbf{C}|\mathbf{K}) \text{ with } \mathbf{C} := \mathsf{EncCt}(x^0), \mathbf{K} := \mathsf{EncKey}(f)$$
$$\text{and}$$
$$\vec{v}^\top(\mathbf{C}|\mathbf{K}) \text{ with } \mathbf{C} := \mathsf{EncCt}(x^1), \mathbf{K} := \mathsf{EncKey}(f),$$

where $\vec{v} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{n+1}$.

Example: Identity-Based Encryption. Each function is described by an identity $\mathsf{id} \in \mathbb{Z}_p$ and a vector $\vec{y} \in [0, B]^d$, takes as input another identity $\mathsf{id}' \in \mathbb{Z}_p$ and a vector $\vec{x} \in [0, B]^d$, and outputs $\vec{x}^\top \vec{y}$ if $\mathsf{id} = \mathsf{id}'$, 0 otherwise. The partial information $\mathsf{part}(\vec{x}, \mathsf{id}) = \mathsf{id}$.

- Param: returns the parameters $(2d, |\mathsf{ct}| = d, |\mathsf{sk}| = n + 1)$.
- $\mathsf{EncCt}(\vec{x}, \mathsf{id})$: given $\vec{x} \in \mathbb{Z}_p^n$ and $\mathsf{id} \in \mathbb{Z}_p$, returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{(2d+1)\times d}$ such that $\mathbf{C}^\top(w_0, \vec{w}_1, \vec{w}_2) = (w_0\vec{x} + \vec{w}_1 + \mathsf{id}\vec{w}_2) \in \mathbb{Z}_p^d$.
- $\mathsf{EncKey}(\vec{y}, \mathsf{id}')$: given $\vec{y} \in \mathbb{Z}_p^n$ and $\mathsf{id}' \in \mathbb{Z}_p$, returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{(2d+1)\times 1}$ such that $\mathbf{K}^\top(w_0, \vec{w}_1, \vec{w}_2) = \vec{y}^\top(\vec{w}_1 + \mathsf{id}'\vec{w}_2) \in \mathbb{Z}_p$.
- $\mathsf{Decode}(\mathsf{id}, \mathsf{id}', \vec{y})$: if $\vec{x}^\top \vec{y} = 0$, it returns the vector $\vec{d} := (\vec{y}, -1) \in \mathbb{Z}_p^{d+1}$.

Our modular construction is presented in Fig. 7. Proofs of correctness and security are given below.

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}})}$:
$\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \mathsf{PGGen}(1^\lambda)$, $\vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\vec{b} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$, $(n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow \mathsf{Param}(\mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}})$, for all $i \in [0, n]$, $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 3}$, $\mathsf{mpk} := \left([\vec{a}]_1, \{[\mathbf{W}_i^\top \vec{a}]_1\}_{i\in[n]}\right)$, $\mathsf{msk} := \left([\vec{b}]_2, \{[\mathbf{W}_i\vec{b}]_2\}_{i\in[n]}\right)$. Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{P}, \vec{x})}$:
$s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{c}_1]_1 := [\vec{a}s]_1 \in \mathbb{G}_1^2$, $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}, \vec{x}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{ct}|}$, $[\mathbf{C}_2]_1 := [(\mathbf{W}_0^\top \vec{c}_1 | \dots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_1 \in \mathbb{G}_1^{3\times|\mathsf{ct}|}$. Return $(\mathsf{part}(\mathsf{P}, \vec{x}), [\vec{c}_1]_1, [\mathbf{C}_2]_1)$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{att}, \vec{y})}$:
$r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{k}_1]_2 := [\vec{b}r]_2 \in \mathbb{G}_2^3$, $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}$, $[\mathbf{K}_2]_2 := [(\mathbf{W}_0\vec{k}_1 | \dots | \mathbf{W}_n\vec{k}_1)\mathbf{K}]_2 \in \mathbb{G}_2^{2\times|\mathsf{sk}|}$, $[\vec{k}_3]_2 := [\mathbf{W}_0\vec{k}_1]_2 \in \mathbb{G}_2^2$. Return $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2, [\vec{k}_3]_2)$.

$\underline{\mathsf{Dec}\big(\mathsf{part}(\mathsf{P}, \vec{x}), [\vec{c}_1]_1, [\mathbf{C}_2]_1, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2, [\vec{k}_3]_2\big)}$:
$[\vec{d}_1]_T := e([\mathbf{C}_2]_1^\top, [\vec{k}_1]_2) \in \mathbb{G}_T^{|\mathsf{ct}|}$, $[\vec{d}_2^\top]_T := e([\vec{c}_1]_1^\top, [\mathbf{K}_2]_2) \in \mathbb{G}_T^{1\times|\mathsf{sk}|}$, $\vec{d} := \mathsf{Decode}(\mathsf{part}(\mathsf{P}, \vec{x}), \mathsf{att})$, $[\gamma]_T := [(\vec{d}_1, \vec{d}_2)^\top \vec{d}]_T \in \mathbb{G}_T$, Return $\mathsf{out} \in [0, dB^2]$ such that $[\gamma]_T = [\vec{c}_1^\top \vec{k}_3 \cdot \mathsf{out}]_T$. If there isn't such out, return $\perp$.

Fig. 7. An adaptively-secure FE from pairings, for the function family $\mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}}$.

Correctness. Observe that for all predicates $\mathsf{P} \in \mathcal{P}$ and vectors $\vec{x} \in [0, B]^d$, the vector $[(\mathbf{W}_0^\top \vec{c}_1 | \mathbf{W}_1^\top \vec{c}_1 | \dots | \mathbf{W}_n^\top \vec{c}_1)]_1 \in \mathbb{G}_1^{3\times n}$ can be computed from mpk and the randomness $s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ used by the encryption algorithm to compute $[\vec{c}_1]_1 := [\vec{a}s]_1$. Then, the encryption algorithm multiplies by the matrix $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}, \vec{x}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{ct}|}$ to obtain $[\mathbf{C}_2]_1 \in \mathbb{G}_1^{3\times|\mathsf{ct}|}$. Similarly, for all attributes $\mathsf{att} \in \mathcal{U}$, the vector $[(\mathbf{W}_0\vec{k}_1 | \mathbf{W}_1\vec{k}_1 | \dots | \mathbf{W}_n\vec{k}_1)]_2 \in \mathbb{G}_2^{2\times n}$ can be computed from mpk, msk, and the randomness $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ used by the key generation algorithm to compute $[\vec{k}_1]_2 := [\vec{b}r]_2$. Then, the key generation algorithm multiplies by the matrix $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}$ to obtain $[\mathbf{K}_2]_1 \in \mathbb{G}_2^{2\times|\mathsf{sk}|}$.

Let $\mathsf{P} \in \mathcal{P}$ and $\mathsf{att} \in \mathcal{U}$ such that $\mathsf{P}(\mathsf{att}) = 1$, $\vec{x}, \vec{y} \in [0, B]^d$, $(\mathsf{part}(\mathsf{P}, \vec{x}), [\vec{c}_1]_1, [\mathbf{C}_2]_1) \leftarrow_{\mathrm{R}} \mathsf{Enc}(\mathsf{mpk}, \mathsf{P}, \vec{x})$, and $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2, [\vec{k}_3]_2) \leftarrow_{\mathrm{R}} \mathsf{KeyGen}(\mathsf{msk}, \mathsf{att}, \vec{y})$. The values computed by the decryption algorithm are such that $[\vec{d}_1]_T := \left[\mathbf{C}^\top \begin{pmatrix} \vec{c}_1^\top \mathbf{W}_0 \vec{k}_1 \\ \hline \vdots \\ \hline \vec{c}_1^\top \mathbf{W}_n \vec{k}_1 \end{pmatrix}\right]_T$, which implies that $[\vec{d}_1^\top]_T = [(\vec{c}_1^\top \mathbf{W}_0\vec{k}_1 | \vec{c}_1^\top \mathbf{W}_1\vec{k}_1 | \dots | \vec{c}_1^\top \mathbf{W}_n\vec{k}_1)\mathbf{C}]_T \in \mathbb{G}_T^{1\times|\mathsf{ct}|}$, where $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}, \vec{x}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{ct}|}$, and the second equality holds because $\vec{c}_1^\top \mathbf{W}_i \vec{k}_1 \in \mathbb{Z}_p$, for every $i \in \{0 \dots n\}$. Also, $[\vec{d}_2^\top]_T := [(\vec{c}_1^\top \mathbf{W}_0\vec{k}_1 | \vec{c}_1^\top \mathbf{W}_1\vec{k}_1 | \dots | \vec{c}_1^\top \mathbf{W}_n\vec{k}_1)\mathbf{K}]_T \in \mathbb{G}_T^{1\times|\mathsf{sk}|}$, where $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y}) \in \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}$.

Thus, by correctness of the function encoding $(\mathsf{Param}, \mathsf{EncCt}, \mathsf{EncKey}, \mathsf{Decode})$, we have $[\gamma]_T := [\vec{c}_1^\top \mathbf{W}_0 \vec{k}_1 \cdot \vec{x}^\top \vec{y}]_T = [\vec{c}_1^\top \vec{k}_3 \cdot \vec{x}^\top \vec{y}] \in \mathbb{G}_T$. Therefore, assuming the value $B^2 d$ is polynomial in the security parameter, the decryption can efficiently recover $\mathsf{out} = \vec{x}^\top \vec{y} \in [0, B^2 d]$.

$\underline{\mathsf{G}_0, \boxed{\mathsf{G}_1, \boxed{\mathsf{G}_2}}:}$

$\beta \leftarrow_{\mathrm{R}} \{0,1\}, \mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda), \vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}, \vec{b} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3, (n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow \mathsf{Param}(\mathcal{F}_{\mathsf{ipfe}(d,B)}, \mathcal{P})$, for all $i \in [0,n]$, $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 3}$, $\mathsf{mpk} := \big([\vec{a}]_1, \{[\mathbf{W}_i^\top \vec{a}]_1, \}_{i\in[n]}\big)$

$\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big) \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(1^\lambda, \mathsf{mpk})$

$\mathsf{ct}^\star \leftarrow_{\mathrm{R}} \mathsf{OEnc}\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big)$

$\beta' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{mpk}, \mathsf{ct}^\star)$

Return 1 if $\beta' = \beta$, 0 otherwise.

$\underline{\mathsf{OEnc}\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big):}$

$s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{c}_1]_1 := [\vec{a}s]_1$, $\boxed{[\vec{c}_1]_1 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2}$ $\mathbf{C} := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta)$, $\boxed{\mathbf{C} := \mathsf{EncCt}(\mathsf{P}^0, \vec{x}^0)}$, $[\mathbf{C}_2]_1 := [(\mathbf{W}_0^\top \vec{c}_1 | \mathbf{W}_1^\top \vec{c}_1 | \ldots | \mathbf{W}_n^\top \vec{c}_1)\mathbf{C}]_T$, Return $\mathsf{ct}^\star := (\mathsf{part}(\mathsf{P}^\beta, \vec{x}^\beta), [\vec{c}_1]_1, [\mathbf{C}_2]_1)$

$\underline{\mathsf{OKeyGen}(\mathsf{att}, \vec{y}):}$

$r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $[\vec{k}_1]_2 := [\vec{b}r]_1$, $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y})$, $[\mathbf{K}_2]_2 := [(\mathbf{W}_0 \vec{k}_1 | \mathbf{W}_1 \vec{k}_1 | \cdots | \mathbf{W} \vec{k}_1)\mathbf{K}]_2$, $[\vec{k}_3]_2 := [\mathbf{W}_0 \vec{k}_1]_2$. Return $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\mathbf{K}_2]_2, [\vec{k}_3]_2)$

Fig. 8. Hybrid games for the proof of Theorem 3.6.

Theorem 3.6 (AD-IND security). *If the underlying function encoding is secure, then the FE scheme from Fig. 7 is AD-IND secure. Namely, for any PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that:*

$$\mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{FE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1,\mathcal{B}_1}(\lambda) + 4Q\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2,\mathcal{B}_2}(\lambda),$$

where Q denotes the number of queries to $\mathsf{OKeyGen}$.

Proof. The proof uses a series of hybrid games, described in Fig. 8. For each game G, we define by $\mathsf{Adv}_{\mathsf{G}}(\mathcal{A})$ the advantage of $\mathcal{A}$ in G, that is: $2 \cdot |\Pr[1 \leftarrow_{\mathrm{R}} \mathsf{G}(\mathcal{A})] - 1/2|$.

Game G_0: is defined such that $\mathsf{Adv}_{\mathsf{G}_0}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{FE},\mathcal{A}}(\lambda)$.

Game G_1: here we change the distribution of the vector $[\vec{c}_1]_1$ that is part of the challenge ciphertext to uniformly random over $\mathbb{G}_1^2$, using the DDH assumption in $\mathbb{G}_1$. Namely, we build a PPT adversary $\mathcal{B}_1$ such that:

$$|\mathsf{Adv}_{\mathsf{G}_0}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{G}_1}(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1,\mathcal{B}_1}(\lambda).$$

Upon receiving a challenge $(\mathcal{PG}, [\vec{a}]_1, [\vec{z}]_1)$, where $[\vec{z}]_1 := [\vec{a}s]_1$ for $s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, or $[\vec{z}]_1 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2$, the adversary $\mathcal{B}_1$ samples $(n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow \mathsf{Param}(\mathcal{F}_{\mathsf{ipfe}(d,B)}, \mathcal{P})$, for all $i \in [0,n]$, $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 3}$, and simulate $\mathcal{A}$'s view in a straightforward way, setting $[\vec{c}_1]_1 := [\vec{z}]_1$ in the challenge ciphertext.

Game G_2: here we change the distribution of the challenge ciphertext so that it doesn't depend on the random bit $\beta \leftarrow_{\mathrm{R}} \{0,1\}$ anymore. Clearly,

$$\mathsf{Adv}_{\mathsf{G}_2}(\mathcal{A}) = 0.$$

We show that G_1 and G_2 are computationally indistinguishable using the security of a private-key variant of our scheme. Namely, we exhibit a PPT adversary $\mathcal{B}_2$ such that:

$$|\mathbf{Adv}_{\mathsf{G}_1}(\mathcal{A}) - \mathbf{Adv}_{\mathsf{G}_2}(\mathcal{A})| \leq \mathsf{Adv}_{\mathsf{H}_0}(\mathcal{B}_2),$$

where $\mathsf{Adv}_{\mathsf{H}_0}(\mathcal{B}_2)$ denotes the advantage of $\mathcal{B}_2$ in game H_0, which is the private-key analogue of game G_0 (see Fig. 10). We use the fact that for any $i \in [0,n]$: $(\mathbf{W}_i^\top \vec{a}, \mathbf{W}_i^\top \vec{c}_1)$ with $\mathbf{W}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2\times 3}$, $\vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\vec{c}_1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$, is within negligible statistical distance from $(\mathbf{W}_i^\top \vec{a}, \vec{w}_i)$ with $\vec{w}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$. Roughly speaking, the vectors $\vec{w}_i$ can be used as a fresh private-key, independent of the public key $\{[\mathbf{W}_i^\top \vec{a}]_1\}$. Note that when $\vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$ and $\vec{a}^\perp \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \{\mathbf{0}\}$ such that $\vec{a}^\top \vec{a}^\perp = 0$, we have that the vectors $(\vec{a}|\vec{a}^\perp)$ form a basis of $\mathbb{Z}_p^2$. Thus we can write $\mathbf{W}_i^\top := \widetilde{\vec{w}_i}\vec{a}^\top + \vec{w}_i(\vec{a}^\perp)^\top$, where $\widetilde{\vec{w}_i}, \vec{w}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$, and $\vec{a}^\perp \in \mathbb{Z}_p^2$ is such that $\vec{a}^\top \vec{a}^\perp = 0$ and $\vec{c}_1^\top \vec{a}^\perp = 1$. This way, the public key can be written as:

$$\mathsf{mpk} := \left([\vec{a}]_1, \{[\widetilde{\vec{w}_i}\vec{a}^\top \vec{a}]_1\}_{i\in[n]}\right),$$

the challenge ciphertext can be written as:

$$\begin{aligned}(\mathsf{part}(\mathsf{P}^\beta, \vec{x}^\beta), [\vec{c}_1]_1, [\mathbf{C}_2]_1), \text{ with } & [\vec{c}_1]_1 \leftarrow_{\mathrm{R}} \mathbb{G}_1^2,\\ & \mathbf{C} := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta),\\ & [\mathbf{C}_2]_1 := [(\vec{w}_0^\top|\vec{w}_1^\top|\ldots|\vec{w}_n^\top)\mathbf{C}]_1,\end{aligned}$$

which corresponds exactly to game H_0. The functional decryption keys can be written as:

$$\begin{aligned}& r \leftarrow_{\mathrm{R}} \mathbb{Z}_p, [\vec{k}_1]_2 := [\vec{b}r]_1, \mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y}),\\ & [\mathbf{K}_2]_2 := [(\vec{a}\widetilde{\vec{w}_0}^\top + \vec{a}^\perp\vec{w}_0^\top)\vec{k}_1|\cdots|(\vec{a}\widetilde{\vec{w}_n}^\top + \vec{a}^\perp\vec{w}_n^\top)\vec{k}_1)\mathbf{K}]_2,\\ & [\vec{k}_3]_2 := [(\vec{a}\widetilde{\vec{w}_0}^\top + \vec{a}^\perp\vec{w}_0^\top)\vec{k}_1]_2.\end{aligned}$$

The adversary $\mathcal{B}_2$ samples $\widetilde{\vec{w}_i} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$ for all $i \in [0,n]$ and $\vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\vec{a}^\perp \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ such that $\vec{a}^\top \vec{a}^\perp = 0$, thanks to which it can simulate the public key to $\mathcal{A}$. To generate the challenge ciphertext, $\mathcal{B}_2$ forwards the query $\left((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\right)$ to its own encryption oracle, and forwards its challenge ciphertext to $\mathcal{A}$. When $\mathcal{A}$ queries $\mathsf{OKeyGen}(\mathsf{att}, \vec{y})$, $\mathcal{B}_2$ queries its own oracle to get $\mathsf{sk}_{\mathsf{att},\vec{y}} := (\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\vec{k}_2]_2, [k_3]_2)$, where $[\vec{k}_2^\top]_2 := [(\vec{w}_0^\top\vec{k}_1|\ldots|\vec{w}_n^\top\vec{k}_1)\mathbf{K}]_2$ for $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y})$, and $[k_3]_2 := [\vec{w}_0^\top\vec{k}_1]_2$. $\mathcal{B}_2$ computes $[\mathbf{K}'_2]_2 := [\vec{a}^\perp\vec{k}_2^\top]_2 + [\vec{a}(\widetilde{\vec{w}_0}^\top|\ldots|\widetilde{\vec{w}_n}^\top)\mathbf{K}]_2$, and $[\vec{k}'_3]_2 := [\vec{a}^\perp k_3]_2 + [\vec{a}\widetilde{\vec{w}_0}^\top\vec{k}_1]_2$, and returns $([\vec{k}_1]_2, [\mathbf{K}'_2]_2, [\vec{k}'_3]_2)$ to $\mathcal{A}$.
In Lemma 3.7, we show that $\mathsf{Adv}_{\mathsf{H}_0}(\mathcal{B}_2)$ is negligible.

Lemma 3.7. *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that:*

$$\mathsf{Adv}_{\mathsf{H}_0}(\mathcal{A}) \leq 4Q \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2,\mathcal{B}}(\lambda),$$

where Q denotes the number of queries to $\mathsf{OKeyGen}$.

Proof. The proof uses a series of hybrid games, described in Figs. 9 and 10. For any game H, we denote by $\mathsf{Adv}_{\mathsf{H}}(\mathcal{A})$ the advantage of $\mathcal{A}$ in H, that is, the probability $\Pr[1 \leftarrow_{\mathrm{R}} \mathsf{H}(\mathcal{A})]$ that the game H returns 1 when interacting with $\mathcal{A}$, where the probability is taken over the random coins of $\mathcal{A}$ and the game H itself.

Game H_0: is described in Fig. 10. We use the basis $(\vec{b}^*|\vec{b}_2^*|\vec{b}_3^*)$ of $\mathbb{Z}_p^3$ to write $\vec{w}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$ as $\vec{w}_i := w_i^1\vec{b}^* + w_i^2\vec{b}_1^* + w_i^3\vec{b}_3^*$, with $w_i^1, w_i^2, w_i^3 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, for all $i \in [0,n]$. This basis is dual to the random basis $(\vec{b}|\vec{b}_2|\vec{b}_3)$, that is: $\vec{b}_2^\top\vec{b}^* = \vec{b}_3^\top\vec{b}^* = 0$, $\vec{b}^\top\vec{b}^* = 1$, $\vec{b}^\top\vec{b}_2^* = \vec{b}_3^\top\vec{b}_2^* = 0$, $\vec{b}_2^\top\vec{b}_2^* = 1$, $\vec{b}^\top\vec{b}_3^* = \vec{b}_2^\top\vec{b}_3^* = 0$, $\vec{b}_3^\top\vec{b}_3^* = 1$.

Game H_1: we change the component of the ciphertext along the vector $\vec{b}_3^*$, using the one-time security of the function encoding. Namely, we use the fact that $(w_0^3|\ldots|w_n^3)\mathbf{C}^\beta$ is identically distributed to $(w_0^3|\ldots|w_n^3)\mathbf{C}^0$, where $w_0^3, \ldots, w_n^3 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\mathbf{C}^\beta := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta)$, and $\mathbf{C}^0 := \mathsf{EncCt}(\mathsf{P}^0, \vec{x}^0)$, by security of the function encoding (see Definition 3.5). Note that we rely on the fact that the values $(w_0^3|\ldots|w_n^3)$ do not appear in the functional decryption keys, since all of these keys contain vectors $[\vec{k}_1]_2$ such that $\vec{k}_1^\top\vec{b}_3^* = 0$.

Game $\mathsf{H}_{1.q}$: is defined in Fig. 9 for all $q \in [Q]$ where Q denotes the number of queries to $\mathsf{OKeyGen}$. Note that $\mathsf{H}_{1.0}$ is the same as game H_1. To show that $\mathsf{H}_{1.q-1}$ is computationally indistinguishable from $\mathsf{H}_{1.q}$ for all $q \in [Q]$, we use intermediate hybrid games $\mathsf{H}_{1.q-1.1}$,$\mathsf{H}_{1.q-1.2}$, and $\mathsf{H}_{1.q-1.3}$, also defined in Fig. 9.

Game $\mathsf{H}_{1.q-1.1}$**:** in this game, we switch the distribution of the output to the q-th query to $\mathsf{OKeyGen}$. Namely, the vector $[\vec{k}_1]_2$ is sampled uniformly in $\mathsf{span}([\vec{b}_2]_2)$ instead of $\mathsf{span}([\vec{b}]_2)$. This is done using the DDH assumption in $\mathbb{G}_2$. The reduction from DDH must not know the vectors $\vec{b}^*$ and $\vec{b}_2^*$, which are trapdoors for the DDH assumption (they permit to test membership in $\mathsf{span}([\vec{b}]_2)$ or $\mathsf{span}([\vec{b}_2]_2)$ respectively). Indeed, it is possible to sample the vectors $w_i^1\vec{b}^* + w_i^2\vec{b}_2^*$ with $w_i^1, w_i^2 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ for all $i \in [0,n]$, without knowing explicitly the vectors $\vec{b}^*$ and $\vec{b}_2^*$. We consider an intermediate game $\mathsf{H}'_{1.q-1.1}$ which is like $\mathsf{H}_{1.q-1.1}$ except that the vector $[\vec{k}_1]_2$ from the q-th key is sampled uniformly over $\mathsf{span}([\vec{b}]_2, [\vec{b}_2]_2)$. We build PPT adversaries $\mathcal{B}'_{1.q-1}$ and $\mathcal{B}'_{1.q-1.1}$ such that: $|\mathsf{Adv}_{\mathsf{H}_{1.q-1}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}'_{1.q-1.1}}(\mathcal{A})| \leq \mathsf{Adv}^{\mathrm{As-1}}_{\mathcal{PG},\mathcal{B}'_{1.q-1}}(\lambda)$, and $|\mathsf{Adv}_{\mathsf{H}'_{1.q-1.1}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}_{1.q-1.1}}(\mathcal{A})| \leq \mathsf{Adv}^{\mathrm{As-2}}_{\mathcal{PG},\mathcal{B}'_{1.q-1.1}}(\lambda)$. By Lemma 3.8, this implies the existence of a PPT adversary $\mathcal{B}_{1.q-1}$ such that

$$|\mathsf{Adv}_{\mathsf{H}_{1.q-1}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}_{1.q-1.1}}(\mathcal{A})| \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2,\mathcal{B}_{1.q-1}}(\lambda).$$

Upon receiving $(\vec{u}_1, \vec{u}_2, \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, [\vec{z}]_2)$, $\mathcal{B}'_{1.q-1}$ computes the challenge ciphertext as $\vec{c}_2^{\beta} + \vec{c}_2^{0}$ with $\vec{c}_2^{\beta\top} := (\vec{u}_1^\top w_0^1 + \vec{u}_2^\top w_0^2 | \dots | \vec{u}_1^\top w_n^1 + \vec{u}_2^\top w_n^2)\mathbf{C}^\beta$ and $\vec{c}_2^{0\top} := (\vec{b}_3^{*\top} w_0^3 | \dots | \vec{b}_3^{*\top} w_n^3)\mathbf{C}^0$ with $w_i^1, w_i^2, w_i^3 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ for all $i \in [0,n]$, $\mathbf{C}^\beta := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta)$ and $\mathbf{C}^0 := \mathsf{EncCt}(\mathsf{P}^0, \vec{x}^0)$. It computes the keys as follows. For the $q-1$ first keys, it samples $[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}_3]_2)$; for the q-th key, it uses $[\vec{k}_1]_2 := [\vec{z}]_2$, and for the last $Q-q-1$ keys it computes $[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}]_2)$. The rest of the keys are computed as $[\vec{k}_2^\top]_2 := [(\vec{w}_0^\top \vec{k}_1 | \vec{w}_1^\top \vec{k}_1 | \cdots | \vec{w}_n^\top \vec{k}_1)\mathbf{K}]_T$, $[k_3]_2 := [\vec{w}_0^\top \vec{k}_1]_2$, with $\vec{w}_i := \vec{u}_1 w_i^1 + \vec{u}_2 w_i^2 + \vec{b}_3^* w_i^3$ for all $i \in [0,n]$. The adversary $\mathcal{B}'_{1.q-1.1}$ works similarly.

Game $\mathsf{H}_{1.q-1.2}$**:** Here we switch the distribution of the challenge ciphertext, using the fact that the following distributions are identical:

$$(w_0^2 | \dots | w_0^2)^\top \mathbf{C}^\beta, (w_0^2 | \dots | w_0^2)^\top \mathbf{K} \text{ and } (w_0^2 | \dots | w_0^2)^\top \mathbf{C}^0, (w_0^2 | \dots | w_0^2)^\top \mathbf{K},$$

where $w_0^2, \dots, w_n^2 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\mathbf{C}^\beta := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta)$, $\mathbf{C}^0 := \mathsf{EncCt}(\mathsf{P}^0, \vec{x}^0)$, and $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y})$. This holds by security of the function encoding (see Definition 3.5), since by definition of the security game, we have $\mathsf{P}^0(\mathsf{att}) \cdot \vec{y}^\top \vec{x}^0 = \mathsf{P}^1(\mathsf{att}) \cdot \vec{y}^\top \vec{x}^1$ for all queries $(\mathsf{att}, \vec{y})$ to $\mathsf{OKeyGen}$. The leftmost distribution corresponds to game $\mathsf{H}_{1.q-1.1}$, whereas the rightmost distribution corresponds to the game $\mathsf{H}_{1.q-1.2}$. Thus, we have:

$$\mathsf{Adv}_{\mathsf{H}_{1.q-1.1}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{H}_{1.q-1.2}}(\mathcal{A}).$$

Game $\mathsf{H}_{1.q-1.3}$**:** in this game, we switch the distribution of the output to the q-th query to $\mathsf{OKeyGen}$. Namely, the vector $[\vec{k}_1]_2$ is sampled uniformly in $\mathsf{span}([\vec{b}_3]_2)$ instead of $\mathsf{span}([\vec{b}_2]_2)$. This is done using the DDH assumption in $\mathbb{G}_2$. As for the transition between games $\mathsf{H}_{1.q-1.1}$ and $\mathsf{H}_{1.q-1.2}$, we consider an intermediate game $\mathsf{H}'_{1.q-1.2}$ which is like $\mathsf{H}_{1.q-1.2}$ except the vector $[\vec{k}_1]_2$ from the q-th key is sampled uniformly over $\mathsf{span}([\vec{b}_2]_2, [\vec{b}_3]_2)$. We build PPT adversaries $\mathcal{B}'_{1.q-1.2}$ and $\mathcal{B}'_{1.q-1.3}$ such that: $|\mathsf{Adv}_{\mathsf{H}_{1.q-1.2}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}'_{1.q-1.2}}(\mathcal{A})| \leq \mathsf{Adv}^{\mathrm{As-3}}_{\mathcal{PG},\mathcal{B}'_{1.q-1.2}}(\lambda)$, and $|\mathsf{Adv}_{\mathsf{H}'_{1.q-1.2}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}_{1.q-1.3}}(\mathcal{A})| \leq \mathsf{Adv}^{\mathrm{As-4}}_{\mathcal{PG},\mathcal{B}'_{1.q-1.3}}(\lambda)$. By Lemma 3.8, this implies the existence of a PPT adversary $\mathcal{B}_{1.q-1.2}$ such that

$$|\mathsf{Adv}_{\mathsf{H}_{1.q-1.2}}(\mathcal{A}) - \mathsf{Adv}_{\mathsf{H}_{1.q-1.3}}(\mathcal{A})| \leq 2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2,\mathcal{B}_{1.q-1.2}}(\lambda).$$

Upon receiving $(\vec{u}_1, \vec{u}_2, \vec{b}^*, [\vec{b}]_2, [\vec{b}_3]_2, [\vec{z}]_2)$, $\mathcal{B}'_{1.q-1}$ computes the challenge ciphertext as $\vec{c}_2^{\beta} + \vec{c}_2^{0}$ with $\vec{c}_2^{\beta\top} := (\vec{b}^{*\top} w_0^1 | \dots | \vec{b}^{*\top} w_n^1)\mathbf{C}^\beta$ and $\vec{c}_2^{0\top} := (\vec{u}_1^\top w_0^2 + \vec{u}_2^\top w_0^3 | \dots | \vec{u}_1^\top w_n^2 + \vec{u}_2^\top w_n^3)\mathbf{C}^0$ with $w_i^1, w_i^2, w_i^3 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ for all $i \in [0,n]$. It computes the keys as follows. For the $q-1$ first keys, it samples $[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}_3]_2)$; for the q-th key, it uses $[\vec{k}_1]_2 := [\vec{z}]_2$, and for the last $Q-q-1$ keys it computes $[\vec{k}_1]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}]_2)$. The rest of the keys are computed as $[\vec{k}_2^\top]_2 := [(\vec{w}_0^\top \vec{k}_1 | \vec{w}_1^\top \vec{k}_1 | \cdots | \vec{w}_n^\top \vec{k}_1)\mathbf{K}]_T$, $[k_3]_2 := [\vec{w}_0^\top \vec{k}_1]_2$, with $\vec{w}_i := \vec{b}^* w_i^1 + \vec{u}_1 w_i^2 + \vec{u}_2 w_i^3$ for all $i \in [0,n]$. The adversary $\mathcal{B}'_{1.q-1.3}$ works similarly.

Game $\mathsf{H}_{1.q}$**:** the transition between games $\mathsf{H}_{1.q-1.3}$ and $\mathsf{H}_{1.q}$ is similar to the transition between games H_0 and H_1. Namely,we switch the distribution of the challenge ciphertext, using the fact that the following distributions are identical:

$$(w_0^2 | \dots | w_0^2)^\top \mathbf{C}^\beta \text{ and } (w_0^2 | \dots | w_0^2)^\top \mathbf{C}^0,$$

where $w_0^2, \ldots, w_n^2 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\mathbf{C}^\beta := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta)$, and $\mathbf{C}^0 := \mathsf{EncCt}(\mathsf{P}^0, \vec{x}^0)$. The leftmost distribution corresponds to game $\mathsf{H}_{1.q}$, whereas the rightmost distribution corresponds to the game $\mathsf{H}_{1.q-1.3}$. Thus, we have:

$$\mathsf{Adv}_{\mathsf{H}_{1.q-1.3}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{H}_{1.q}}(\mathcal{A}).$$

Game H_2: the transition between games $\mathsf{H}_{1.Q}$ and H_2 is also similar to the transition between games H_0 and H_1. Namely, we switch the distribution of the challenge ciphertext, using the fact that the following distributions are identical:

$$(w_0^1|\ldots|w_0^1)^\top \mathbf{C}^\beta \text{ and } (w_0^1|\ldots|w_0^1)^\top \mathbf{C}^0,$$

where $w_0^1, \ldots, w_n^1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\mathbf{C}^\beta := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta)$, and $\mathbf{C}^0 := \mathsf{EncCt}(\mathsf{P}^0, \vec{x}^0)$. The leftmost distribution corresponds to game $\mathsf{H}_{1.Q}$, whereas the rightmost distribution corresponds to the game H_2. Thus, we have:

$$\mathsf{Adv}_{\mathsf{H}_{1.Q}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{H}_2}(\mathcal{A}).$$

$\mathsf{H}_{1.q}$, $\mathsf{H}_{1.q.1}$,$\mathsf{H}_{1.q.2}$, $\mathsf{H}_{1.q.3}$, H_2:

$\beta \leftarrow_{\mathrm{R}} \{0,1\}$, $\mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda)$, $(n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow \mathsf{Param}(\mathcal{F})$, let $(\vec{b}|\vec{b}_2|\vec{b}_3)$ and $(\vec{b}^*|\vec{b}_2^*|\vec{b}_3^*)$ be two random dual basis of $\mathbb{Z}_p^3$. For all $i \in [0,n]$, $j \in [3]$, $\vec{w}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$. We write $\vec{w}_i := w_i^1\vec{b}^* + w_i^2\vec{b}_1^* + w_i^3\vec{b}_3^*$

$\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big) \leftarrow \mathcal{A}(1^\lambda)$

$\mathsf{ct}^\star \leftarrow_{\mathrm{R}} \mathsf{OEnc}\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big)$

$\beta' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{ct}^\star)$

Return 1 if $\beta' = \beta$, 0 otherwise.

$\mathsf{OEnc}\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big)$: $\mathsf{H}_{1.q}$, $\mathsf{H}_{1.q.1}$, $\boxed{\mathsf{H}_{1.q.2}, \mathsf{H}_{1.q.3}}$, $\boxed{\mathsf{H}_2}$

$\mathbf{C}^\beta := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta)$, $\mathbf{C}^0 := \mathsf{EncCt}(\mathsf{P}^0, \vec{x}^0)$

$\vec{c}_2^{\beta\top} := (\vec{b}^{*\top}w_0^1 + \vec{b}_2^{*\top}w_0^2|\ldots|\vec{b}^{*\top}w_n^1 + \vec{b}_2^{*\top}w_n^2)\mathbf{C}^\beta$

$\boxed{\vec{c}_2^{\beta\top} := (\vec{b}^{*\top}w_0^1|\ldots|\vec{b}^{*\top}w_n^1)\mathbf{C}^\beta}$

$\boxed{\vec{c}_2^{\beta\top} := \mathbf{0}^\top}$

$\vec{c}_2^{0\top} := (\vec{b}_3^{*\top}w_0^3|\ldots|\vec{b}_3^{*\top}w_n^3)\mathbf{C}^0$

$\boxed{\vec{c}_2^{0\top} := (\vec{b}_2^{*\top}w_0^2 + \vec{b}_3^{*\top}w_0^3|\ldots|\vec{b}_2^{*\top}w_n^2 + \vec{b}_3^{*\top}w_n^3)\mathbf{C}^0}$

$\boxed{\vec{c}_2^{0\top} := (\vec{w}_0^\top|\ldots|\vec{w}_n^\top)\mathbf{C}^0}$

$\vec{c}_2 := \vec{c}_2^\beta + \vec{c}_2^0$

Return $\mathsf{ct}^\star := (\mathsf{part}(\mathsf{P}^\beta, \vec{x}^\beta), \vec{c}_2)$

$\mathsf{OKeyGen}(\mathsf{att}, \vec{y})$: $\mathsf{H}_{1.q}$, $\boxed{\mathsf{H}_{1.q.1},\mathsf{H}_{1.q.2}}$, $\boxed{\mathsf{H}_{1.q.3}}$, H_2

On the ρ-th query if $\rho \le q$, then $\vec{k}_1 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}_3)$. If $\rho > q$, then $\vec{k}_1 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b})$.

$\boxed{\text{If } \rho = q+1, \text{ then } \vec{k}_1 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}_2).}$ $\boxed{\text{If } \rho = q+1, \text{ then } \vec{k}_1 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}_3).}$

For all ρ, $\vec{k}_1 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}_3)$

$\mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y})$, $[\vec{k}_2^\top]_2 := [(\vec{w}_0^\top\vec{k}_1|\vec{w}_1^\top\vec{k}_1|\cdots|\vec{w}_n^\top\vec{k}_1)\mathbf{K}]_T$, $[k_3]_2 := [\vec{w}_0^\top\vec{k}_1]_2$.

Return $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\vec{k}_2^\top]_2, [k_3]_2)$

Fig. 9. Hybrid games for the proof of Lemma 3.7, where $q \in [Q]$, and Q denotes the number of queries to $\mathsf{OKeyGen}$.

Assumptions. All assumptions are relative to a pairing group $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \mathsf{PGGen}(1^\lambda)$ and two random dual basis $(\vec{b}|\vec{b}_2|\vec{b}_3)$ and $(\vec{b}^*|\vec{b}_2^*|\vec{b}_3^*)$, that is, the vectors are sampled randomly subject to: $\vec{b}_2^\top \vec{b}^* = \vec{b}_3^\top \vec{b}^* = 0$, $\vec{b}^\top \vec{b}^* = 1$, $\vec{b}^\top \vec{b}_2^* = \vec{b}_3^\top \vec{b}_2^* = 0$, $\vec{b}_2^\top \vec{b}_2^* = 1$, $\vec{b}^\top \vec{b}_3^* = \vec{b}_2^\top \vec{b}_3^* = 0$, $\vec{b}_3^\top \vec{b}_3^* = 1$. For any assumption i and PPT adversary $\mathcal{A}$, we define the advantage $\mathsf{Adv}^{\mathrm{As-i}}_{\mathcal{PG},\mathcal{A}}(\lambda)$ accordingly.

Assumption 1: $\left(\vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}^*, \vec{b}_2^*), \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, \boxed{[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}]_2)}\right) \approx_c$
$\left(\vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}^*, \vec{b}_2^*), \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, \boxed{[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}]_2, [\vec{b}_2]_2)}\right)$.

Assumption 2: $\left(\vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}^*, \vec{b}_2^*), \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, \boxed{[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}_2]_2)}\right) \approx_c$
$\left(\vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}^*, \vec{b}_2^*), \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, \boxed{[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}]_2, [\vec{b}_2]_2)}\right)$.

Assumption 3: $\left(\vec{b}^*, \vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}_2^*, \vec{b}_3^*), [\vec{b}]_2, [\vec{b}_3]_2, \boxed{[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}_2]_2)}\right) \approx_c$
$\left(\vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}^*, \vec{b}_2^*), \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, \boxed{[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}_2]_2, [\vec{b}_3]_2)}\right)$.

Assumption 4: $\left(\vec{b}^*, \vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}_2^*, \vec{b}_3^*), [\vec{b}]_2, [\vec{b}_3]_2, \boxed{[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}_3]_2)}\right) \approx_c$
$\left(\vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b}^*, \vec{b}_2^*), \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, \boxed{[\vec{z}]_2 \leftarrow_{\mathrm{R}} \mathsf{span}([\vec{b}_2]_2, [\vec{b}_3]_2)}\right)$.

Lemma 3.8 (Reduction from DDH to the assumptions [OT09, Lew12]). *For all the above assumptions, there is a reduction from DDH in $\mathbb{G}_2$.*

Proof. Let $\mathcal{A}$ be a PPT adversary. For each assumption i, we build a PPT adversary $\mathcal{B}_i$ that has a greater advantage in breaking DDH in $\mathbb{G}_2$ that the advantage of $\mathcal{A}$ in breaking Assumption i. Upon receiving $(\mathcal{PG}, [\vec{a}]_2, [\vec{t}]_2)$, where $\mathcal{PG} \leftarrow_{\mathrm{R}} \mathsf{PGGen}(1^\lambda)$, $\vec{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $[\vec{t}]_2 := [\vec{a}s]_2$ with $s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, or $[\vec{t}]_2 \leftarrow_{\mathrm{R}} \mathbb{G}_2^2$, adversary $\mathcal{B}_i$ does the following:

Assumption 1: $\mathcal{B}_1$ samples $\vec{u}_1, \vec{u}_2, \vec{b}_3^* \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$, compute $[\widetilde{a}]_2$ such that $\binom{\vec{a}}{\widetilde{a}}^\top \vec{b}_3^* = 0$. Writing $\vec{b}_3^* := \begin{pmatrix}\alpha\\ \beta\\ \gamma\end{pmatrix}$ and assuming $\gamma \neq 0$, this can be done efficiently by setting $[\widetilde{a}]_2 := \frac{-[\vec{a}]_2^\top \binom{\alpha}{\beta}}{\gamma}$. Similarly, compute $[\widetilde{t}]_2 := \frac{-[\vec{t}]_2^\top \binom{\alpha}{\beta}}{\gamma}$. Set $[\vec{b}]_2 := \begin{bmatrix}\vec{a}\\ \widetilde{a}\end{bmatrix}_2$, $[\vec{z}]_2 := \begin{bmatrix}\vec{z}\\ \widetilde{z}\end{bmatrix}_2$ sample $\vec{b}_3 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$ such that $\vec{b}_3^\top \vec{b}_3^* = 0$ and return $\left(\vec{u}_1, \vec{u}_2, \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, [\vec{z}]_2\right)$ to $\mathcal{A}$.

Assumption 2: $\mathcal{B}_2$ samples $\vec{u}_1, \vec{u}_2, \vec{b}_3^* \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$, compute $[\widetilde{\vec{z}}]_2$ as $\mathcal{B}_1$. Set $[\vec{z}]_2 := \begin{bmatrix}\vec{z}\\ \widetilde{z}\end{bmatrix}_2$, sample $\vec{b}, \vec{b}_3 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$ such that $\vec{b}^\top \vec{b}_3^* = \vec{b}_3^\top \vec{b}_3^* = 0$, and return $\left(\vec{u}_1, \vec{u}_2, \vec{b}_3^*, [\vec{b}]_2, [\vec{b}_3]_2, [\vec{z}]_2\right)$ to $\mathcal{A}$.

Assumption 3: $\mathcal{B}_3$ samples $\vec{b}^*, \vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$, compute $[\widetilde{z}]_2$ such that $\binom{\vec{z}}{\widetilde{z}}^\top \vec{b}^* = 0$, which can be done efficiently, as $\mathcal{B}_1$ (replacing $\vec{b}_3^*$ with $\vec{b}^*$). Set $[\vec{z}]_2 := \begin{bmatrix}\vec{z}\\ \widetilde{z}\end{bmatrix}_2$, sample $\vec{b}, \vec{b}_3 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$ such that $\vec{b}_3^\top \vec{b}_3^* = 0$ and $\vec{b}^\top \vec{b}^* = 1$, and return $\left(\vec{b}^*, \vec{u}_1, \vec{u}_2, [\vec{b}]_2, [\vec{b}_3]_2, [\vec{z}]_2\right)$ to $\mathcal{A}$.

Assumption 4: $\mathcal{B}_4$ samples $\vec{b}^*, \vec{u}_1, \vec{u}_2 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$, compute $[\widetilde{a}]_2$ and $[\widetilde{z}]_2$ such that $\binom{\vec{a}}{\widetilde{a}}^\top \vec{b}^* = 0$ and $\binom{\vec{z}}{\widetilde{z}}^\top \vec{b}^* = 0$, respectively. Set $[\vec{b}_3]_2 := \begin{bmatrix}\vec{a}\\ \widetilde{a}\end{bmatrix}_2$, $[\vec{z}]_2 := \begin{bmatrix}\vec{z}\\ \widetilde{z}\end{bmatrix}_2$, sample $\vec{b} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$ such that $\vec{b}^\top \vec{b}^* = 1$, and return $\left(\vec{b}^*, \vec{u}_1, \vec{u}_2, [\vec{b}]_2, [\vec{b}_3]_2, [\vec{z}]_2\right)$ to $\mathcal{A}$.

$\underline{\mathsf{H}_0, \boxed{\mathsf{H}_1}}$:

$\beta \leftarrow_{\mathrm{R}} \{0,1\}$, $\mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda)$, $(n, |\mathsf{ct}|, |\mathsf{sk}|) \leftarrow \mathsf{Param}(\mathcal{F}_{\mathsf{ipfe}(d,B)}, \mathcal{P})$, let $(\vec{b}|\vec{b}_2|\vec{b}_3)$ and $(\vec{b}^*|\vec{b}_2^*|\vec{b}_3^*)$ be two random dual basis of $\mathbb{Z}_p^3$. For all $i \in [0,n]$, $\vec{w}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$. We write $\vec{w}_i := w_i^1\vec{b}^* + w_i^2\vec{b}_1^* + w_i^3\vec{b}_3^*$, with $w_i^1, w_i^2, w_i^3 \leftarrow_{\mathrm{R}} \mathbb{Z}_p$

$\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big) \leftarrow \mathcal{A}(1^\lambda)$

$\mathsf{ct}^\star \leftarrow_{\mathrm{R}} \mathsf{OEnc}\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big)$

$\beta' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot)}(\mathsf{ct}^\star)$

Return 1 if $\beta' = \beta$, 0 otherwise.

$\underline{\mathsf{OEnc}\Big((\mathsf{P}^0, \vec{x}^0), (\mathsf{P}^1, \vec{x}^1)\Big)}$:

$\mathbf{C}^\beta := \mathsf{EncCt}(\mathsf{P}^\beta, \vec{x}^\beta)$, $\mathbf{C}^0 := \mathsf{EncCt}(\mathsf{P}^0, \vec{x}^0)$,

$\vec{c}_2^{\beta\top} := (\vec{w}_0^\top|\ldots|\vec{w}_n^\top)\mathbf{C}^\beta$, $\boxed{\vec{c}_2^{\beta\top} := (\vec{b}^{*\top}w_0^1 + \vec{b}_2^{*\top}w_0^2|\ldots|\vec{b}^{*\top}w_n^1 + \vec{b}_2^{*\top}w_n^2)\mathbf{C}^\beta}$,

$\vec{c}_2^{0\top} := \mathbf{0}^\top$, $\boxed{\vec{c}_2^{0\top} := (\vec{b}_3^{*\top}w_0^3|\ldots|\vec{b}_3^{*\top}w_n^3)\mathbf{C}^0}$,

$\vec{c}_2 := \vec{c}_2^{\beta} + \vec{c}_2^{0}$.

Return $\mathsf{ct}^\star := (\mathsf{part}(\mathsf{P}^\beta, \vec{x}^\beta), \vec{c}_2)$

$\underline{\mathsf{OKeyGen}(\mathsf{att}, \vec{y})}$:

$\vec{k}_1 \leftarrow_{\mathrm{R}} \mathsf{span}(\vec{b})$, $\mathbf{K} := \mathsf{EncKey}(\mathsf{att}, \vec{y})$, $[\vec{k}_2^\top]_2 := [(\vec{w}_0^\top\vec{k}_1|\vec{w}_1^\top\vec{k}_1|\cdots|\vec{w}_n^\top\vec{k}_1)\mathbf{K}]_2$, $[k_3]_2 := [\vec{w}_0^\top\vec{k}_1]_2$. Return $(\mathsf{att}, \vec{y}, [\vec{k}_1]_2, [\vec{k}_2]_2, [k_3]_2)$

Fig. 10. Hybrid games for the proofs of Theorem 3.6 and Lemma 3.7.

4 A Lattice-Based Identity-Based Functional Encryption in the Random-Oracle Model

In this section, we build an identity-based functional encryption (IFE) for the inner-product functionality from LWE in the random-oracle model. In Section 5, we provide a lattice-based scheme that is proven secure in the standard model.

4.1 Lattice Preliminaries

Matrix norms For a vector $\vec{u}$, $\|\vec{u}\|$ denotes its ℓ_2 norm. For a matrix $\mathbf{R}$, by $\widetilde{\mathbf{R}}$ we denote the result of applying Gram Schmidt orthogonalization on the columns of $\mathbf{R}$. In addition:

- $\|\mathbf{R}\|$ denotes the ℓ_2 norm of the longest column of $\mathbf{R}$.
- $\|\mathbf{R}\|_2$ denotes the operator norm of $\mathbf{R}$, where $\|\mathbf{R}\|_2 = \sup_{\|\vec{x}\|=1}\|\mathbf{R}\vec{x}\|$, with $\vec{x} \in \mathbb{Z}^m$
- $s_1(\mathbf{R})$ denotes the spectral norm (the largest singular value of $\mathbf{R}$).

Known facts are that $\|\widetilde{\mathbf{R}}\| \leq \|\mathbf{R}\| \leq \|\mathbf{R}\|_2 \leq \sqrt{k}\|\mathbf{R}\|$. We will also need the fact that concatenating two matrices $\mathbf{R}, \mathbf{S}$ yields $s_1(\mathbf{R}|\mathbf{S}) \leq \sqrt{s_1(\mathbf{R})^2 + s_1(\mathbf{S}^2)}$. For bounding the spectral norm, we will require the following lemma:

Lemma 4.1. [DM14] *Let $\mathbf{X} \in \mathbb{R}^{n\times m}$ be a subgaussian random matrix with parameter s. There exists a universal constant $C \approx \frac{1}{\sqrt{2\pi}}$ such that for any $t \geq 0$, we have $s_1(\mathbf{X}) \leq C \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $\frac{2}{e^{\pi t^2}}$.*

Lattices. For any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and any vector $\vec{p} \in \mathbb{Z}_q^n$, we define the orthogonal q-ary lattice of $\mathbf{A}$: $\Lambda_q^\perp(\mathbf{A}) := \{\vec{u} \in \mathbb{Z}^m : \mathbf{A}\vec{u} = \vec{0} \bmod q\}$ and the shifted lattice: $\Lambda_q^{\vec{p}}(\mathbf{A}) := \{\vec{u} \in \mathbb{Z}^m : \mathbf{A}\vec{u} = \vec{p} \bmod q\}$. Similarly, for any matrix $\mathbf{P} \in \mathbb{Z}_q^{n\times\ell}$, we define: $\Lambda_q^{\mathbf{P}}(\mathbf{A}) := \{\mathbf{U} \in \mathbb{Z}^{m\times\ell} : \mathbf{A}\mathbf{U} = \mathbf{P} \bmod q\}$.

Probability distributions Consider a lattice $\Lambda \subseteq \mathbb{Z}^m$. The normal Gaussian distribution of mean 0 and variance σ^2 is the distribution on $\mathbb{R}$ with probability density function $\frac{1}{\sigma\cdot\sqrt{2\pi}}\frac{1}{e^{x^2/(2\sigma^2)}}$. The lattice gaussian distribution with support $\Lambda \subseteq \mathbb{R}^n$, standard deviation σ and centered at $\vec{c}$ is defined as:

$$\text{for all } \vec{y} \in \Lambda : D_{\Lambda,\sigma,\vec{c}}(\vec{y}) = \frac{e^{-\pi\|\vec{y}-\vec{c}\|^2/\sigma^2}}{\sum_{\vec{x}\in\Lambda} e^{-\pi\|\vec{x}-\vec{c}\|^2/\sigma^2}}$$

Lemma 4.2 (Bounding Gaussian Noise). [MR04] *For any n-dimensional lattice Λ, $\vec{c} \in span(\Lambda)$, real $\epsilon \in (0,1)$, and $s \geq \eta_\epsilon(\Lambda)$,*

$$\Pr_{x\leftarrow_R \mathcal{D}_{\Lambda,s,c}}[\|\vec{x}-\vec{c}\| > s\sqrt{n}] \leq \frac{1+\epsilon}{1-\epsilon}\frac{1}{2^n}.$$

Also from [MR04, Lemma 3.2], for any n-dimensional lattice Λ, $\eta_\epsilon(\Lambda) \leq \sqrt{n}/\lambda_1(\Lambda^*)$, where $\epsilon = \frac{1}{2^n}$. This was improved by [GPV08] to obtain that for any $\omega(\sqrt{\log n})$ function, there is a negligible $\epsilon(n)$ such that: $\eta_\epsilon(\mathbb{Z}) \leq \omega(\sqrt{\log n})$. In particular, when sampling integers, we have that for any $\epsilon \in (0, \frac{1}{2})$ any $s \geq \eta_\epsilon(\mathbb{Z})$, and any $t \geq \omega(\sqrt{\log n})$:

$$\Pr_{x\leftarrow_R \mathcal{D}_{\mathbb{Z},s,c}}[|x-c| > s\cdot t] \leq \mathsf{negl}(n).$$

Learning with errors (LWE) [Reg05] Let q be a prime, χ be a public distribution over $\mathbb{Z}_q$ and $\vec{s}$ be uniformly random over $\mathbb{Z}_q^n$. Moreover, $\vec{s}$ is constant across calls to oracles $\mathcal{O}_{\vec{f}}$ or $\mathcal{O}_\$$, defined below:

- Oracle $\mathcal{O}_{\vec{s}}$ outputs samples $(\vec{a}, \vec{a}^\top\vec{s}+\vec{x})$, where $\vec{a} \leftarrow_R \mathbb{Z}_q^n$ and $x \leftarrow_R \chi$ are fresh and independently sampled.
- Oracle $\mathcal{O}_\$$ outputs uniformly random elements of $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Define another oracle $\mathcal{O}$, which across all calls, is either $\mathcal{O}_{\vec{s}}$ or $\mathcal{O}_\$$. The learning with errors $\mathsf{LWE}_{q,\chi,n}$ problem is to distinguish with non-negligible probability, given access to oracle $\mathcal{O}$, whether it corresponds to $\mathcal{O}_{\vec{s}}$ or $\mathcal{O}_\$$.

Proposition 4.3 ([Reg05]). *Let $\alpha = \alpha(n) \in (0,1)$ and let $q = q(n)$ be a prime such that $\alpha\cdot q > 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that solves $\mathsf{LWE}_{q,\bar{\Psi}_\alpha}$, then there exists an efficient quantum algorithm for approximating* SIVP *and* GapSVP *in the ℓ_2 norm, in the worst case, to within $\widetilde{O}(n/\alpha)$ factors.*

In [Pei07], this result has been generalized for any ℓ_p norm, with $2 \leq p \leq \infty$ and for similar $\widetilde{O}(n/\alpha)$ approximation factors.

Theorem 4.4. *[GPV08, Theorem 4.1] There is a probabilistic polynomial-time algorithm* SampleD *that, given a basis $\mathbf{B}$ of an n-dimensional lattice $\Delta = \mathcal{L}(\mathbf{B})$, a parameter $s \geq \|\widetilde{\mathbf{B}}\|\cdot\omega(\sqrt{\log n})$, and a center $\vec{c} \in \mathcal{R}$, outputs a sample from a distribution that is statistically close to $D_{\Lambda,s,\vec{c}}$.*

Proposition 4.5. *[Ajt99] For any prime $q = poly(n)$ and any $m \geq 5n\lg q$, there is a probabilistic polynomial-time algorithm* SampleMat *that, on input 1^n, outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a full-rank set $\mathbf{S} \subset \Lambda_q^\perp(\mathbf{A})$, where the distribution of $\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$ and the length $\|\mathbf{S}\| \leq L = m^{2.5}$.*

Also, by [MG02, Lemma 7.1, page 129], $\mathbf{S}$ can be converted efficiently to a "good" basis $\mathbf{T}$ of $\Lambda_q^\perp(\mathbf{A})$ such that $\|\widetilde{\mathbf{T}}\| \leq \|\widetilde{\mathbf{S}}\| \leq L$.

Lemma 4.6 ([GPV08]Preimage Samplable Functions). *For any prime $q = poly(n)$, any $m \geq 5n\lg q$, and any $s \geq L\cdot\omega(\sqrt{\log m})$, it holds that there exist PPT algorithms* TrapGen, SampleD, SamplePre *such that:*

1. TrapGen *computes $(\mathbf{A},\mathbf{T}) \leftarrow_R$ TrapGen$(1^n,1^m)$, where $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ is statistically close to uniform and $T \subset \Lambda_q^\perp(\mathbf{A})$ is a good basis with $\|\widetilde{\mathbf{T}}\| \leq L$. The matrix $\mathbf{A}$ (and q) are public, while the good basis $\mathbf{T}$ is the trapdoor.*
2. SampleD *samples vectors $\vec{e}$ from $\mathcal{D}_{\mathbb{Z}^{\ell\times m},s}$.*
3. *The trapdoor inversion algorithm* SamplePre$(\mathbf{A},\mathbf{T},s,\mathbf{U})$ *outputs a matrix $\mathbf{Z} \in \mathbb{Z}^{\ell\times m}$ such that $\mathbf{U} = \mathbf{ZA}$.*

In addition, it holds that the following distributions D_1, D_2 are statistically close:

$$D_1 := (\mathbf{A},\mathbf{Z},\mathbf{U}),\ \text{s.t. } (\mathbf{A},\mathbf{T}) \leftarrow_R \mathsf{TrapGen}(1^n,1^m), \mathbf{U} \leftarrow_R \mathbb{Z}^{\ell\times n}, \mathbf{Z} \leftarrow_R \mathsf{SamplePre}(\mathbf{A},\mathbf{T},s,\mathbf{U})$$

$$D_2 := (\mathbf{A},\mathbf{Z},\mathbf{ZA}),\ \text{where } \mathbf{A} \leftarrow_R \mathbb{Z}_q^{m\times n}, \mathbf{Z} \leftarrow_R \mathcal{D}_{\mathbb{Z}^{\ell\times m},s} : \|\vec{z}_i\| \leq s\sqrt{m}, i \in \{1\ldots m\}$$

where $\vec{z}_i$ denotes the i-th row of $\mathbf{Z}$.

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y})}$:
$(\mathbf{A}, \mathbf{T}) \leftarrow_{\mathrm{R}} \mathsf{TrapGen}(1^n, 1^m)$
$\mathsf{mpk} \leftarrow \mathbf{A}, \mathsf{msk} \leftarrow \mathbf{T}$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \vec{x})}$:
$\mathbf{U}_{\mathsf{id}} \leftarrow H(\mathsf{id})$
$\vec{s} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^n$
$\vec{f}_1 \leftarrow_{\mathrm{R}} \mathcal{D}_{\mathbb{Z}^m, \sigma}$
$\vec{f}_2 \leftarrow_{\mathrm{R}} \mathcal{D}_{\mathbb{Z}^\ell, \sigma}$
$\mathsf{ct}_1 \leftarrow \mathbf{A}\vec{s} + \vec{f}_1$
$\mathsf{ct}_2 = \mathbf{U}_{\mathsf{id}}\vec{s} + \vec{f}_2 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x}$
Return $(\mathsf{ct}_1, \mathsf{ct}_2)$

$\underline{\mathsf{KeyGen}(\mathsf{id}, \vec{y})}$:
$\mathbf{U}_{\mathsf{id}} \leftarrow H(\mathsf{id})$
$\mathbf{Z}_{\mathsf{id}} \leftarrow_{\mathrm{R}} \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \rho, \mathbf{U}_{\mathsf{id}})$
Return $(\vec{y}, \mathsf{sk}_{\mathsf{id},\vec{y}} := (\vec{y}^\top \cdot \mathbf{Z}_{\mathsf{id}}))$

$\underline{\mathsf{Dec}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{sk}_{\mathsf{id},\vec{y}}, \vec{y})}$:
$\mu = \vec{y}^\top \cdot \mathsf{ct}_2 - \mathsf{sk}_{\mathsf{id},\vec{y}} \cdot \mathsf{ct}_1$
$\mu' = \arg\min_{\mu' \in \{0 \ldots K+1\}} \left| \left\lfloor \frac{q}{K} \right\rfloor \cdot \mu - \mu' \right|$
Return μ'

Fig. 11. An identity-based inner-product functional encryption scheme $\mathcal{IFE}$ in the random-oracle model, where H denotes the random oracle. Algorithms $\mathsf{TrapGen}$ and $\mathsf{SamplePre}$ are referenced in Lemma 4.6

Noise Rerandomization. The following procedure $\mathsf{NoiseGen}(\mathbf{R}, s)$ for noise rerandomization, was described in [KY16]. $\mathsf{NoiseGen}(\mathbf{R}, s)$: given a matrix $\mathbf{R} \in \mathbb{Z}^{m \times t}$, and $s \in \mathbb{R}^+$ such that $s^2 > s_1(\mathbf{R}\mathbf{R}^\top)$, it first samples $\vec{e}_1 := \mathbf{R}\vec{e} + (s^2\mathbf{I}_m - \mathbf{R}\mathbf{R}^\top)^{\frac{1}{2}}\vec{e}'$, where $\mathbf{I}_m \in \mathbb{Z}^{m \times m}$ denotes the identity matrix, and $\vec{e} \leftarrow_{\mathrm{R}} \mathcal{D}_\sigma^t$ and $\vec{e}' \leftarrow_{\mathrm{R}} \mathcal{D}^m_{\sqrt{2}\sigma}$ are independent spherical continuous Gaussian noises. Then, it samples $\vec{e}_2 \leftarrow_{\mathrm{R}} \mathcal{D}_{\mathbb{Z}^m - \vec{e}_1, s\sqrt{2}\sigma}$, and returns $\vec{e}_1 + \vec{e}_2 \in \mathbb{Z}_q^m$. We have the following lemma:

Lemma 4.7 (Noise Distribution [KY16]). *Let $\mathbf{R} \leftarrow_{\mathrm{R}} \mathbb{Z}^{m \times t}$ and $s > s_1(\mathbf{R})$. The following distributions are statistically close:*
***Distribution 1**: $\vec{e} \leftarrow_{\mathrm{R}} \mathcal{D}_{\mathbb{Z}^t, \sigma}$ and $\vec{e}' \leftarrow_{\mathrm{R}} \mathsf{NoiseGen}(\mathbf{R}, s)$. Output $\mathbf{R}\vec{e} + \vec{e}'$.*
***Distribution 2**: Output $e \leftarrow_{\mathrm{R}} \mathcal{D}_{\mathbb{Z}^m, 2s\sigma}$.*

4.2 Our Construction

In this section, we describe how to obtain an identity-based inner-product funtional encryption scheme based on the hardness of LWE in the random-oracle model. Our idea is to start with a modification of the ALS functional encryption scheme for inner-products [ALS16], proposed by [WFL19] and which we recall in Appendix A. We modify the identity-based encryption scheme of [GPV08] in such a way as to support functional key generation queries, as in ALS. Our construction is described in Fig. 11. Ciphertexts encode vectors $\vec{x} \in \mathcal{X} := \{0, \ldots, P-1\}^\ell$ under an identity id. Secret keys correspond to an identity id and a vector $\vec{y} \in \mathcal{Y} := \{0, \ldots, V-1\}^\ell$. When the identities match, our scheme decrypts the bounded inner-product $\langle \vec{x}, \vec{y} \rangle \in \{0, \ldots, K-1\}$ where $K = \ell PV$.

Since our construction achieves anonymity and the size of input vectors $\vec{x}$ are fixed, no partial information about the input is leaked. That is, $\mathsf{part}(\vec{x}, \mathsf{id}) = \perp$.

Lemma 4.8 (Correctness). *For $q \geq 2K\ell\sqrt{\ell}V\omega(\log^2 n)$, $\sigma = 2C\alpha q(\sqrt{m}+\sqrt{n}+\sqrt{\ell})$, $\rho \geq \omega(\sqrt{\log n})$, $m = 2n \log q$, the scheme from Fig. 11 is correct.*

Proof. When identities match, observe that decryption yields $\vec{y}^\top \mathbf{U}\vec{s} + \vec{y}^\top \vec{f}_2 + \vec{y}^\top \mathbf{Z}\mathbf{A}\vec{s} + \vec{y}^\top \mathbf{Z}\vec{f}_1 + \left\lfloor \frac{q}{K} \right\rfloor \langle \vec{x}, \vec{y} \rangle$, which is equal to:

$$\underbrace{\vec{y}^\top \vec{f}_2 + \vec{y}^\top \mathbf{Z}\vec{f}_1}_{\text{error terms}} + \left\lfloor \frac{q}{K} \right\rfloor \langle \vec{x}, \vec{y} \rangle$$

This decrypts correctly as long as the error terms are small. From Lemma 4.2, we know that every entry of $\mathbf{Z}$ is with overwhelming probability bounded by $\omega(\log n)$, so $\|\mathbf{Z}\| \leq \sqrt{\ell} \cdot \omega(\log n)$, as long as $\rho \geq \omega(\sqrt{\log n})$. We use Lemma 4.2 once again and we bound $\|\vec{y}^\top \mathbf{Z}\vec{e}_1\| \leq \ell\sqrt{\ell}V\omega(\log^2 n)$ and $\|\vec{y}\vec{e}_2\| \leq \ell V\omega(\sqrt{\log n})$, as long as

$\sigma \geq \omega(\sqrt{\log n})$. For decryption to succeed, we want that the error terms are smaller than $\frac{q}{2K}$, which implies: $q \geq 2K\ell\sqrt{\ell}V\omega(\log^2 n)$, which is the case for our choice of parameters.

Remark 4.9 (No smudging noise). We remark that in our setup, we rely on efficient lattice parameters and require no smudging or superpolynomial modulus.

Theorem 4.10 (Security). *Let n be the security parameter, $q \geq 2K\ell\sqrt{\ell}V\omega(\log^2 n)$, $\sigma = 2C\alpha q(\sqrt{m}+\sqrt{n}+\sqrt{\ell})$, $\rho \geq \omega(\sqrt{\log n})$, $m = 2n\log q$, $\alpha \leq \frac{\sigma}{2C\alpha q(\sqrt{m}+\sqrt{n}+\sqrt{\ell})}$, then the scheme from Fig. 11 is AD-IND-secure in the random-oracle model, assuming that $\mathsf{LWE}_{q,\alpha,n}$ is hard.*

Proof. We prove adaptive security in the random-oracle model, following the proof structure and techniques of [GPV08], while making several changes to adapt the proof techniques to functional encryption. First, without loss of generality, we assume that any adversary making key generation queries of the form $\mathsf{id}\|\vec{y}$ will first query the random oracle on id (we can make this assumption because for every adversary $\mathcal{A}$, we can compile it into an adversary $\mathcal{A}'$ that exhibits this behavior). We proceed in a series of hybrids, consider $\mathcal{A}$ to be a PPT adversary, and $n \in \mathbb{N}$ to be the security parameter. We denote by $\mathbf{Adv}_{\text{Game}_i}(\mathcal{A})$ the advantage of $\mathcal{A}$ in game i.

Game 0 This is the original AD-IND game.

Game 1 In this game, we guess what identities id_0^* and id_1^* will be used for the challenge messages. Guessing $\mathsf{id}_0^*, \mathsf{id}_1^*$ themselves would incur an exponential security loss. Therefore, instead of guesssing the identities themselves, we guess the index of the random-oracle query in which the adversary queries the oracle H to get $\mathbf{U}_{\mathsf{id}_0^*}$ and $\mathbf{U}_{\mathsf{id}_1^*}$. This will result in an $\frac{1}{Q^2}$ security loss, where Q is the total number of queries to the random oracle. As explained above, we are guaranteed that the adversary will make this query.

Depending on whether $\mathsf{id}_0^* = \mathsf{id}_1^*$, we will either perform a reduction to the ALS scheme (see Appendix A) or directly to LWE.

Case 1, $\mathsf{id}_0^* = \mathsf{id}_1^*$ We perform a reduction to the security of the ALS [ALS16] encryption scheme, which we recall in Appendix A. We reduce to the AD-IND security of ALS. We first obtain from the challenger public keys $\mathbf{A}_{\text{ALS}}, \mathbf{D}_{\text{ALS}}$. Now, equipped with the knowledge of id^*, we define $\mathbf{Game}_1$ to be the same as $\mathbf{Game}_0$, except for the following changes:

- Matrix $\mathbf{A}$ is not generated with $\mathsf{TrapGen}$ anymore, instead $\mathbf{A}$ is now replaced with $\mathbf{A}_{\text{ALS}}$.
- For every $\mathsf{id} \neq \mathsf{id}^*$ query to the random oracle H, we draw a matrix $\mathbf{Z}_{\mathsf{id}} \leftarrow_{\text{R}} \mathcal{D}_{\mathbb{Z}^{\ell\times m},\rho}$. We program the random oracle $H(\mathsf{id}) = \mathbf{U}_{\mathsf{id}}$, where $\mathbf{U}_{\mathsf{id}}$ is now computed as $\mathbf{U}_{\mathsf{id}} = \mathbf{Z}_{\mathsf{id}}\mathbf{A}$.
  Key queries of the form $\mathsf{id}\|\vec{y}$ are answered by returning $\vec{y}^\intercal \mathbf{Z}_{\mathsf{id}}$.
- For $\mathsf{id} = \mathsf{id}^*$, we program $H(\mathsf{id}) = \mathbf{D}_{\text{ALS}}$.
- For key queries of the form $\mathsf{id}^*\|\vec{y}$, we notice that we cannot answer them ourselves. However, we are allowed to forward $\vec{y}$ to the challenger of the AD-IND security of ALS, which replies with $\vec{y}^\intercal\mathbb{Z}_{\text{ALS}}$, where $\mathbf{Z}_{\text{ALS}}$ is the master secret key of the ALS scheme. We set $\mathsf{sk}_{\mathsf{id}^*\|\vec{y}} := \vec{y}^\intercal\mathbb{Z}_{\text{ALS}}$ and forward it to the adversary.
- When the adversary finally submits its challenge $(\vec{x}_0, \vec{x}_1)$, we forward this to the ALS challenger, which replies with $\mathsf{ct} := (\mathsf{ct}_1, \mathsf{ct}_2)$. We forward ct back to the adversary.

We therefore have that $\mathbf{Adv}_{\text{Game}_1}(\mathcal{A}) \leq \mathbf{Adv}_{\text{ALS}}(\mathcal{A})$, now we only need to show that $|\mathbf{Adv}_{\text{Game}_0}(\mathcal{A}) - \mathbf{Adv}_{\text{Game}_1}(\mathcal{A})| \leq \mathsf{negl}(n)$. For this we use Lemma 4.6, which tells us that:

For any $\mathsf{id} \neq \mathsf{id}^*$, $D_0 := (\mathbf{A}, \mathbf{Z}_{\mathsf{id}}, \mathbf{U}_{\mathsf{id}})$, where $(\mathbf{A}, \mathbf{T}) \leftarrow_{\text{R}} \mathsf{TrapGen}(1^n)$, $\mathbf{U}_{\mathsf{id}} \leftarrow_{\text{R}} \mathbb{Z}^{\ell\times n}$, $\mathbf{Z}_{\mathsf{id}} \leftarrow_{\text{R}} \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, s, \mathbf{U}_{\mathsf{id}})$, (which is the same as the adversarial view in Game_0) is statistically close to $D_1 := (\mathbf{A}, \mathbf{Z}_{\mathsf{id}}, \mathbf{Z}_{\mathsf{id}}\mathbf{A})$, where $\mathbf{A} \leftarrow_{\text{R}} \mathbb{Z}_q^{m\times n}$, $\mathbf{Z}_{\mathsf{id}} \leftarrow_{\text{R}} \mathcal{D}_{\mathbb{Z}^{\ell\times m}} : \|\vec{z}_i\| \leq s\sqrt{m}$ (recall that $\vec{z}_i$ denotes the i-th row of $\mathbf{Z}$). Note that in ALS, matrix $\mathbf{A}_{\text{ALS}}$ is indeed uniformly random, which corresponds to how $\mathbf{A}$ is chosen in Game_1.

Finally, when $\mathsf{id} = \mathsf{id}^*$, we need to argue that ALS public keys are distributed statistically close to honestly generated keys in Game_0. Note that in ALS, the second matrix $\mathbf{D}_{\text{ALS}}$ is not uniformly random like in Game_0, but instead is of the form $\mathbf{D}_{\text{ALS}} = \mathbf{Z}_{\text{ALS}}\mathbf{A}_{\text{ALS}}$ (in fact in can be proven that $\mathbf{U}$ is not statistically close to uniformly random). However, this is not an issue in our case, as the distribution of $\mathbf{U}_{\mathsf{id}^*}$ is in fact uniform conditioned on it being of the form $\mathbf{Z}_{\mathsf{id}^*}\mathbf{A}$. Therefore, we can apply Lemma 4.6 once again, and conclude that $(\mathbf{A}, \mathbf{Z}_{\mathsf{id}^*}, \mathbf{U}_{\mathsf{id}^*})$ with trapdoored $\mathbf{A}$ and uniform $\mathbf{U}_{\mathsf{id}^*}$ is statistically close to $(\mathbf{A}, \mathbf{Z}_{\mathsf{id}^*}, \mathbf{Z}_{\mathsf{id}^*}\mathbf{A})$.

Case 2, $\mathsf{id}_0^* \neq \mathsf{id}_1^*$ In this case, we make the following observation: if the adversary was allowed to obtain secret keys for either $\mathsf{id}_0^* \| \vec{y}$ or $\mathsf{id}_1^* \| \vec{y}$, for any $\vec{y}$, then it could trivially distinguish between encryptions of $\vec{x}_0$ under id_0^* and encryptions of $\vec{x}_1$ under id_1^*. This type of trivial attack is excluded by the AD-IND definition, therefore we conclude that the adversary cannot obtain decryption key queries for neither id_0^* or id_1^*. With this limitation, we can perform a reduction directly to LWE for matrix $\mathbf{B} = \left(\begin{array}{c} \mathbf{A} \\ \hline \mathbf{U}_{\mathsf{id}_0^*} \\ \hline \mathbf{U}_{\mathsf{id}_1^*} \end{array}\right) \in \mathbb{Z}_q^{(m+2\ell)\times n}$, where $\mathbf{B}$ is uniformly random. We obtain a ciphertext $\vec{c} = (\vec{c}_1, \vec{c}_2, \vec{c}_3)$ from the LWE oracle. To simulate the view of $\mathcal{A}$, we set $H(\mathsf{id}_0^*) = \mathbf{U}_{\mathsf{id}_0^*}$ and $H(\mathbf{U}_{\mathsf{id}_1^*})$. For all the other keys, we proceed as in the previous case:

1. For every $\mathsf{id} \notin \{\mathsf{id}_0^*, \mathsf{id}_1^*\}$ query to the random oracle H, we draw a matrix $\mathbf{Z}_{\mathsf{id}} \leftarrow_{\text{R}} \mathcal{D}_{\mathbb{Z}^{\ell\times m},\rho}$. We program the random oracle $H(\mathsf{id}) = \mathbf{U}_{\mathsf{id}}$, where $\mathbf{U}_{\mathsf{id}}$ is now computed as $\mathbf{U}_{\mathsf{id}} = \mathbf{Z}_{\mathsf{id}}\mathbf{A}$.
2. For every $\mathsf{id} \notin \{\mathsf{id}_0^*, \mathsf{id}_1^*\}$, key queries of the form $\mathsf{id}\|\vec{y}$ are answered by returning $\vec{y}^\top \mathbf{Z}_{\mathsf{id}}$.

Challenge queries to answer a challenge encryption for $\vec{x}_0$ under id_0^* or $\vec{x}_1$ under id_1^*, we pick uniformly at random a bit β and reply with:

$$(\vec{c}_0, \vec{c}_1 + \left\lfloor \tfrac{q}{K} \right\rfloor \cdot \vec{x}_0), \text{ if } \beta = 0 \quad \text{and} \quad (\vec{c}_0, \vec{c}_2 + \left\lfloor \tfrac{q}{K} \right\rfloor \cdot \vec{x}_1), \text{ if } \beta = 1$$

If the LWE oracle outputs a uniformly random element, then $\mathbf{Adv}_{\text{Game}_1}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{LWE}_{q,\alpha,n}}(\mathcal{A})$ and the view is identical to Game_1. When the LWE oracle outputs an element of the form $\vec{c} = \mathbf{B}\vec{d} + \vec{g}$, with $\vec{d} \leftarrow_{\text{R}} \mathbb{Z}_q^n$ and $\vec{g} \leftarrow_{\text{R}} \mathcal{D}_{\mathbb{Z}^{m+2\ell}}$, the split into $(\vec{c}_0, \vec{c}_1, \vec{c}_2) \in \mathbb{Z}_q^m \times (\mathbb{Z}_q^\ell)^2$ produces ciphertexts formed exactly as in Game_0.

We are left to argue that $(\mathbf{A}, \mathbf{U}_{\mathsf{id}_0^*}, \mathbf{U}_{\mathsf{id}_1^*})$ in Game_1 is distributed statistically close as in Game_2, which is the case by Lemma 4.6.

We have thus proven that:

$$\mathbf{Adv}_{\text{IND-FE-CPA}}(\mathcal{A}) \leq \tfrac{1}{Q^2} \max(\mathbf{Adv}_{\mathsf{LWE}_{q,\alpha,n}}(\mathcal{A}), \mathbf{Adv}_{\mathsf{LWE}_{q,\sigma/q,n}}(\mathcal{A})) \leq \tfrac{1}{Q^2}\mathbf{Adv}_{\mathsf{LWE}_{q,\alpha,n}}(\mathcal{A}),$$

where Q is the number of random-oracle queries.

5 A Lattice-Based Identity-Based Functional Encryption in the Standard Model

In this section, we present a lattice-based IFE construction for the inner-product functionality in the standard model. However, before doing so, we first recall some known results that we are going to use in our construction and security proofs.

5.1 Preliminaries

Lemma 5.1 ([BGG+14]). *Let n, m, $q > 0$ be integers, where q prime. There exist polynomial time algorithms, such that:*

- $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{A}, \mathbf{T_A})$ ([Ajt99, AP09, MP12]): *a PPT algorithm that, for $m = \Theta(n \log q)$, outputs a full-rank $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T_A} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^\perp(\mathbf{A})$. In addition, $\mathbf{A}$ is $\mathsf{negl}(n)$-close to uniform and $\|\widetilde{\mathbf{T}}_\mathbf{A}\| = O(\sqrt{n \log q})$, with overwhelming probability in n.*
- $\mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}) \to \mathbf{T}_{(\mathbf{A}|\mathbf{B})}$ ([CHKP10]): *a deterministic algorithm that for full-rank $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T_A} \in \mathbb{Z}^{m \times m}$ of $\Lambda_q^\perp(\mathbf{A})$, outputs a basis $\mathbf{T}_{(\mathbf{A}|\mathbf{B})}$ of $\Lambda_q^\perp(\mathbf{A}|\mathbf{B})$. In addition $\|\widetilde{\mathbf{T}}_\mathbf{A}\| = \|\widetilde{\mathbf{T}}_{(\mathbf{A}|\mathbf{B})}\|$.*
- $\mathsf{ExtendLeft}(\mathbf{A}, \mathbf{G}, \mathbf{T_G}, \mathbf{S}) \to \mathbf{T}_{(\mathbf{A}|\mathbf{G}+\mathbf{AS})}$ ([ABB10]): *a deterministic algorithm that for full-rank $\mathbf{A}, \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T_G}$ of $\Lambda_q^\perp(\mathbf{G})$, outputs a basis $\mathbf{T}_{(\mathbf{A}|\mathbf{G}+\mathbf{AS})}$ of $\Lambda_q^\perp(\mathbf{A}|\mathbf{G}+\mathbf{AS})$. In addition $\|\widetilde{\mathbf{T}}_{(\mathbf{A}|\mathbf{G}+\mathbf{AS})}\| \leq \|\widetilde{\mathbf{T}}_\mathbf{G}\|(1 + \|\mathbf{S}\|_2)$.*
- *For $m = n$, there exists a full-rank gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ for which $\Lambda_q^\perp(\mathbf{G})$ has a publicly known basis $\mathbf{T_G}$, where $\|\widetilde{\mathbf{T}}_\mathbf{G}\| \leq \sqrt{5}$*

Theorem 5.2 (SampleLeft [ABB10]). *Let $q > 2$, full rank $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$ with $m > n$, a basis $\mathbf{T_A}$ of $\Lambda_q^\perp(\mathbf{A})$, a matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times \ell}$ and $\sigma > \|\widetilde{\mathbf{T}}_\mathbf{A}\| \cdot \omega(\sqrt{\log m})$. Then there exists* PPT *algorithm* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}, \mathbf{U}, \sigma)$ *that outputs a matrix $\mathbf{X} \in \mathbb{Z}_q^{2m \times \ell}$, distributed statistically close to $D_{\Lambda_q^{\vec{u}}(\mathbf{A}|\mathbf{B}),\sigma}$.*

Theorem 5.3 (SampleRight [ABB10, CHKP10]). *Let $q > 2$, full rank $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with $m > n$, matrices $\mathbf{S} \in \mathbb{Z}_q^{m \times m}$, $\mathbf{U} \in \mathbb{Z}_q^{n \times \ell}$, $y \neq 0 \in \mathbb{Z}_q$ and $\sigma = \sqrt{5} \cdot (1 + \|\mathbf{S}\|_2) \cdot \omega(\sqrt{\log m})$. Then there exists* PPT *algorithm* $\mathsf{SampleRight}(\mathbf{A}, \mathbf{S}, y, \mathbf{U}, \sigma)$ *that outputs a matrix $\mathbf{X} \in \mathbb{Z}_q^{2m \times \ell}$, distributed statistically close to $D_{\Lambda_q^{\vec{u}}(\mathbf{A}|\mathbf{AS}+y\mathbf{G}),\sigma}$, where $\mathbf{G}$ is the gadget matrix from Lemma 5.1.*

Lemma 5.4 (Bounding the Norm of a $\{\pm 1\}^{k \times m}$ Matrix [ABB10]). *Let R be a matrix chosen uniformly at random from $\{\pm 1\}^{k \times m}$. There exists a universal constant C', for which:*

$$\Pr[\|\mathbf{R}\| \geq C'\sqrt{k+m}] < \tfrac{1}{e^{k+m}}.$$

Lemma 5.5 ([Cai98] Gram-Schmidt Minimum). *For any arbitrary integer lattice Λ, it holds that:*

$$1 \leq \min_{\mathbf{B}} \|\widetilde{\mathbf{B}}\| \leq \lambda_1(\Lambda^*) \cdot O(n),$$

with the minimum is over all (ordered) bases $\mathbf{B}$ of lattice Λ.

Lemma 5.6 ([ABB10]). *Let q prime and $m > (n+1)\log q + \omega(\log n)$. Let $\mathbf{S}$ be chosen uniformly from $\{\pm 1\}^{m \times k} \bmod q$, with k polynomial in n. Let $\mathbf{A}$, $\mathbf{B}$ be chosen uniformly from $\mathbb{Z}_q^{n \times m}$ and $\mathbb{Z}_q^{n \times k}$. Then, for every $\vec{e} \in \mathbb{Z}_q^m$, we have that the distributions $(\mathbf{A}, \mathbf{AS}, \mathbf{S}^\top \vec{e})$ and $(\mathbf{A}, \mathbf{B}, \mathbf{S}^\top \vec{e})$ are statistically close.*

To encode identities, we need to recall the following definition from [ABB10]:

Definition 5.7 ([ABB10] Encoding Identities as Matrices). *Let q be a prime and n a positive integer. We say that a function $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n \times n}$ is an encoding with full-rank differences when:*

- *for all distinct $u, v \in \mathbb{Z}_q^n$, the matrix $H(u) - H(v) \in \mathbb{Z}_q^{n \times n}$ has full-rank*
- *H must also be computable in polynomial time in $n \log q$.*

A scheme for encodings with full-rank differences has been introduced by [ABB10].

Definition 5.8 (Identity-Based Encryption). *An Identity-Based Encryption scheme is a tuple of four algorithms* $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$. *The first algorithm,* Setup *produces some public parameters* pk *and a master secret key* mk. $\mathsf{Extract}(\mathsf{mk}, \mathsf{id})$ *generates private keys* sk_id*, associated to any given identity* id. Encrypt *produces encryptions under a target identity* id *using the public parameters* pk*, and* Decrypt *will take as input a key* sk_id *and a ciphertext* ct *of some message x - it will recover the initial message x if* $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, \mathsf{id}, x)$.

Definition 5.9 (Indistinguishability from Random). *This security notion for* IBE *implies both indistinguishability and receiver anonymity (meaning that for every* id*, without* $\mathsf{sk}_{\mathsf{id}}$*, one is not able to determine whether any ciphertext is encrypted under* id*). Moreover, any ciphertext should be indistinguishable from a uniformly random element from the ciphertext space. More formally, consider the following security game for any stateful* PPT *adversary* $\mathcal{A}$*, and every security parameter* λ*, and* $\beta \in \{0,1\}$*:*

Experiment $\mathbf{INDr\text{-}sID\text{-}CPA}_\beta(1^\lambda, \mathcal{A})$*:*

$\mathsf{id}^* \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$x^* \leftarrow \mathcal{A}^{\mathsf{Extract}(\cdot)}(\mathsf{pk})$
$\mathsf{ct}^\star \leftarrow_{\mathrm{R}} \mathcal{C}$
if $\beta = 0$ then $\mathsf{ct}^\star \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{id}^*, x^*)$
$\beta' \leftarrow \mathcal{A}^{\mathsf{Extract}(\cdot)}(\mathsf{pk}, \mathsf{ct}^\star)$
Output: β'

where $\mathcal{C}$ *is the ciphertext space.* $\mathsf{Extract}(\cdot)$ *is an oracle that on input* id*, outputs* $\mathsf{Extract}(\mathsf{mk}, \mathsf{id})$*, only if* $\mathsf{id} \neq \mathsf{id}^*$.

An identity encryption scheme is INDr-sID-CPA*-secure if for every PPT adversary* $\mathcal{A}$*, the following advantage is a negligible function of* λ*:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{INDr\text{-}sID\text{-}CPA}}(\lambda) = \big|\Pr\left[\mathbf{INDr\text{-}sID\text{-}CPA}_0(1^\lambda, \mathcal{A}) = 1\right] - \Pr\left[\mathbf{INDr\text{-}sID\text{-}CPA}_1(1^\lambda, \mathcal{A}) = 1\right]\big|$$

5.2 Our Construction

In this section, we combine the construction idea showcased in Section 4 with the standard-model IBE of [ABB10]. Similarly to the random-oracle model, we prove the resulting scheme anonymous, a property inherited by the underlying IBE scheme [ABB10] we build on. What we obtain is a scheme which is selectively secure with respect to the identities $\mathsf{id}_0^*, \mathsf{id}_1^*$ used for the challenges $\vec{x}_0$ and $\vec{x}_1$. Since the scheme will make use once again of ALS as a building block, the security will be adaptive with respect to the challenge vectors $\vec{x}_0, \vec{x}_1$ themselves. The construction is described in Fig. 12. From our choice of q, this error is small enough to allow correct decryption.

Parameters Recall that plaintext vectors $\vec{x} \in \mathcal{X} = \{0, \ldots, P-1\}$, decryption key vectors $\vec{y} \in \mathcal{Y} = \{0, \ldots, V-1\}$ and when identities match, we recover $\langle \vec{x}, \vec{y} \rangle \in \{0, \ldots, K-1\}$ with $K = \ell PV$. A first-time reader may want to skip this paragraph, as it references steps of the proof. Combining the ALS requirements from Appendix A with the parameters from [ABB10]. our scheme parameters must satisfy the following:

- We start with ALS parameters n_{ALS} (the security parameter), q_{ALS}, σ_{ALS}, ρ_{ALS}, $m_{\mathrm{ALS}} = 2n_{\mathrm{ALS}} \log q$, $n = n_{\mathrm{ALS}}$, α_{ALS}, chosen as in Appendix A.
- From the ALS reduction in Game_5, $\sigma = \sigma_{\mathrm{ALS}}$, $q = q_{\mathrm{ALS}}$, $n = n_{\mathrm{ALS}}$. $m = m_{\mathrm{ALS}}$, $\rho = \rho_{\mathrm{ALS}}$.
- $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$, we require that $\rho > m \cdot \omega(\sqrt{\log m})$.
- $\mathsf{TrapGen}$ requires that $m > 6n \log q$.
- Leftover hash lemma, this carries over from ALS parameters.
- Guessing $\mathbf{T_A}$ step: $\rho > n \cdot \omega(\sqrt{n})$. Notice that increasing ρ_{ALS} can be done without invalidating LWE security. We only need to increase q_{ALS} to satisfy correctness.
- Hardness of LWE: $\alpha q > 2\sqrt{n}$.
- $\mathsf{NoiseGen}$: The spectral norm of $\mathbf{S}^*$ can be upper-bounded (by using the Frobenius norm) by $s_1(\mathbf{S}^*) \leq m$. Using Lemma 4.1, $s_1(\mathbf{Z}_2) \leq C \cdot \rho(2\sqrt{n} + \sqrt{m})$, which implies that $\tau \geq C \cdot m\rho(2\sqrt{n} + \sqrt{m})$.
- Correctness: $q > 2K\ell V(\sigma + \tau) + C'\sigma\rho 4m\sqrt{\ell nm}V$.

  Therefore, we choose ALS parameters, and then modify the following to satisfy our additional constraints:
  - $m \geq 6n \log q$.
  - q is now $q > 2K\ell V(\sigma + \tau) + C'\sigma\rho 4m\sqrt{\ell nm}V$.

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y})}$:
$(\mathbf{A}, \mathbf{T_A}) \leftarrow_R \mathsf{TrapGen}(1^n, 1^m)$
$\mathbf{B}, \mathbf{D} \leftarrow_R \mathbb{Z}_q^{n \times m}$
$\mathsf{mpk} \leftarrow (\mathbf{A}, \mathbf{B}, \mathbf{D})$
$\mathsf{msk} \leftarrow \mathbf{T_A}$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{KeyGen}(\mathsf{id}, \vec{y})}$:
$\mathbf{H}_{\mathsf{id}} \leftarrow (\mathbf{A}|\mathbf{B} + H(\mathsf{id}) \cdot \mathbf{G})$
$\mathbf{R}_{\mathsf{id}} \leftarrow_R \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{B} + H(\mathsf{id}) \cdot \mathbf{G}, \mathbf{D}, \rho)$
Return $(\vec{y}, \mathsf{sk}_{\mathsf{id},\vec{y}} := (\mathbf{R}_{id} \cdot \vec{y}))$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \vec{x})}$:
$\mathbf{H}_{\mathsf{id}} \leftarrow (\mathbf{A}|\mathbf{B} + H(\mathsf{id}) \cdot \mathbf{G}) \in \mathbb{Z}_q^{n \times 2m}$
$\vec{s} \leftarrow_R \mathbb{Z}_q^n$
$\mathbf{S} \leftarrow_R \{\pm 1\}^{m \times m}$
$\vec{e}_1 \leftarrow_R \mathcal{D}_{\mathbb{Z}^{2m},\sigma}$
$\vec{e}_2 \leftarrow_R \mathcal{D}_{\mathbb{Z}^n,\sigma}$
$\vec{e}_3 \leftarrow_R \mathcal{D}_{\mathbb{Z}^n,\tau}$
$\vec{f} \leftarrow (\mathbf{I}_m|\mathbf{S})^\top \cdot \vec{e}_1$
$\mathsf{ct}_1 \leftarrow \mathbf{H}_{\mathsf{id}}^\top \vec{s} + \vec{f}$
$\mathsf{ct}_2 = \mathbf{D}^\top \vec{s} + \vec{e}_2 + \vec{e}_3 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x}$
Return $(\mathsf{ct}_1, \mathsf{ct}_2)$

$\underline{\mathsf{Dec}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{sk}_{\mathsf{id},\vec{y}}, \vec{y})}$:
$\mu = \vec{y}^\top \cdot \mathsf{ct}_2 - \mathsf{sk}_{\mathsf{id},\vec{y}}^\top \cdot \mathsf{ct}_1$
$\mu' = \arg\min_{\mu' \in \{0 \ldots K+1\}} \left| \left\lfloor \frac{q}{K} \right\rfloor \cdot \mu - \mu' \right|$
Return μ'

Fig. 12. An identity-based inner-product functional encryption scheme $\mathcal{IFE}$. Function H is an encoding with full-rank difference, as in Definition 5.7. Algorithms $\mathsf{TrapGen}$ and $\mathsf{SampleLeft}$ are described in Lemma 5.1 and Theorem 5.2. Noise $\vec{e}_3$ is non-smudging and is of a different standard deviation τ, this is needed for the security proof.

- $\tau \geq C \cdot m\rho(2\sqrt{n} + \sqrt{m})$.

Lemma 5.10 (Decryption correctness). *For parameters $n, m, q, \sigma, \rho, \tau$, chosen as in the previous paragraph, the scheme from Fig. 11 is correct.*

Proof. $\mu = \vec{y}^\top \cdot \mathsf{ct}_2 - \mathsf{sk}_{\mathsf{id},\vec{y}}^\top \cdot \mathsf{ct}_1 = \vec{y}^\top \mathbf{D}^\top \vec{s} + \vec{y}^\top \vec{e}_2 + \vec{y}^\top \vec{e}_3 + \left\lfloor \frac{q}{K} \right\rfloor \langle \vec{x}, \vec{y} \rangle - \vec{y}^\top \mathbf{R}_{\mathsf{id}}^\top \mathbf{H}_{\mathsf{id}}^\top \vec{s} - \vec{y}^\top \mathbf{R}_{\mathsf{id}}^\top \vec{f}$. We know that $\mathbf{H}_{\mathsf{id}} \mathbf{R}_{\mathsf{id}} = \mathbf{D}$, therefore the expression above can be rewritten as:

$$\mu = \left\lfloor \frac{q}{K} \right\rfloor \langle \vec{x}, \vec{y} \rangle + \underbrace{\vec{y}^\top \vec{e}_2 + \vec{y}^\top \vec{e}_3 - \vec{y}^\top \mathbf{R}_{\mathsf{id}}^\top \vec{f}}_{\text{error terms}}$$

By Lemma 4.2, we now bound the error terms, by rewritting them as: $\vec{y}^\top \vec{e}_2 + \vec{y}^\top \vec{e}_3 - \vec{y}^\top \mathbf{R}_{\mathsf{id}}^\top \vec{f} = \vec{y}^\top \vec{e}_2 + \vec{y}^\top \vec{e}_3 - \vec{y}^\top \mathbf{R}_{\mathsf{id}}^\top (\mathbf{I}_m|\mathbf{S})^\top \vec{e}_1$. Since $\mathbf{R}_{\mathsf{id}} \in \mathbb{Z}_q^{2m \times n}$, we know that $\|\mathbf{R}_{\mathsf{id}}\| \leq \rho\sqrt{2mn}$. Also $\|\mathbf{S}\| \leq C\sqrt{2m}$, $\|\vec{e}_1\| \leq \sigma\sqrt{2m}$, $\|\vec{e}_2\| \leq \sigma\sqrt{\ell}$ and $\|\vec{e}_3\| \leq \tau\sqrt{\ell}$. Now, $\|\vec{y}\| \leq \sqrt{\ell}V$, therefore $\|\vec{y}^\top \vec{e}_2\| \leq \sigma\ell V$, $\|\vec{y}^\top \vec{e}_3\| \leq \tau\ell V$ and $\|\vec{y}^\top \mathbf{R}_{\mathsf{id}}^\top (\mathbf{I}_m|\mathbf{S})^\top \vec{e}_1\| \leq C'\sigma\rho 4m\sqrt{\ell nm}V$, where the last inequality is derived from Lemma 5.4. Therefore, the final error term is upper bounded by $\ell V(\sigma + \tau) + C'\sigma\rho 4m\sqrt{\ell nm}V$.

Remark 5.11 (Inheriting Anonymity). Towards proving security, we recall the observation made during the proof of our random-oracle construction. Assume that $\mathsf{id}_0^* \neq \mathsf{id}_1^*$ - if the adversary was allowed to obtain secret keys for either $\mathsf{id}_0^* \| \vec{y}$ or $\mathsf{id}_1^* \| \vec{y}$, for any $\vec{y}$, then it could trivially distinguish between encryptions of $\vec{x}_0$ under id_0^* and encryptions of $\vec{x}_1$ under id_1^*. This type of trivial attack is excluded by the SEL-IND definition, therefore we conclude that the adversary cannot obtain decryption key queries for neither id_0^* or id_1^*. Then, there are no functional keys and the adversary interacts with what is essentially an identity-based encryption scheme (in our case, exactly the scheme of [ABB10]). Therefore, since [ABB10] satisfies INDr-sID-CPA (see Definition 5.9), our scheme will trivially also satisfy the same security notion.

We can now state the following security theorem:

Theorem 5.12 (SEL-IND Security). *Let n be the security parameter and consider the ALS functional encryption scheme (Appendix A) with parameters q, σ, ρ, m, n, α. As long as ALS is secure under the $\mathsf{LWE}_{q,\alpha,n}$ and the ALS parameters satisfy in addition $m \geq 6n \log q$, $q > 2K\ell V(\sigma + \tau) + C'\sigma\rho 4m\sqrt{\ell nm}V$, the scheme in Fig. 12 with $\tau \geq C \cdot m\rho(2\sqrt{n} + \sqrt{m})$ is* SEL-IND *secure under the $\mathsf{LWE}_{q,\alpha,n}$ assumption.*

Proof. Since the case of $\mathsf{id}_0^* \neq \mathsf{id}_1^*$ follows from Remark 5.11, we focus on the more difficult case when $\mathsf{id}^* = \mathsf{id}_0^* = \mathsf{id}_1^*$. The proof follows the outline of the original [ABB10]. The novelty is the adaptation for supporting functional decryption keys. An additional step we did not encounter in the random oracle construction is that here, when matrix $\mathbf{A}$ is switched from its trapdoored version to uniformly random, the simulator still needs to have access to a short basis in order to answer key decryption queries for the case id^*. This is a difference from the proof of [ABB10], where they do not need to answer any queries for the punctured identity id^*. Since our proof is selective with respect to the identity of the challenges, we assume that we have access to id^*. We also make use of noise rerandomization and correction techniques from [KY16], which is why we require an additional non-smudging noise $\vec{e}_3$.

Game 0 This is the original SEL-IND game, the challenge plaintexts are $(\vec{x}_0, \vec{x}_1)$.

Game 1 Instead of choosing $(\mathbf{A}, \mathbf{T}) \leftarrow_{\mathrm{R}} \mathsf{TrapGen}(1^n, 1^m)$, choose $\mathbf{A}$ to be uniformly random. To compute decryption keys, we can enumerate all short possible basis $\mathbf{T}_{\mathbf{A}}$ of $\Lambda_q^{\perp}(\mathbf{A})$ and use one of these short basis to generate decryption keys. This game is inefficient and indistinguishable from the previous one by the properties of $\mathsf{TrapGen}()$ (see Lemma 5.1). Since the indistinguishability is derived from a statistical argument, it is not problematic that this game is inefficient.

However, we need to ensure that the lattice spanned by $\mathbf{A}$ actually has a short basis. For this, we use Lemma 5.5, which says that: $\min_{\mathbf{B}} \|\widetilde{\mathbf{B}}\| \leq O(n)$, which means that in order to use the $\mathsf{SampleD}$ algorithm, we need to set our standard deviation $\rho > n \cdot \omega(\sqrt{n})$. This will require us to increase the standard deviation ρ used in $\mathsf{SampleLeft}$, which in turn will lead to a larger modulus than in our random oracle scheme.

Game 2 In this game, we switch the computation of $\mathbf{D}$ to be programmed to $\mathbf{D} = \mathbf{H}_{\mathsf{id}^*}\mathbf{Z}$, where $\mathbf{Z} \leftarrow_{\mathrm{R}} D_{\mathbb{Z}^{2m\times n},\rho}$, using $\mathsf{SampleD}$. We now justify that this change is (statistically) indistinguishable for the adversary. Recall that $\mathbf{H}_{\mathsf{id}^*} = (\mathbf{A}|\mathbf{B} + H(\mathsf{id}^*)\mathbf{G})$. Since $\mathbf{B}$ and $\mathbf{A}$ are uniformly random, we can apply Lemma 4.6 and argue that $\mathbf{H}_{\mathsf{id}^*}\mathbf{Z} = (\mathbf{A}|\mathbf{B} + H(\mathsf{id}^*)\mathbf{G})\mathbf{Z}$ is close to uniformly random over $\mathbb{Z}^{n\times\ell}$ (which is how $\mathbf{D}$ was generated in Game_1).

The role of $\mathbf{R}_{\mathsf{id}^*}$ is played by $\mathbf{Z}$. Use $\mathbf{Z}$ to answer decryption key queries for $\mathsf{id} = \mathsf{id}^*$, as $\mathsf{sk}_{\mathsf{id}^*\|\vec{y}} = \mathbf{Z} \cdot y$. For all other decryption key queries, we still compute $\mathbf{R}_{\mathsf{id}} \leftarrow_{\mathrm{R}} \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathbf{B} + H(\mathsf{id})\mathbf{G}, \mathbf{D})$ and output $\mathbf{R}_{\mathsf{id}^*}\vec{y}$

Game 3 Matrix $\mathbf{T}_{\mathbf{A}}$ must still be found by enumeration. The matrix $\mathbf{S}^*$ which will be chosen for the challenge ciphertext is now picked at ley generation, just as in [ABB10]. $\mathbf{B}$ is now chosen as $\mathbf{B} = \mathbf{A}\mathbf{S}^* - H(\mathsf{id}^*)\mathbf{G}$. $\vec{f} := (\mathbf{I}_m|\mathbf{S})^{\top}\vec{e}_1$. By Lemma 5.6, we know that if $\mathbf{A}$ is uniform over $\mathbb{Z}_q^{n\times m}$ and $\mathbf{S}$ uniform over $\{\pm 1\}^{m\times m}$, the distribution $(\mathbf{A}, \mathbf{B}, (\mathbf{S}^*)^{\top}\vec{e}_1)$ is statistically close to $(\mathbf{A}, \mathbf{A}\mathbf{S}^*, (\mathbf{S}^*)^{\top}\vec{e}_1)$, where $\mathbf{B}$ is uniform over $\mathbb{Z}_q^{n\times m}$. Note that the distributions are close for any choice of $\vec{e}$, so $\vec{e}$ can be known to a distinguishing adversary in this step.

Game 4 This game is now efficient. Use $\mathsf{SampleRight}(\mathbf{A}, \mathbf{S}, H(\mathsf{id}) - H(\mathsf{id}^*), \mathbf{D}, \rho)$ with standard deviation ρ to generate $\mathbf{R}_{\mathsf{id}}$, for $\mathsf{id} \neq \mathsf{id}^*$. This is possible due to the identity encoding function (Definition 5.7), which ensures that $H(\mathsf{id}) - H(\mathsf{id}^*)$ is non-zero, one of the requirements of the $\mathsf{SampleRight}$ algorithm. For $\mathsf{id} = \mathsf{id}^*$, the decryption keys are generated using the secret matrix $\mathbf{Z}$. This game is indistinguishable from the previous game due to Theorem 5.2, which says that the computed $\mathbf{Z}_{\mathsf{id}}$ is statistically close to $D_{\Lambda_q^{\mathbf{U}}(\mathbf{A}|\mathbf{A}\mathbf{S}+(H(\mathsf{id})-H(\mathsf{id}^*)\mathbf{G}),\sigma}$. This holds as long as the standard deviation ρ satisfies the constraints of Theorem 5.3.

Game 5 In this final game, we rely on the security of ALS to argue indistinguishability of ciphertexts. We interact with the AD-CPA challenger for ALS. We receive as public keys $\mathbf{A}_{\mathrm{ALS}}$ and $\mathbf{D}_{\mathrm{ALS}}$. We simulate the view of the adversary in the following manner:

- **Setup:** set $\mathbf{A} := \mathbf{A}_{\mathrm{ALS}}^{\top}$, pick $\mathbf{Z}_2 \leftarrow_{\mathrm{R}} \mathcal{D}_{\mathbb{Z}^{n\times m},\sigma}$, $\mathbf{S}^* \leftarrow_{\mathrm{R}} \{\pm 1\}^{m\times m}$ and set $\mathbf{D} = \mathbf{D}_{\mathrm{ALS}} + \mathbf{A}\mathbf{S}^*\mathbf{Z}_2$.
- **Decryption keys for $\mathsf{id}^*\|\tilde{\mathbf{y}}$:** Ask the ALS functional key oracle for key $\vec{y}$, obtain $\mathsf{sk}_{\vec{y}}$, return $\mathsf{sk}_{\mathsf{id}^*\|\vec{y}} = \left(\dfrac{\mathsf{sk}_{\vec{y}}}{\mathbf{Z}_2\vec{y}}\right)$, which we can compute since we know $\mathbf{Z}_2$.

– **Decryption keys for** $\mathsf{id} \neq \mathsf{id}^*$**:** Using SampleRight as in previous games.
– **Challenge ciphertext** upon receiving $\vec{x}_0, \vec{x}_1$ from the adversary, we forward them to the ALS challenger and receive $(\mathsf{ct}_1^{\text{ALS}}, \mathsf{ct}_2^{\text{ALS}})$. Then, compute and return:

$$\mathsf{ct}_1 = \mathsf{ct}_1^{\text{ALS}} + (\mathbf{S}^*)^\top \mathsf{ct}_1^{\text{ALS}}$$

$$\mathsf{ct}_2 = \mathsf{ct}_2^{\text{ALS}} + \mathbf{Z}_2^\top (\mathbf{S}^*)^\top \mathsf{ct}_1^{\text{ALS}} + \mathsf{NoiseGen}(\mathbf{Z}_2^\top (\mathbf{S}^*)^\top, s)$$

In this game the advantage of the adversary is upper bounded by the advantage of breaking the ALS scheme. It remains to show that Game_5 is indistinguishable from Game_4. We start by spliting $\mathbf{R}_{\mathsf{id}^*} := \begin{pmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \end{pmatrix}$. Then the fact that $(\mathbf{A} \| \mathbf{A}\mathbf{S}^*)\mathbf{R}_{\mathsf{id}^*} = \mathbf{D}$ translates to the fact that $\mathbf{A}\mathbf{Z}_1 + \mathbf{A}\mathbf{S}^*\mathbf{Z}_2 = \mathbf{D}$. The second matrix $\mathbf{Z}_2$ we can simply sample from $\mathcal{D}_{\mathbb{Z}^{n \times m}, \rho, c}$, while $\mathbf{S}^*$ is known and $\mathbf{D}$ is programmable (since Game_2). Now we set $\mathbf{A}^\top = \mathbf{A}_{\text{ALS}}$ and implicitly $\mathbf{Z}_1^\top \mathbf{A}^\top = \mathbf{D}_{\text{ALS}}$ (Recall that $\mathbf{Z}_1$ is the master secret key of ALS, so we do not have access to it). Notice that:

$$\mathsf{ct}_1 = (\mathbf{I}_m | \mathbf{S}^*)^\top (\mathbf{A}^\top \vec{s} + \vec{e}_1) = (\mathbf{A} | \mathbf{A}\mathbf{S}^*)^\top \vec{s} + (\mathbf{I}_m | \mathbf{S}^*)^\top \vec{e}_1$$

$$\mathsf{ct}_2 = \mathbf{D}^\top \vec{s} + \left\lfloor \tfrac{q}{K} \right\rfloor \cdot \vec{x} + \vec{e}_2 = \mathbf{Z}_1^\top \mathbf{A}^\top \vec{s} + \mathbf{Z}_2^\top (\mathbf{S}^*)^\top \mathbf{A}^\top \vec{s} + \left\lfloor \tfrac{q}{K} \right\rfloor \cdot \vec{x}^\beta$$

,

Let e_1 be the noise of the first ALS ciphertext. Then we can rewrite to:

$$\mathsf{ct}_1 = \mathsf{ct}_1^{\text{ALS}} + (\mathbf{S}^*)^\top \mathsf{ct}_1^{\text{ALS}}$$

$$\mathsf{ct}_2 = \mathbf{D}^\top \vec{s} + \left\lfloor \tfrac{q}{K} \right\rfloor \cdot \vec{x} + \vec{e}_2 + \vec{e}_3 = \mathsf{ct}_2^{\text{ALS}} + \mathbf{Z}_2^\top (\mathbf{S}^*)^\top \mathsf{ct}_1^{\text{ALS}} + \mathsf{NoiseGen}(\mathbf{Z}_2^\top (\mathbf{S}^*)^\top, s)$$

As already explained in our parameter section, we can upper-bound $s_1(\mathbf{Z}_2^\top (\mathbf{S}^*))$ and we let $s > s_1(\mathbf{Z}_2^\top (\mathbf{S}^*))$. To conclude, by the properties of $\mathsf{NoiseGen}(\mathbf{Z}_2^\top (\mathbf{S}^*)^\top, s)$, we know that the noise in ct_2 will be statistically close to $\vec{e}_2 + \vec{e}_3$, therefore Game_5 is statistically close to Game_4.

6 Multi-Input Inner-Product Functional Encryption with Rich Access Control

In this section, we build a multi-input FE scheme for the family of functions $\mathcal{F}^{\mathsf{multi}}_{\mathsf{ipfe}(d,B),\mathsf{P}_1,\ldots,\mathsf{P}_n}$ for a dimension $d \in \mathbb{N}$, a bound $B \in \mathbb{N}$, and n predicates $\mathsf{P}_1, \ldots, \mathsf{P}_n \in \mathcal{P}$, where each predicate takes as input an attribute in the universe $\mathcal{U}$, and returns a bit. Each function is described by a set of indices $\mathcal{S} \subseteq [n]$, and for each index $\in \mathcal{S}$, an associated pair $(\mathsf{att}_i \in \mathcal{U}, \vec{y}_i \in [0, B]^d)$. It takes as input n vectors $\vec{x}_1, \ldots, \vec{x}_n \in [0, B]^d$, and outputs $\sum_{i \in \mathcal{S}} \vec{x}_i^\top \vec{y}_i \in [0, ndB^2]$ if $\mathsf{P}_i(\mathsf{att}_i) = 1$ for all $i \in \mathcal{S}$.

Note that to each input slot corresponds to a different predicate P_i, which represents a different policy access to the encrypted data. For instance, some medical data can be considered very sensitive, and should only be computed on by doctors, whereas other records might be slightly less sensitive, and could pertain to the output of a computation performed by a user with lower credentials, say, a health insurance company. Each functional secret key permits to compute a weighted sum on some part of the data for which it has credentials (this is represented by the set S). That is, a key decrypts successfully provided it is associated with attributes that satisfy the predicates associated with the encrypted data on which it computes.

Our construction generically transforms any single-input FE for the family of functions $\mathcal{F}_{\mathsf{ipfe}(d+1,B),\mathcal{P}}$ (described in Section 3) satisfying some structural properties into an MIFE for the family of functions $\mathcal{F}^{\mathsf{multi}}_{\mathsf{ipfe}(d,B),\mathsf{P}_1,\ldots,\mathsf{P}_n}$. We first recall the definition of multi-input FE and then describe our construction.

6.1 Definitions

Definition 6.1 (Multi-Input Functional Encryption [GGG+14]). *Let $\mathcal{F}$ be a family of n-ary functions, with $f \in \mathcal{F}$ defined as $f : \mathcal{X}^n \to \mathcal{Y}$. A functional encryption scheme for $\mathcal{F}$ consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda, \mathcal{F})$*: takes as input the security parameter λ and a description of the function family $\mathcal{F}$, and outputs a master public key* mpk *(which is implicitly part of the inputs of all other algorithms), a master secret key* msk*, and n encryption keys* $\mathsf{ek}_1, \ldots, \mathsf{ek}_n$*.*
- $\mathsf{Enc}(\mathsf{ek}_i, x_i)$*: takes as input an encryption key* ek_i*, a message $x_i \in \mathcal{X}$, and outputs a ciphertext* ct_i*.*
- $\mathsf{KeyGen}(\mathsf{msk}, f)$*: takes as input the master secret key* msk*, a function $f \in \mathcal{F}$. It outputs a functional decryption key* sk_f*.*
- $\mathsf{Dec}\Big(\mathsf{sk}_f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_n)\Big)$*: takes as input the functional decryption key* sk_f *along with ciphertexts* $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$*. It outputs a value $y \in \mathcal{Y}$ or the special symbol $\perp$ if it fails.*

A scheme as defined above is correct if for all security parameter λ, $f \in \mathcal{F}$, $x_1, \ldots, x_n \in \mathcal{X}$, we have: $\Pr\left[\mathsf{Dec}\Big(\mathsf{sk}_f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_n)\Big) = f(x_1, \ldots, x_n)\right] = 1$ *where the probability is taken over* $(\mathsf{mpk}, \mathsf{msk}, (\mathsf{ek}_1, \ldots, \mathsf{ek}_n)) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i)$ *for all $i \in [n]$,* $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$*.*

Definition 6.2 (AD-IND security). *For every functional encryption $\mathcal{MIFE}$, every security parameter λ, every stateful adversary $\mathcal{A}$, we define the following experiments for $\beta \in \{0, 1\}$:*

Experiment $\mathbf{AD\text{-}IND}_\beta^{\mathcal{FE}}(1^\lambda, \mathcal{A})$*:*

$(\mathsf{mpk}, \mathsf{msk}, (\mathsf{ek}_1, \ldots, \mathsf{ek}_n)) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F})$
$\beta' \leftarrow \mathcal{A}^{\mathsf{OCorrupt}(\cdot), \mathsf{OKeyGen}(\cdot), \mathsf{OEnc}(\cdot,\cdot,\cdot)}(\mathsf{mpk})$
Output: β'

where n is the number of users; on input $i \in [n]$, the oracle $\mathsf{OCorrupt}(i)$ *returns* ek_i *and adds i to the set* Corr *of corrupted users; on input tuples of the form (i, x^0, x^1) where $i \in [n]$, $x^0, x^1 \in \mathcal{X}$, the oracle* $\mathsf{OEnc}(i, x^0, x^1)$ *returns* $\mathsf{Enc}(\mathsf{ek}_i, x^\beta)$ *and adds (i, x^0, x^1) to the list of queries, denoted by $\mathcal{Q}$; on input a function $f \in \mathcal{F}$ of arity n, the oracle* $\mathsf{OKeyGen}(f)$ *returns the functional decryption key* $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$*.*

We denote by $\mathcal{I}$ the set of pairs $\Big((x_1^0, \ldots, x_\ell^0), (x_1^1, \ldots, x_\ell^1)\Big)$ *where for all $i \in \mathsf{Hon}$, $(i, x_i^0, x_i^1) \in \mathcal{Q}$ and for all $i \in \mathsf{Corr}$, $x_i^0 = x_i^1 \in \mathcal{X}$. We require that for all functions f queried to* $\mathsf{OKeyGen}$*, we have: $f(x_1^0, \ldots, x_\ell^0) = f(x_1^1, \ldots, x_\ell^1)$ for all pairs* $\Big((x_1^0, \ldots, x_\ell^0), (x_1^1, \ldots, x_\ell^1)\Big) \in \mathcal{I}$*.*

Moreover, we require that for all users i that are corrupted during the game, all the queries of the form $(i, x^0, x^1) \in \mathcal{Q}$ are such that $x^0 = x^1$.

Using generic transformations from [ABKW19, Gay19], which build upon [AGRW17, DOT18], we can assume without loss of generality that, if $i \in \mathsf{Hon}$, then there exists a query $(i, x_i^0, x_i^1) \in \mathcal{Q}$.

A multi-input functional encryption scheme $\mathcal{FE}$ is AD-IND-secure if for every PPT adversary $\mathcal{A}$, the following advantage is a negligible function of λ:

$$\mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{FE},\mathcal{A}}(\lambda) = \left|\Pr\left[\mathbf{AD\text{-}IND}_0^{\mathcal{FE}}(1^\lambda, \mathcal{A}) = 1\right] - \Pr\left[\mathbf{AD\text{-}IND}_1^{\mathcal{FE}}(1^\lambda, \mathcal{A}) = 1\right]\right|$$

We define one-time security with its associated games $\mathbf{ONE\text{-}AD\text{-}IND}_b^{\mathcal{FE}}$ *and advantage* $\mathsf{Adv}^{\mathsf{ONE\text{-}AD\text{-}IND}}_{\mathcal{FE},\mathcal{A}}(\lambda)$ *similarly, except the adversary can make at most one query of the form (i, x_i^0, x_i^1) for each slot $i \in [n]$.*

There are weaker security notions, where the set of corrupted users Corr*, or the set of queries $\mathcal{Q}$ is chosen beforehand by the adversary. The former restriction is referred to as static corruptions, whereas the second restriction is referred to as selective security.*

6.2 Generic Construction

We show that the techniques used in [ACF+18, AGRW17] to generically transform any single-input FE for inner products into a multi-input FE for inner products are compatible with our single-input FE. Thus, we obtain an MIFE for the family of functions $\mathcal{F}^{\mathsf{multi}}_{\mathsf{ipfe}(d,B),\mathsf{P}_1,\ldots,\mathsf{P}_n}$, described in Fig. 13, which relies on any single-input FE $(\mathsf{Setup}', \mathsf{Enc}', \mathsf{KeyGen}', \mathsf{Dec}')$ for the family of functions $\mathcal{F}_{\mathsf{ipfe}(d+1,B),\mathcal{P}}$, such as described in Section 3, satisfying the following properties:

1. **Pairing-based:** ciphertexts contain groups elements in $\mathbb{G}_1$ and functional decryption keys contain groups elements in $\mathbb{G}_2$, for a pairing group $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \mathsf{PGGen}(1^\lambda)$, with $p \gg B$.
2. **Large inputs:** The algorithms Enc' and KeyGen' can take as input arbitrary vectors in $\mathbb{Z}_p^d$ (as opposed to vectors in $[0, B]^d$ for a polynomial bound B) and the decryption returns $[\vec{x}^\top \vec{y}]_T \in \mathbb{G}_T$.
3. **Linear homomorphism:** There is a PPT algorithm Add such that for all $\vec{x}, \vec{x}' \in \mathbb{Z}_p^d$ and $\mathsf{P} \in \mathcal{P}$, the following are identically distributed: $(\mathsf{Enc}'(\mathsf{mpk}, \mathsf{P}, \vec{x}), \mathsf{Enc}'(\mathsf{mpk}, \mathsf{P}, \vec{x}+\vec{x}'))$ and $(\mathsf{Enc}'(\mathsf{mpk}, \mathsf{P}, \vec{x}), \mathsf{Add}(\mathsf{Enc}'(\mathsf{mpk}, \mathsf{P}, \vec{x}), \vec{x}'))$. That is, Add can produce a fresh encryption of $\vec{x} + \vec{x}'$ from an encryption of $\vec{x}$, for a given predicate $\mathsf{P} \in \mathcal{P}$.

It is not hard to verify that our scheme in Section 3.1 satisfies these properties.

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{multi}}_{\mathsf{ipfe}(d,B),\mathsf{P}_1,\ldots,\mathsf{P}_n})}$:
For all $i \in [n]$: $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{Setup}'(1^\lambda, \mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}})$, $\vec{u}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$.
Return $\mathsf{mpk} := \{\mathsf{mpk}_i\}_{i\in[n]}$, $\mathsf{msk} := \{\mathsf{msk}_i, \vec{u}_i\}_{i\in[n]}$, $\{\mathsf{ek}_i := \vec{u}_i\}_{i\in[n]}$.

$\underline{\mathsf{Enc}(\mathsf{ek}_i, \vec{x}_i)}$:
$\mathsf{ct}_i \leftarrow \mathsf{Enc}'(\mathsf{mpk}_i, \mathsf{P}_i, \vec{x}_i + \vec{u}_i \bmod p)$. Return ct_i.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathcal{S}, (\mathsf{att}_j, \vec{y}_j)_{j\in\mathcal{S}})}$:
If there exists $j \in \mathcal{S}$ such that $\mathsf{P}_j(\mathsf{att}_j) = 0$, then return $\perp$. Otherwise, for all $j \in \mathcal{S}$, $\mathsf{sk}'_j \leftarrow \mathsf{KeyGen}'(\mathsf{msk}_j, \mathsf{att}_j, \vec{y}_j)$. Return $\mathsf{sk}_{\mathsf{att},\vec{y}} := \left(\mathcal{S}, \{\mathsf{sk}'_j\}_{j\in\mathcal{S}}, \sum_{j\in\mathcal{S}} \vec{y}_j^\top \vec{u}_j\right)$.

$\underline{\mathsf{Dec}\left(\mathsf{sk}_{\mathsf{att},\vec{y}}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_n)\right)}$:
Parse $\mathsf{sk}_{\mathsf{att},\vec{y}} := (\mathcal{S}, \{\mathsf{sk}'_j\}_{j\in\mathcal{S}}, z)$
For all $j \in \mathcal{S}$, $[d_j]_T \leftarrow \mathsf{Dec}'(\mathsf{sk}'_j, \mathsf{ct}_j)$. $[\mathsf{out}]_T := \sum_{j\in\mathcal{S}} [d_j]_T - [z]_T$. Return out.

Fig. 13. MIFE for the family $\mathcal{F}^{\mathsf{multi}}_{\mathsf{ipfe}(d,B),\mathsf{P}_1,\ldots,\mathsf{P}_n}$, where $\mathsf{P}_1, \ldots, \mathsf{P}_n \in \mathcal{P}$. Here, $\mathcal{FE}' := (\mathsf{Setup}', \mathsf{Enc}', \mathsf{KeyGen}', \mathsf{Dec}')$ is a SEL-IND secure FE for the family $\mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}}$, satisfying the structural properties listed in Section 6.2.

Correctness. Let $\mathcal{S} \subseteq [n]$, $\vec{x}_i \in [0, B]^d$ for $i \in [n]$, $\vec{y}_j \in [0, B]^d$, $\mathsf{att}_j \in \mathcal{U}$ for $j \in \mathcal{S}$, such that for all $j \in \mathcal{S}$, $\mathsf{P}_j(\mathsf{att}_j) = 1$. We have, by correctness of the single-input FE, for all $j \in \mathcal{S}$, $[d_j]_T := [\vec{x}_j^\top \vec{y}_j + \vec{u}_j^\top \vec{y}_j]_T$. Thus, $[\mathsf{out}]_T := [\sum_{j\in\mathcal{S}} \vec{x}_j^\top \vec{y}_j + \vec{u}_j^\top \vec{y}_j]_T - [z]_T = [\sum_{j\in\mathcal{S}} \vec{x}_j^\top \vec{y}_j]_T$, and the decryption can efficiently recover out, since the output belongs to $[0, ndB^2]$, which is polynomially bounded.

Theorem 6.3 (IND security). *The MIFE presented in Fig. 4 $\mathcal{MIFE}$ is AD-IND secure (respectively SEL-IND secure) as long as the underlying single-input FE $\mathcal{FE}'$ is AD-IND secure (respectively SEL-IND secure). Namely, for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that:*

$$\mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{A}}(\lambda) \leq n \cdot \mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{FE}',\mathcal{B}}(\lambda) + \mathsf{negl}(\lambda).$$

Similarly, for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that:

$$\mathsf{Adv}^{\mathsf{SEL\text{-}IND}}_{\mathcal{MIFE},\mathcal{A}}(\lambda) \leq n \cdot \mathsf{Adv}^{\mathsf{SEL\text{-}IND}}_{\mathcal{FE}',\mathcal{B}}(\lambda) + \mathsf{negl}(\lambda).$$

The proof, as in [ACF+18, AGRW17], proceeds in two steps, where we first prove perfect one-time security, then boost it to many-time security using the many-time security of the underlying single-input FE. The adaptive and selective settings have very similar proofs, thus, we only give the proof in the adaptive setting, which is the most technical.

Proof. First, in Lemma 6.4, we show that for any adversary $\mathcal{A}$, we have:

$$\mathsf{Adv}^{\mathsf{ONE\text{-}AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{A}}(\lambda) = 0.$$

Next, for all PPT adversaries $\mathcal{A}$, we show there exist PPT adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that:

$$\mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ONE\text{-}AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{B}_1}(\lambda) + n \cdot \mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{FE}',\mathcal{B}_2}(\lambda) + \mathsf{negl}(\lambda).$$

The proof uses a series of hybrid games, described in Fig. 14. For game G_j for all $j \in [1,3]$, we denote by $\mathsf{Adv}_j(\mathcal{A})$ the probability that game G_j outputs 1 when interacting with $\mathcal{A}$. Here, for any input slot $i \in [n]$, we denote by $(i, \vec{x}_i^{j,0}, \vec{x}_i^{j,1})$ the j-th query for slot $i \in [n]$ (note that there can be many such queries for each slot).

It is clear that games G_1 and G_3 correspond to $\mathbf{AD\text{-}IND}_0^{\mathcal{FE}}(1^\lambda, \mathcal{A})$ and $\mathbf{AD\text{-}IND}_1^{\mathcal{FE}}(1^\lambda, \mathcal{A})$, respectively. We show in Lemma 6.5 that there exists a PPT adversary $\mathcal{B}_1$ such that

$$|\mathsf{Adv}_1(\mathcal{A}) - \mathsf{Adv}_2(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{ONE\text{-}AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{B}_1}(\lambda).$$

We show in Lemma 6.6 that there exists a PPT adversary $\mathcal{B}_2$ such that

$$|\mathsf{Adv}_2(\mathcal{A}) - \mathsf{Adv}_3(\mathcal{A})| \leq n \cdot \mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{FE}',\mathcal{B}_2}(\lambda) + \mathsf{negl}(\lambda).$$

Lemma 6.4. *For any adversary $\mathcal{A}$, we have:*

$$\mathsf{Adv}^{\mathsf{ONE\text{-}AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{A}}(\lambda) = 0.$$

Proof. We first define a game G_0, which samples a random bit $\beta \leftarrow_{\mathrm{R}} \{0,1\}$, and a guess of all the queries $(\vec{w}_i^0, \vec{w}_i^1) \leftarrow_{\mathrm{R}} ([0,B]^d)^2$ for all $i \in [n]$. Then the experiment behaves as $\mathbf{AD\text{-}IND}_\beta^{\mathcal{FE}}(1^\lambda, \mathcal{A})$. At the end of the experiment, the adversary $\mathcal{A}$ outputs a bit β'. If the guess was successful, that is, the queries made by $\mathcal{A}$ were predicted correctly, with $\vec{w}_i^0 = \vec{x}_i^0$ and $\vec{w}_i^1 = \vec{x}_i^1$, then the experiment outputs β'. Otherwise, it outputs a random bit $\beta' \leftarrow_{\mathrm{R}} \{0,1\}$. We define the advantage $\mathsf{Adv}_{\mathsf{G}_0}(\lambda) := 2 \cdot |1/2 - \Pr[\beta' = \beta | \beta' \leftarrow \mathsf{G}_0]|$. We have: $\mathsf{Adv}_{\mathsf{G}_0}(\lambda) = B^{2dn} \cdot \mathsf{Adv}^{\mathsf{ONE\text{-}AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{A}}(\lambda)$.

Now, we show that $\mathsf{Adv}_{\mathsf{G}_0}(\lambda) = 0$, which implies that $\mathsf{Adv}^{\mathsf{ONE\text{-}AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{A}}(\lambda) = 0$. We use the fact that $\{\vec{u}_i\}_{i\in[n]}$, is identically distributed to $\{\vec{u}_i + \vec{w}_i^1 - \vec{w}_i^0\}_{i\in[n]}$, where $\vec{u}_i, \vec{w}_i^0, \vec{w}_i^1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$.

When the guess is successful, that is, when $\vec{w}_i^0 = \vec{x}_i^0$ and $\vec{w}_i^1 = \vec{x}_i^1$ for all $i \in [n]$, the first distribution corresponds to case where $b = 0$ in the experiment G_0, with challenge ciphertexts of the form $\mathsf{Enc}'(\mathsf{mpk}_i, \mathsf{P}_i, \vec{x}_i^0 + \vec{u}_i \bmod p)$ and keys of the form $\left(\{\mathsf{KeyGen}'(\mathsf{msk}_i, \mathsf{att}_i, \vec{y}_i)\}_{i\in[n]}, \sum_{i\in[n]} \vec{u}_i^\top \vec{y}_i\right)$, whereas the second distribution corresponds to the case where $b = 1$ in the experiment G_0, with challenge ciphertexts of the form $\mathsf{Enc}'(\mathsf{mpk}_i, \mathsf{P}_i, \vec{x}_i^0 + (\vec{u}_i + \vec{x}_i^1 - \vec{x}_i^0) \bmod p) = \mathsf{Enc}'(\mathsf{mpk}_i, \mathsf{P}_i, \vec{x}_i^1 + \vec{u}_i \bmod p)$ and keys of the form $(\mathcal{S}, \{\mathsf{KeyGen}'(\mathsf{msk}_i, \mathsf{att}_i, \vec{y}_i)\}_{i\in\mathcal{S}}, \sum_{i\in\mathcal{S}}(\vec{u}_i + \vec{x}_i^1 - \vec{x}_i^0)^\top \vec{y}_i = \sum_{i\in\mathcal{S}} \vec{u}_i^\top \vec{y}_i)$, where the last equality uses the fact that $\sum_{i\in\mathcal{S}} \vec{y}_i^\top \vec{x}_i^0 = \sum_{i\in\mathcal{S}} \vec{y}_i^\top \vec{x}_i^1$, by definition of the security game.

Lemma 6.5 (From Game G_1 to G_2). *There exists a PPT adversary $\mathcal{B}_1$ such that*

$$|\mathsf{Adv}_1(\mathcal{A}) - \mathsf{Adv}_2(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{ONE\text{-}AD\text{-}IND}}_{\mathcal{MIFE},\mathcal{B}_1}(\lambda).$$

Proof. Adversary $\mathcal{B}_1$ receives mpk which it forwards to $\mathcal{A}$. When $\mathcal{A}$ queries $\mathsf{OKeyGen}$ or $\mathsf{OCorrupt}$, $\mathcal{B}$ forwards the query to its own oracle and gives back the answer to $\mathcal{A}$. When $\mathcal{A}$ queries $\mathsf{OEnc}(i, \vec{x}_i^{1,0}, \vec{x}_x^{1,1})$ on slot $i \in [n]$ for the first time, $\mathcal{B}_1$ forwards the query to its own oracle and gives back the answer to $\mathcal{A}$, which is of the form $\mathsf{ct}'_i \leftarrow_{\mathrm{R}} \mathsf{Enc}'(\mathsf{mpk}_i, \mathsf{P}_i, \vec{x}_i^{1,b} + \vec{u}_i \bmod p)$. For any subsequent queries $(i, \vec{x}_i^{j,0}, \vec{x}_x^{j,1})$ to OEnc, that is, for $j > 1$, $\mathcal{B}_1$ uses the linear homomorphism to compute $\mathsf{ct}_i \leftarrow_{\mathrm{R}} \mathsf{Add}(\mathsf{ct}'_i, \vec{x}_i^{j,0} - \vec{x}_i^{1,0})$, which by property 3 is distributed as a fresh encryption of the form $\mathsf{Enc}'(\mathsf{mpk}_i, \mathsf{P}_i, \vec{x}_i^{j,0} + \vec{x}_i^{1,b} - \vec{x}_i^{1,0})$. Note that in the case $b = 0$, this corresponds to the game G_1, whereas it corresponds to the game G_2 when $b = 1$.

Lemma 6.6 (From Game G_2 to G_3). *There exists a PPT adversary $\mathcal{B}_2$ such that*

$$|\mathsf{Adv}_2(\mathcal{A}) - \mathsf{Adv}_3(\mathcal{A})| \leq n \cdot \mathsf{Adv}^{\mathsf{AD\text{-}IND}}_{\mathcal{FE}',\mathcal{B}_2}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. We switch the distribution of the challenge ciphertexts output by OEnc, one slot $i \in [n]$ at a time, using a hybrid argument. To change the ciphertexts for slot $i \in [n]$, we use the security of $\mathcal{FE}'$ on $(\mathsf{mpk}_i, \mathsf{msk}_i)$. Namely, the reduction $\mathcal{B}_2$ samples $(\mathsf{mpk}_\theta, \mathsf{msk}_\theta) \leftarrow_{\mathrm{R}} \mathsf{Setup}'(\mathcal{F}_{\mathsf{ipfe}(d,B),\mathcal{P}})$ for all $\theta \neq i$ and $\vec{u}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$ for $i \in [n]$, and it uses these values to simulate mpk and answer the queries $(\theta, \vec{x}_\theta^{j,0}, \vec{x}_\theta^{j,1})$ for all $\theta \neq i$. $\mathcal{B}_2$ can also answer queries to $\mathsf{OCorrupt}$, since it knows $\mathsf{ek}_i := \vec{u}_i$ for all $i \in [n]$. To answer $\mathcal{A}$'s queries to $\mathsf{OKeyGen}$, it uses its own oracle to compute the sk'_i component of the secret key and its knowledge of msk_θ for all $\theta \neq i$ and $\vec{u}_i$ for $i \in [n]$ to compute the remaining components, and forwards the answer to $\mathcal{A}$. Upon receiving the queries $(i, \vec{x}_i^{j,0}, \vec{x}_i^{j,1})$, $\mathcal{B}_2$ sends the queries $(\vec{x}_i^{j,0} + \vec{x}_i^{1,1} - \vec{x}_i^{1,0} + \vec{u}_i \bmod p, \vec{x}_i^{j,1} + \vec{u}_i \bmod p)$ to its own experiment. The left challenge corresponds to the game G_2, whereas the right challenge corresponds to the game G_3. To use the AD-IND security of $\mathcal{FE}'$, we rely on the fact that

$$(\vec{x}_i^{j,0} - \vec{x}_i^{1,0})^\top \vec{y}_i = (\vec{x}_i^{j,1} - \vec{x}_i^{1,1})^\top \vec{y}_i. \tag{1}$$

This is required by definition of the security game, since this information is given by the ideal functionality (correctness of the scheme). For all $i \in [n]$, any vector $\vec{w}_i \in [-B, 2B]^d$, with probability at least $1 - \frac{3dB}{p}$ over the choice of $\vec{u}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$, we have $(\vec{u}_i + \vec{w}_i) \bmod p = (\vec{u}_i \bmod p) + \vec{w}_i$. Together with (1), this implies that: $(\vec{u}_i + \vec{x}_i^{j,0} + \vec{x}_i^{1,1} - \vec{x}_i^{1,0} \bmod p)^\top \vec{y}_i = (\vec{x}_i^{j,0} + \vec{x}_i^{1,1} - \vec{x}_i^{1,0})^\top \vec{y}_i + (\vec{u}_i \bmod p)^\top \vec{y}_i = \vec{x}_i^{j,1^\top} \vec{y}_i + (\vec{u}_i \bmod p)^\top \vec{y}_i = (\vec{x}_i^{j,1} + \vec{u}_i \bmod p)^\top \vec{y}_i$. That means the queries of $\mathcal{B}_2$ to its experiment are valid.

$\underline{\mathsf{G}_1, \boxed{\mathsf{G}_2}, \boxed{\mathsf{G}_3}:}$
$(\mathsf{mpk}, \mathsf{msk}, (\mathsf{ek}_1, \ldots, \mathsf{ek}_n)) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{multi}}_{\mathsf{ipfe}(d,B),\mathsf{P}_1,\ldots,\mathsf{P}_n})$
$\beta' \leftarrow \mathcal{A}^{\mathsf{OCorrupt}(\cdot),\mathsf{OKeyGen}(\cdot,\cdot),\mathsf{OEnc}(\cdot,\cdot,\cdot)}(\mathsf{mpk})$

$\underline{\mathsf{OCorrupt}(i):}$
Return ek_i.

$\underline{\mathsf{OEnc}(i, \vec{x}_i^{j,0}, \vec{x}_i^{j,1}):}$
$\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \vec{x}_i^{j,0} + \boxed{\vec{x}_i^{1,1} - \vec{x}_i^{1,0}})$ $\boxed{\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \vec{x}_i^{j,1})}$
Return ct_i.

$\underline{\mathsf{OKeyGen}(\mathcal{S}, (\mathsf{att}_j, \vec{y}_j)_{j \in \mathcal{S}}):}$
$\mathsf{sk}_{\mathsf{att},\vec{y}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathcal{S}, (\mathsf{att}_j, \vec{y}_j)_{j \in \mathcal{S}})$.
Return $\mathsf{sk}_{\mathsf{att},\vec{y}}$.

Fig. 14. Hybrid games for the proof of Theorem 6.3.

Acknowledgments. The first author was supported in part by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement 780108 (FENTEC), by the ERC Project aSCEND (H2020 639554), and by the French FUI project ANBLIC. The third author was partially supported by a Google PhD Fellowship in Privacy and Security. The fourth author was partially supported by the ERC Project PREP-CRYPTO (H2020 724307). Part of this work was done while the third author was at École normale supérieure, Paris, France, at UC Berkeley, California, USA, and at Cornell Tech, NY, USA.

References

AARV17. B. Applebaum, B. Arkis, P. Raykov, and P. N. Vasudevan. Conditional disclosure of secrets: Amplification, closure, amortization, lower-bounds, and separations. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 727–757. Springer, Heidelberg, August 2017. (Cited on page 3.)

ABB10. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT 2010*, *LNCS* 6110, pages 553–572. Springer, Heidelberg, May / June 2010. (Cited on pages 4, 29, 30, 31, 32, and 42.)

ABDP15. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *PKC 2015*, *LNCS* 9020, pages 733–751. Springer, Heidelberg, March / April 2015. (Cited on page 1.)

ABDP16. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Better security for functional encryption for inner product evaluations. Cryptology ePrint Archive, Report 2016/011, 2016. http://eprint.iacr.org/2016/011. (Cited on page 1.)

ABG19. M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner-product functional encryption. In *ASIACRYPT 2019, Part III*, *LNCS* 11923, pages 552–582. Springer, Heidelberg, December 2019. (Cited on page 1.)

ABKW19. M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC 2019, Part II*, *LNCS* 11443, pages 128–157. Springer, Heidelberg, April 2019. (Cited on pages 1 and 34.)

ABS17. M. Ambrona, G. Barthe, and B. Schmidt. Generic transformations of predicate encodings: Constructions and applications. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 36–66. Springer, Heidelberg, August 2017. (Cited on page 9.)

AC16. S. Agrawal and M. Chase. A study of pair encodings: Predicate encryption in prime order groups. In *TCC 2016-A, Part II*, *LNCS* 9563, pages 259–288. Springer, Heidelberg, January 2016. (Cited on page 9.)

AC17. S. Agrawal and M. Chase. Simplifying design and analysis of complex predicate encryption schemes. In *EUROCRYPT 2017, Part I*, *LNCS* 10210, pages 627–656. Springer, Heidelberg, April / May 2017. (Cited on page 9.)

ACF$^+$18. M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO 2018, Part I*, *LNCS* 10991, pages 597–627. Springer, Heidelberg, August 2018. (Cited on pages 1, 4, 35, and 36.)

ACGU20. M. Abdalla, D. Catalano, R. Gay, and B. Ursu. Inner-product functional encryption with fine-grained access control. In *ASIACRYPT 2020*, LNCS. Springer, Heidelberg, December 2020. (Cited on page 6.)

AGRW17. M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT 2017, Part I*, *LNCS* 10210, pages 601–626. Springer, Heidelberg, April / May 2017. (Cited on pages 1, 4, 34, 35, and 36.)

AGW20. M. Abdalla, J. Gong, and H. Wee. Functional encryption for attribute-weighted sums from k-Lin. In *CRYPTO 2020, Part I*, *LNCS* 12170, pages 685–716. Springer, Heidelberg, August 2020. (Cited on page 5.)

AJS18. P. Ananth, A. Jain, and A. Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. https://eprint.iacr.org/2018/615. (Cited on page 5.)

Ajt99. M. Ajtai. Generating hard instances of the short basis problem. In *ICALP 99*, *LNCS* 1644, pages 1–9. Springer, Heidelberg, July 1999. (Cited on pages 25 and 29.)

AL10. N. Attrapadung and B. Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In *PKC 2010*, *LNCS* 6056, pages 384–402. Springer, Heidelberg, May 2010. (Cited on page 44.)

ALS16. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *CRYPTO 2016, Part III*, *LNCS* 9816, pages 333–362. Springer, Heidelberg, August 2016. (Cited on pages 1, 3, 4, 8, 12, 26, 27, 42, 43, and 44.)

AP09. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, *LIPIcs* 3, pages 75–86. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009. (Cited on page 29.)

AS17. P. Ananth and A. Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT 2017, Part I*, *LNCS* 10210, pages 152–181. Springer, Heidelberg, April / May 2017. (Cited on page 1.)

Att14. N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *EUROCRYPT 2014*, *LNCS* 8441, pages 557–577. Springer, Heidelberg, May 2014. (Cited on pages 3, 9, and 17.)

Att16. N. Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In *ASIACRYPT 2016, Part II*, *LNCS* 10032, pages 591–623. Springer, Heidelberg, December 2016. (Cited on page 9.)

BB04. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *EUROCRYPT 2004*, *LNCS* 3027, pages 223–238. Springer, Heidelberg, May 2004. (Cited on page 44.)

BBG05. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005*, *LNCS* 3494, pages 440–456. Springer, Heidelberg, May 2005. (Cited on page 44.)

BBL17. F. Benhamouda, F. Bourse, and H. Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In *PKC 2017, Part II*, *LNCS* 10175, pages 36–66. Springer, Heidelberg, March 2017. (Cited on page 1.)

BCFG17. C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 67–98. Springer, Heidelberg, August 2017. (Cited on page 1.)

BCSW19. M. Barbosa, D. Catalano, A. Soleimanian, and B. Warinschi. Efficient function-hiding functional encryption: From inner-products to orthogonality. In *CT-RSA 2019*, *LNCS* 11405, pages 127–148. Springer, Heidelberg, March 2019. (Cited on page 1.)

BF01. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, *LNCS* 2139, pages 213–229. Springer, Heidelberg, August 2001. (Cited on page 2.)

BGG+14. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT 2014*, *LNCS* 8441, pages 533–556. Springer, Heidelberg, May 2014. (Cited on pages 2, 4, and 28.)

BH08. D. Boneh and M. Hamburg. Generalized identity based and broadcast encryption schemes. In *ASIACRYPT 2008*, *LNCS* 5350, pages 455–470. Springer, Heidelberg, December 2008. (Cited on page 44.)

BJK15. A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. In *ASIACRYPT 2015, Part I*, *LNCS* 9452, pages 470–491. Springer, Heidelberg, November / December 2015. (Cited on page 1.)

BSW07. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007. (Cited on page 2.)

BSW11. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC 2011*, *LNCS* 6597, pages 253–273. Springer, Heidelberg, March 2011. (Cited on pages 1 and 7.)

Cai98. J.-Y. Cai. A relation of primal-dual lattices and the complexity of shortest lattice vector problem. *Theor. Comput. Sci.*, 207(1):105–116, October 1998. (Cited on page 29.)

CDG+18. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT 2018, Part II*, *LNCS* 11273, pages 703–732. Springer, Heidelberg, December 2018. (Cited on pages 1 and 5.)

CGW15. J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In *EUROCRYPT 2015, Part II*, *LNCS* 9057, pages 595–624. Springer, Heidelberg, April 2015. (Cited on pages 9 and 44.)

CGW18. J. Chen, J. Gong, and H. Wee. Improved inner-product encryption with adaptive security and full attribute-hiding. In *ASIACRYPT 2018, Part II*, *LNCS* 11273, pages 673–702. Springer, Heidelberg, December 2018. (Cited on pages 3 and 17.)

CHKP10. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, *LNCS* 6110, pages 523–552. Springer, Heidelberg, May / June 2010. (Cited on page 29.)

CLT18. G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In *ASIACRYPT 2018, Part II*, *LNCS* 11273, pages 733–764. Springer, Heidelberg, December 2018. (Cited on page 1.)

CS19. R. J. Connor and M. Schuchard. Blind bernoulli trials: A noninteractive protocol for hidden-weight coin flips. In *USENIX Security 2019*, pages 1483–1500. USENIX Association, August 2019. (Cited on page 1.)

CZY19. Y. Chen, L. Zhang, and S.-M. Yiu. Practical attribute based inner product functional encryption from simple assumptions. Cryptology ePrint Archive, Report 2019/846, 2019. https://eprint.iacr.org/2019/846. (Cited on page 5.)

DDM16. P. Datta, R. Dutta, and S. Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC 2016, Part I*, *LNCS* 9614, pages 164–195. Springer, Heidelberg, March 2016. (Cited on page 1.)

DM14. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014, Part I*, *LNCS* 8616, pages 335–352. Springer, Heidelberg, August 2014. (Cited on page 24.)

DOT18. P. Datta, T. Okamoto, and J. Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In *PKC 2018, Part II*, *LNCS* 10770, pages 245–277. Springer, Heidelberg, March 2018. (Cited on pages 1 and 34.)

DP19. E. Dufour Sans and D. Pointcheval. Unbounded inner-product functional encryption with succinct keys. In *ACNS 19*, *LNCS* 11464, pages 426–441. Springer, Heidelberg, June 2019. (Cited on pages 1 and 5.)

Gay19. R. Gay. *Public-Key Encryption, Revisited: Tight Security and Richer Functionalities*. PhD thesis, PSL Research University, Paris, France, 2019. (Cited on page 34.)

Gay20. R. Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In *PKC 2020, Part I*, *LNCS* 12110, pages 95–120. Springer, Heidelberg, May 2020. (Cited on page 1.)

GGG+14. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *EUROCRYPT 2014*, *LNCS* 8441, pages 578–602. Springer, Heidelberg, May 2014. (Cited on page 34.)

GIKM00. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, 60(3):592–629, 2000. (Cited on page 3.)

GJLS20. R. Gay, A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. Cryptology ePrint Archive, Report 2020/764, 2020. https://eprint.iacr.org/2020/764. (Cited on page 5.)

GKW15. R. Gay, I. Kerenidis, and H. Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In *CRYPTO 2015, Part II*, *LNCS* 9216, pages 485–502. Springer, Heidelberg, August 2015. (Cited on page 3.)

GPSW06. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Cited on pages 2 and 44.)

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Cited on pages 4, 25, 26, 27, and 42.)

GVW13. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *45th ACM STOC*, pages 545–554. ACM Press, June 2013. (Cited on page 2.)

GVW15. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In *CRYPTO 2015, Part II*, *LNCS* 9216, pages 503–523. Springer, Heidelberg, August 2015. (Cited on page 2.)

JLMS19. A. Jain, H. Lin, C. Matt, and A. Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In *EUROCRYPT 2019, Part I*, *LNCS* 11476, pages 251–281. Springer, Heidelberg, May 2019. (Cited on page 5.)

JLS19. A. Jain, H. Lin, and A. Sahai. Simplifying constructions and assumptions for $i\mathcal{O}$. Cryptology ePrint Archive, Report 2019/1252, 2019. https://eprint.iacr.org/2019/1252. (Cited on page 5.)

KLM+18. S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu. Function-hiding inner product encryption is practical. In *SCN 18*, *LNCS* 11035, pages 544–562. Springer, Heidelberg, September 2018. (Cited on page 1.)

KSW08. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, *LNCS* 4965, pages 146–162. Springer, Heidelberg, April 2008. (Cited on page 2.)

KW93. M. Karchmer and A. Wigderson. On span programs. In *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 102–111. IEEE, 1993. (Cited on page 44.)

KY16. S. Katsumata and S. Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In *ASIACRYPT 2016, Part II*, *LNCS* 10032, pages 682–712. Springer, Heidelberg, December 2016. (Cited on pages 26, 32, and 42.)

Lew12. A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT 2012*, *LNCS* 7237, pages 318–335. Springer, Heidelberg, April 2012. (Cited on page 23.)

Lin17. H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 599–629. Springer, Heidelberg, August 2017. (Cited on page 1.)

LT19. B. Libert and R. Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In *ASIACRYPT 2019, Part III*, *LNCS* 11923, pages 520–551. Springer, Heidelberg, December 2019. (Cited on page 1.)

LVW17. T. Liu, V. Vaikuntanathan, and H. Wee. Conditional disclosure of secrets via non-linear reconstruction. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 758–790. Springer, Heidelberg, August 2017. (Cited on page 3.)

LW10. A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC 2010*, *LNCS* 5978, pages 455–479. Springer, Heidelberg, February 2010. (Cited on page 44.)

MG02. D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*. Kluwer Academic Publishers, Boston, Massachusetts, 2002. (Cited on page 25.)

MP12. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, *LNCS* 7237, pages 700–718. Springer, Heidelberg, April 2012. (Cited on page 29.)

MR04. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004. (Cited on page 25.)

O'N10. A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/2010/556. (Cited on pages 1 and 7.)

OT09. T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT 2009*, *LNCS* 5912, pages 214–231. Springer, Heidelberg, December 2009. (Cited on page 23.)

OT12. T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT 2012*, *LNCS* 7237, pages 591–608. Springer, Heidelberg, April 2012. (Cited on pages 3 and 17.)

Pei07. C. Peikert. Limits on the hardness of lattice problems in ℓ_p norms. *IEEE Conference on Computational Complexity*, pages 333–346, 2007. (Cited on page 25.)

PR06. C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC 2006*, *LNCS* 3876, pages 145–166. Springer, Heidelberg, March 2006. (Cited on page 44.)

Reg05. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93. ACM Press, May 2005. (Cited on page 25.)

RPB+19. T. Ryffel, D. Pointcheval, F. Bach, E. Dufour-Sans, and R. Gay. Partially encrypted deep learning using functional encryption. In *Advances in Neural Information Processing Systems 32*, pages 4519–4530. Curran Associates, Inc., 2019. (Cited on page 1.)

Sha84. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, *LNCS* 196, pages 47–53. Springer, Heidelberg, August 1984. (Cited on page 2.)

SW05. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, *LNCS* 3494, pages 457–473. Springer, Heidelberg, May 2005. (Cited on page 2.)

TT18. J. Tomida and K. Takashima. Unbounded inner product functional encryption from bilinear maps. In *ASIACRYPT 2018, Part II*, *LNCS* 11273, pages 609–639. Springer, Heidelberg, December 2018. (Cited on page 1.)

Wat09. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO 2009*, *LNCS* 5677, pages 619–636. Springer, Heidelberg, August 2009. (Cited on pages 3, 8, and 11.)

Wee14. H. Wee. Dual system encryption via predicate encodings. In *TCC 2014*, *LNCS* 8349, pages 616–637. Springer, Heidelberg, February 2014. (Cited on pages 3, 9, and 17.)

Wee17. H. Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In *TCC 2017, Part I*, *LNCS* 10677, pages 206–233. Springer, Heidelberg, November 2017. (Cited on page 5.)

WFL19. Z. Wang, X. Fan, and F.-H. Liu. FE for inner products and its application to decentralized ABE. In *PKC 2019, Part II*, *LNCS* 11443, pages 97–127. Springer, Heidelberg, April 2019. (Cited on pages 26, 42, and 43.)

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y})}$:
$\mathbf{A} \leftarrow_R \mathbb{Z}_q^{m \times n}$
$\mathbf{Z} \leftarrow_R D_{\mathbb{Z}^{\ell \times m}, \rho}$
$\mathbf{D} \leftarrow \mathbf{ZA} \in \mathbb{Z}_q^{\ell \times n}$
$\mathsf{mpk} \leftarrow (\mathbf{A}, \mathbf{D})$
$\mathsf{msk} \leftarrow \mathbf{Z}$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \vec{y})}$:
Return $(\vec{y}, \mathsf{sk}_{\mathsf{id},\vec{y}} := (\vec{y}^\top \mathbf{Z}))$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \vec{x})}$:
$\vec{s} \leftarrow_R \mathbb{Z}_q^n$
$\vec{e}_1 \leftarrow_R \mathcal{D}_{\mathbf{Z}^m, \sigma}$
$\vec{e}_2 \leftarrow_R \mathcal{D}_{\mathbf{Z}^\ell, \sigma}$
$\mathsf{ct}_1 \leftarrow \mathbf{A}\vec{s} + \vec{e}_1$
$\mathsf{ct}_2 = \mathbf{D}\vec{s} + \vec{e}_2 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x}$
Return $(\mathsf{ct}_1, \mathsf{ct}_2)$

$\underline{\mathsf{Dec}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{sk}_{\mathsf{id},\vec{y}}, \vec{y})}$:
$\mu = \vec{y}^\top \cdot \mathsf{ct}_2 - \mathsf{sk}_{\mathsf{id},\vec{y}} \cdot \mathsf{ct}_1$
$\mu' = \arg\min_{\mu' \in \{0 \ldots K+1\}} \left| \left\lfloor \frac{q}{K} \right\rfloor \cdot \mu - \mu' \right|$
Return μ'

Fig. 15. The ALS inner-product functional encryption scheme $\mathcal{FE}$ from [ALS16].

A The ALS inner-product functional encryption scheme

In this appendix, we recall the inner-product functional encryption scheme from [ALS16], which we denote as ALS. The proof described here is the same as [ALS16], except for a small modification described in [WFL19]. The novelty of [WFL19] is that they described how a rerandomization technique due to [KY16] can be used to simplify the proof of the ALS scheme and improve its parameters. What is changed is that the matrix $\mathbf{Z}$ has a different distribution from [ALS16], which allows us to combine this scheme with the random-oracle based IBE of [GPV08] and the standard model IBE of [ABB10]. The construction is described in Fig. 15. This variant of ALS has the matrix $\mathbf{Z} \leftarrow_R D_{\mathbb{Z}^{\ell \times m}, \rho}$, rather than the slightly more complicated distribution in the original description of the scheme.

Choice of Parameters We first set $\mathcal{X} = \{0, \ldots, P-1\}^\ell, \mathcal{Y} = \{0, \ldots, V-1\}^\ell$, the output must belong to gs to $\{0, \ldots, K-1\}$ with $K = \ell PV$. First we set the parameters for correctness:

- From Lemma 4.2, we know that every entry of $\mathbf{D}$ is with overwhelming probability bounded by $\omega(\log n)$, so $\|\mathbf{D}\| \leq \sqrt{\ell} \cdot \omega(\log n)$, as long as $\rho \geq \omega(\sqrt{\log n})$.
- Now we bound $\|\vec{y}^\top \mathbf{D}\vec{e}_1\| \leq \ell\sqrt{\ell}V\omega(\log^2 n)$ and $\|\vec{y}\vec{e}_2\| \leq \ell V\omega(\sqrt{\log n})$, as long as $\sigma \geq \omega(\sqrt{\log n})$.
- For decryption to suceed, we want that the error terms are smaller than $\frac{q}{2K}$, which implies: $q \geq 2K\ell\sqrt{\ell}V\omega(\log^2 n)$

We therefore have the following lemma:

Lemma A.1. *For $q \geq 2K\ell\sqrt{\ell}V\omega(\log^2 n)$, $\sigma, \rho > \omega(\sqrt{\log n})$ the* ALS *scheme from Fig. 15 is correct.*

The security reduction is with respect to $\mathsf{LWE}_{q,\alpha,n}$, where the modulus will be much larger than K. Denote the standard deviation $r := \alpha q$. To ensure security, we need to satisfy the following requirements:

- NoiseGen rerandomization: looking ahead in the security proof, we require that the singular value $s_1(\mathbf{V})$ of $\mathbf{V}$ is bounded by $h := \Omega(\rho(\sqrt{\ell} + \sqrt{m} + \sqrt{n}))$ (see the security proof for exact values and security parameter n). Then, in order for NoiseGen to provide security guarantees, we must set $\sigma \geq 2 \cdot r \cdot h$. Since h depends on the gaussian parameter ρ of matrix $\mathbf{Z}$, this establishes a relation between ρ and σ.
- Min-entropy requirement: since $\rho > \omega(\sqrt{\log n})$, the min-entropy requirement is satisfied (as long as $n > 2.303$) and $m \geq (2n \log q + 2n)/\log(4/3)$.
- Leftover hash lemma: this is satisfied as long as m is much larger than n (see [ALS16, Lemma 11]). For the purposes of this paper, we will choose $m = 2n \log q$.

Putting all requirements together, we choose as parameters:

$$\begin{gathered} q \geq 2K\ell\sqrt{\ell}V\omega(\log^2 n) \\ \sigma = 2C\alpha q(\sqrt{m} + \sqrt{n} + \sqrt{\ell}) \\ \rho \geq \omega(\sqrt{\log n}). \\ m = 2n \log q \end{gathered}$$

$\mathsf{LWE}_{q,\alpha,n}$ is reducible to lattice problems in dimension n when $q\alpha \geq \Omega(\sqrt{n})$. With this choice of parameters, we have the following theorem:

Theorem A.2. [ALS16, WFL19] *Let n be the security parameter, $m \geq 2n \log q$ and q,σ,ρ, $\alpha \leq \frac{\sigma}{2C\alpha q(\sqrt{m}+\sqrt{n}+\sqrt{\ell})}$ as described above. Then, the scheme from Fig. 15 is* AD-IND*-secure, assuming that $\mathsf{LWE}_{q,\alpha,n}$ is hard.*

Proof. As mentioned previously, we recall the proof of the ALS scheme from [ALS16] in which Game 2 is modified with the idea described in [WFL19].

Game 1 This is the AD-IND game.

Game 2 Draw an error $\vec{e} \leftarrow_{\mathrm{R}} \mathcal{D}_{\mathbb{Z}^m,r}$ and then compute $\vec{c} = \mathbf{A}\vec{s}+\vec{e}$. Let $\mathbf{I}_m$ denote the $m \times m$ identity matrix. Now we apply Lemma 4.7 for $\mathbf{V} = \begin{pmatrix} \mathbf{I}_m \\ \mathbf{Z} \end{pmatrix} \in \mathbb{Z}^{(m+\ell)\times m}$. Let $\sigma' > s_1(\mathbf{V})$. By Lemma 4.1, we know that $s_1(\mathbf{Z}) \leq C\rho(\sqrt{\ell} + \sqrt{m} + \sqrt{\lambda})$ except with $\frac{1}{e^{2\pi\lambda}}$ probability and we use this to argue that $s_1(\mathbf{V}) \leq \sqrt{(C\rho(\sqrt{\ell}+\sqrt{m}+\sqrt{\lambda}))^2+1}$

By computing $\vec{c}' = \mathbf{V}\vec{c} + \mathsf{NoiseGen}(\mathbf{V},\sigma')$, we obtain $\vec{c}' = \mathbf{V}\cdot\vec{c}+\vec{f}$, where $\vec{f} \leftarrow_{\mathrm{R}} \mathcal{D}_{\mathbb{Z}^{m+\ell},2r\sigma'}$. Notice that we can split $\vec{c}'$ into $\mathsf{ct}_1 \leftarrow \mathbf{A}\vec{s}+\vec{e}_1$ and $\mathsf{ct}_2 \leftarrow \mathbf{D}\vec{s}+\vec{e}_2$, where $e_1 \leftarrow \mathcal{D}_{\mathbb{Z}^m,2r\sigma'}$ and $e_2 \leftarrow \mathcal{D}_{\mathbb{Z}^\ell,2r\sigma'}$. Now, to simulate ct_2 correctly, we need to compute $\mathsf{ct}_2' = \mathsf{ct}_2 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x}_\beta$.

Game 3 By the LWE assumption, we change $\vec{c} = \mathbf{A}\vec{s}+\vec{e}$ into a uniformly random element $\vec{u}$ of $\mathbb{Z}_q^m$. Now we argue that in this game, the probability the adversary has of winning is $\frac{1}{2}$. This step of the proof is identical to [ALS16, Theorem 2], which we recall here for completeness.

Consider $\vec{y}^i$ the decryption key queries made by the adversary. Then, $\langle \vec{x}_0, \vec{y}^i \rangle = \langle \vec{x}_1, \vec{y}^i \rangle$ over $\mathbb{Z}$, for all i. Define $\vec{x} := \frac{1}{g}(\vec{x}_1 - \vec{x}_0) = \frac{1}{g}(x_1, \ldots, x_\ell)$, where $g \neq 0$ is the gcd of all coefficients of $\vec{x}_0 - \vec{x}_1$. Since $\langle \vec{x}, \vec{y}^i \rangle = 0$, this means all decryption key queries must belong to the lattice $\{\vec{y} \in \mathbb{Z}^\ell : \langle \vec{x}, \vec{y} \rangle = 0\}$. Assume without loss of generality that the first n_0 entries of $\vec{x}$ are zero, while the rest are non-zero.

Define matrix $\mathbf{Y}_{top} = \left(\begin{array}{c|ccccc} \mathbf{I}_{n_0} & & & & & \\ \hline & -x_{n_0+2} & -x_{n_0+1} & & & \\ & & -x_{n_0+3} & & & \\ & & & \ddots & \ddots & \\ & & & & & \\ & & & & -x_\ell & x_{\ell-1} \end{array}\right) \in \mathbb{Z}^{\ell-1\times(\ell)}$

Then they write $\mathbf{Y}_{bot} = \vec{x}^\top \in \mathbb{Z}^{1\times\ell}$ and define $\mathbf{Y} := \begin{pmatrix} \mathbf{Y}_{top} \\ \mathbf{Y}_{bot} \end{pmatrix} \in \mathbb{Z}_q^{\ell\times\ell}$. By induction, they show that $\det(\mathbf{Y}\mathbf{Y}^\top) = (\prod_{k=n_0+2}^{\ell-1} x_k^2) \cdot \|\vec{x}\|^4$. As long as q is large enough, this quantity is non-zero modulo q, which implies that $\mathbf{Y}$ is invertible modulo q. Observe that for $(\vec{g}_1, \vec{g}_2) = \mathsf{NoiseGen}(\mathbf{V},\sigma') : \mathsf{ct}_1 = \vec{u} + \vec{g}_1$ and $\mathsf{ct}_2 = \mathbf{Z}\cdot\vec{u} + \vec{g}_2 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x}_\beta$, where $\vec{u} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^m$.

The proof strategy of [ALS16] is to show that the distribution of $\mathbf{Y}\cdot\mathsf{ct}_2 \in \mathbb{Z}_q^{\ell\times 1}$ is almost independent of β and therefore the probability of winning the game is almost $\frac{1}{2}$.

Rewrite $\mathsf{ct}_2 = \mathbf{Y}^{-1}\cdot\mathbf{Y}\cdot(\mathbf{Z}\vec{u} + \vec{g}_2 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x}_\beta) \bmod q$.

Notice that $\mathbf{Y}\mathsf{ct}_2 = \begin{pmatrix} \mathbf{Y}_{top}\mathsf{ct}_2 \\ \mathbf{Y}_{bot}\mathsf{ct}_2 \end{pmatrix}$. Now $\mathbf{Y}_{top}\mathsf{ct}_2$ does not depend on β because $\mathbf{Y}_{top}\cdot\vec{x}_0 = \mathbf{Y}_{top}\cdot\vec{x}_1$, from the functional encryption constraints.

It remains to analyze $\mathbf{Y}_{bot}\mathsf{ct}_2$. Here, [ALS16] prove the following variant of the leftover hash lemma:

Lemma A.3. [ALS16] *Conditioned on $(\mathbf{A}, \mathbf{Z}\mathbf{A}, \mathbf{Y}_{top}, \mathbf{Y}_{top}\mathbf{Z})$, the min-entropy of $\mathbf{Y}_{bot}Z$ is greater or equal than $n\log q + 2\lambda$.*

This lemma can be applied as long as the following lemma due to [PR06] and adapted in [ALS16] holds for the standard deviation used to generate $\mathbf{Z}$:

Lemma A.4. [ALS16] *Let $\Lambda = k \cdot \mathbb{Z}$ be a 1-dimensional lattice. For any $\sigma \geq 10 \cdot k, b \in \lambda$ and $c \in \mathbb{R}$, we have that $\mathcal{D}_{\Lambda,\sigma,c}(b) \leq 3/4$. In particular, we have $H_\infty(D_{\Lambda,\sigma,c}) \geq 0.4$, where $H_\infty()$ denotes the min-entropy.*

As long as the previous two lemmas hold, one can use the leftover hash lemma for the seed $\vec{u}$ and randomness $\mathbf{Y}_{bot}\mathbf{Z}$ to argue that given $(\mathbf{A}, \mathbf{ZA}, \mathbf{Y}_{top}, \mathbf{Y}_{top}\mathbf{Z})$, the pair $(\vec{u}, \mathbf{Y}_{bot}\mathbf{Z}\vec{u})$ is $1/2^\lambda$ statistically close from the uniform distribution over $\mathbb{Z}_q^m \times \mathbb{Z}_q$, therefore hiding β. The variant of the leftover hash lemma used is the following:

Lemma A.5 (Lemma 10 in [ALS16]). *Let $n, m, q \geq 2$ be positive integers. Assume that $q = p^k$ for p prime and $k \geq 1$. Assume further that $m \geq 2n \log_2 q$. Let $\sigma \geq \Omega(\sqrt{n + \log m})$ and $c \in \mathbb{Z}^m$. Then for $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ sampled uniformly and $\vec{e} \in \mathbb{Z}^m$ sampled from $\mathcal{D}_{\mathbb{Z}^m,\sigma,c}$, the distribution of the pair $(\mathbf{A}, \vec{e}^\top \cdot \mathbf{A})$ is within statistical distance $2^{-\Omega(n)}$ of uniform over $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^n$.*

B Instantiations of Predicate Encodings

B.1 Inner-product encryption with short ciphertexts [BBG05]

The predicate is described by a vector $\vec{y} \in \mathbb{Z}_p^n$, takes as input an attribute $\vec{x} \in \mathbb{Z}_p^n$, and outputs 1 if $\vec{x}^\top \vec{y} = 0$, 0 otherwise.

- Param: returns the parameters $(n + 2, |\mathsf{ct}| = 1, |\mathsf{sk}| = n + 1)$.
- $\mathsf{EncCt}(\vec{x})$: given $\vec{x} \in \mathbb{Z}_p^n$ returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{(n+2)\times 1}$ such that $\mathbf{C}^\top(v_1, v_2, \vec{w}) = v_1 + \vec{x}^\top \vec{w} \in \mathbb{Z}_p$.
- $\mathsf{EncKey}(\vec{y})$: given $\vec{y} \in \mathbb{Z}_p^n$ returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{(n+3)\times(n+1)}$ such that $\mathbf{K}^\top(\alpha, v_1, v_2, \vec{w}) = (v_2\vec{y} + \vec{w}, v_1 + \alpha) \in \mathbb{Z}_p^{n+1}$.
- $\mathsf{Decode}(\vec{x}, \vec{y})$: if $\vec{x}^\top \vec{y} = 0$, it returns the vector $\vec{d} := \begin{pmatrix} -1 \\ \vec{x} \\ 1 \end{pmatrix} \in \mathbb{Z}_p^{n+2}$.

B.2 Ciphertext-policy ABE for monotone span programs [GPSW06]

We recall the definition of read-once monotone span programs [KW93]. The predicate is described by a matrix $\mathbf{M} \in \mathbb{Z}_p^{\ell \times n}$, takes as input an attribute $\vec{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, and outputs 1 if the vector $(1, 0, \ldots, 0) \in \mathbb{Z}_p^n$ is in the column span of $\mathbf{M}_{\vec{x}}$, which denotes the collection of vectors $\{\mathbf{M}_j : x_j = 1\}$ where $\mathbf{M}_j$ denotes the j'th column of $\mathbf{M}$. That is, $\vec{x}$ satisfies $\mathbf{M}$ if there exists constants $\omega_1, \ldots, \omega_n \in \mathbb{Z}_p$ such that

$$\sum_{j:x_j=1} \omega_j \mathbf{M}_j = (1, 0, \ldots, 0) \tag{2}$$

Observe that the constants $\{\omega_j\}_{j\in[n]}$ can be computed in time polynomial in the size of the matrix $\mathbf{M}$ via Gaussian elimination.

- Param: returns the parameters $(n + \ell, |\mathsf{ct}| = n, |\mathsf{sk}| = n + 1)$.
- $\mathsf{EncCt}(\mathbf{M})$: given $\mathbf{M} \in \mathbb{Z}_p^{n\times\ell}$ returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{(n+2)\times 1}$ such that $\mathbf{C}^\top(\vec{w}, \vec{u}, \gamma) = (w_1 + (\gamma, \vec{u})^\top \mathbf{M}_1, \ldots, w_n + (\gamma, \vec{u})^\top \mathbf{M}_n) \in \mathbb{Z}_p^n$, where $\mathbf{M}_j$ denotes the j'th column of $\mathbf{M}$.
- $\mathsf{EncKey}(\vec{x})$: given $\vec{x} \in \{0,1\}^n$, returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{(n+3)\times(n+1)}$ such that $\mathbf{K}^\top(\alpha, \vec{w}, \vec{u}, \gamma) = (\gamma + \alpha, x_1 w_1, \ldots, x_n w_n) \in \mathbb{Z}_p^{n+1}$.
- $\mathsf{Decode}(\vec{x}, \vec{y})$: if $\vec{x}^\top \vec{y} = 0$, it returns the vector $\vec{d} := (x_1\omega_1, \ldots, x_n\omega_n, 1, \omega_1, \ldots, \omega_n) \in \mathbb{Z}_p^{2n+1}$.

More predicate encodings can be found in [CGW15], yielding inner-product encryption with short keys [BB04], non-zero inner-product encryption (introduced in [AL10]), spatial encryption [BH08, BBG05, LW10], ciphertext-policy for arithmetic span programs [CGW15].

C Instantiations of Function Encodings

C.1 Identity-Based Inner-Product Functional Encryption

Each function is described by an identity $\mathsf{id} \in \mathbb{Z}_p$ and a vector $\vec{y} \in [0,B]^d$, takes as input another identity $\mathsf{id}' \in \mathbb{Z}_p$ and a vector $\vec{x} \in [0,B]^d$, and outputs $\vec{x}^\top \vec{y}$ if $\mathsf{id} = \mathsf{id}'$, 0 otherwise. The partial information $\mathsf{part}(\vec{x}, \mathsf{id}) = \mathsf{id}$.

- Param: returns the parameters $(2d, |\mathsf{ct}| = d, |\mathsf{sk}| = n+1)$.
- $\mathsf{EncCt}(\vec{x}, \mathsf{id})$: given $\vec{x} \in \mathbb{Z}_p^n$ and $\mathsf{id} \in \mathbb{Z}_p$, returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{(2d+1)\times d}$ such that $\mathbf{C}^\top(w_0, \vec{w}_1, \vec{w}_2) = (w_0\vec{x} + \vec{w}_1 + \mathsf{id}\vec{w}_2) \in \mathbb{Z}_p^d$.
- $\mathsf{EncKey}(\vec{y}, \mathsf{id}')$: given $\vec{y} \in \mathbb{Z}_p^n$ and $\mathsf{id}' \in \mathbb{Z}_p$, returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{(2d+1)\times 1}$ such that $\mathbf{K}^\top(w_0, \vec{w}_1, \vec{w}_2) = \vec{y}^\top(\vec{w}_1 + \mathsf{id}'\vec{w}_2) \in \mathbb{Z}_p$.
- $\mathsf{Decode}(\mathsf{id}, \mathsf{id}', \vec{y})$: if $\vec{x}^\top \vec{y} = 0$, it returns the vector $\vec{d} := (\vec{y}, -1) \in \mathbb{Z}_p^{d+1}$.

Correctness. Let $\mathsf{id} \in \mathbb{Z}_p$, $\vec{x}, \vec{y} \in [0,B]^d$, $\mathbf{C} := \mathsf{EncCt}(\vec{x}, \mathsf{id})$, $\mathbf{K} := \mathsf{EncKey}(\vec{y}, \mathsf{id})$ and $\vec{d} := \mathsf{Decode}(\mathsf{id}, \mathsf{id}', \vec{y})$. We have $(w_0, \vec{w}_1, \vec{w}_2)^\top(\mathbf{C}|\mathbf{K})\vec{d} = (w_0\vec{x} + \vec{w}_1 + \mathsf{id}\vec{w}_2)^\top \vec{y} - \vec{y}^\top(\vec{w}_1 + \mathsf{id}\vec{w}_2) = w_0\vec{x}^\top\vec{y}$.

Security. Let $\mathsf{id}^0 = \mathsf{id}^1, \mathsf{id}' \in \mathbb{Z}_p$, $\vec{x}^0, \vec{x}^1, \vec{y} \in [0,B]^d$ for all $b \in \{0,1\}$, $\mathbf{C}^b := \mathsf{EncCt}(\vec{x}^b, \mathsf{id}^b)$, $\mathbf{K} := \mathsf{EncKey}(\vec{y}, \mathsf{id})$. If $\mathsf{id}^0 = \mathsf{id}^1 = \mathsf{id}'$, then it must be that $\vec{y}^\top\vec{x}^0 = \vec{y}^\top\vec{x}^1$. In that case, security relies on the fact that the following are identically distributed: $\vec{w}_1$ and $\vec{w}_1 + \vec{x}^1 - \vec{x}^0$, for $\vec{w}_1 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$. The left most distribution corresponds to $(w_0|\vec{w}_1^\top|\vec{w}_2^\top)(\mathbf{C}^0|\mathbf{K})$ whereas the rightmost distribution corresponds to $(w_0|\vec{w}_1^\top|\vec{w}_2^\top)(\mathbf{C}^1|\mathbf{K})$.

If $\mathsf{id}^0 = \mathsf{id}^1 \neq \mathsf{id}'$, the security relies on the fact that the following are identically distributed: $(\vec{w}_1 + \mathsf{id}\vec{w}_2, \vec{w}_1 + \mathsf{id}'\vec{w}_2)$ and $(\vec{w}_1 + \mathsf{id}\vec{w}_2 + \vec{x}^1 - \vec{x}^0, \vec{w}_1 + \mathsf{id}'\vec{w}_2)$, where$\vec{w}_1, \vec{w}_2 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$. The left most distribution corresponds to $(w_0|\vec{w}_1^\top|\vec{w}_2^\top)(\mathbf{C}^0|\mathbf{K})$ whereas the rightmost distribution corresponds to $(w_0|\vec{w}_1^\top|\vec{w}_2^\top)(\mathbf{C}^1|\mathbf{K})$.

C.2 Orthogonality Testing with Inner-Product Functional Encryption

Each function is described by a predicate vector $\widetilde{\vec{y}} \in \mathbb{Z}_p^\ell$ and a message vector $\vec{y} \in [0,B]^d$, takes as input a pair $\left(\widetilde{\vec{x}} \in \mathbb{Z}_p^\ell \setminus \{\mathbf{0}\}, \vec{x} \in [0,B]^d\right)$, and returns $\vec{x}^\top\vec{y}$ if $\widetilde{\vec{x}}^\top\widetilde{\vec{y}} = 0$. The partial information is $\mathsf{part}(\widetilde{\vec{x}}, \vec{x}) = \widetilde{\vec{x}}$.

- Param: returns the parameters $(d\ell + 1, |\mathsf{ct}| = 1, |\mathsf{sk}| = n+1)$.
- $\mathsf{EncCt}(\vec{x}, \widetilde{\vec{x}})$: returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{(d\ell+2)\times d}$ such that $\mathbf{C}^\top(w_0, u, \vec{w}_1, \ldots, \vec{w}_\ell) = \mathbf{W}\widetilde{\vec{x}} + w_0\vec{x} \in \mathbb{Z}_p^d$, where $\mathbf{W} \in \mathbb{Z}_p^{d\times\ell}$ denotes the matrix whose i'th column is $\vec{w}_i$, for all $i \in [\ell]$.
- $\mathsf{EncKey}(\vec{y}, \widetilde{\vec{y}})$: returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{(d\ell+2)\times\ell}$ such that $\mathbf{K}^\top(w_0, u, \vec{w}_1, \ldots, \vec{w}_\ell) = \mathbf{W}^\top\vec{y} + u\widetilde{\vec{y}} \in \mathbb{Z}_p^\ell$.
- $\mathsf{Decode}(\vec{x}, \vec{y})$: if $\vec{x}^\top\vec{y} = 0$, it returns the vector $\vec{d} := (\vec{y}, \widetilde{\vec{x}}) \in \mathbb{Z}_p^{d+\ell}$.

Correctness. Let $\widetilde{\vec{x}}, \widetilde{\vec{y}} \in \mathbb{Z}_p^\ell$ such that $\widetilde{\vec{x}}^\top\widetilde{\vec{y}} = 0$, $\vec{x}, \vec{y} \in [0,B]^d$, $\mathbf{C} := \mathsf{EncCt}(\vec{x}, \widetilde{\vec{x}})$, $\mathbf{K} := \mathsf{EncKey}(\vec{y}, \widetilde{\vec{y}})$ and $\vec{d} := \mathsf{Decode}(\widetilde{\vec{x}}, \widetilde{\vec{y}}, \vec{y})$. We have $(w_0, u, \vec{w}_1, \ldots, \vec{w}_\ell)^\top(\mathbf{C}|\mathbf{K})\vec{d} = \vec{y}^\top(\mathbf{W}\widetilde{\vec{x}} + w_0\vec{x}) - (\vec{y}^\top\mathbf{W} + u\widetilde{\vec{y}})\widetilde{\vec{x}} = w_0\vec{x}^\top\vec{y}$, where the last equality uses the fact that $\widetilde{\vec{x}}^\top\widetilde{\vec{y}} = 0$.

Security. Let $\widetilde{\vec{x}}^0 = \widetilde{\vec{x}}^1, \widetilde{\vec{y}} \in \mathbb{Z}_p^\ell$, $\vec{x}^0, \vec{x}^1, \vec{y} \in [0,B]^d$, for all $b \in \{0,1\}$, $\mathbf{C}^b := \mathsf{EncCt}(\vec{x}^b, \widetilde{\vec{x}}^b)$, $\mathbf{K} := \mathsf{EncKey}(\vec{y}, \widetilde{\vec{y}})$. If $\widetilde{\vec{y}}^\top\widetilde{\vec{x}}^0 = \widetilde{\vec{y}}^\top\widetilde{\vec{x}}^1 = 0$, then it must be that $\vec{y}^\top\vec{x}^0 = \vec{y}^\top\vec{x}^1$. In that case, security relies on the fact that the following are identically distributed: $\mathbf{W}$ and $\mathbf{W} + w_0(\vec{x}^1 - \vec{x}^0)\vec{v}^\top$, where $\mathbf{W} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{d\times\ell}$, and $\vec{v} \in \mathbb{Z}_p^\ell$ is an arbitrary vector such that $\widetilde{\vec{x}}^\top\vec{v} = 1$ (such a vector exists since $\widetilde{\vec{x}} \neq \mathbf{0}$).

If $\widetilde{\vec{y}}^\top\widetilde{\vec{x}}^0 = \widetilde{\vec{y}}^\top\widetilde{\vec{x}}^1 \neq 0$, then security relies on the fact that the following are identically distributed: $(\mathbf{W}, u)$ and $\left(\mathbf{W} + \frac{w_0(\vec{x}^1 - \vec{x}^0)\widetilde{\vec{y}}^\top}{\widetilde{\vec{y}}^\top\widetilde{\vec{x}}^0}, u - \frac{w_0(\vec{x}^1 - \vec{x}^0)^\top\vec{y}}{\widetilde{\vec{y}}^\top\widetilde{\vec{x}}^0}\right)$, with $\mathbf{W} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{d\times\ell}$ and $u \leftarrow_{\mathrm{R}} \mathbb{Z}_p$. The left most distribution corresponds to $(w_0, u, \vec{w}_1, \ldots, \vec{w}_\ell)^\top(\mathbf{C}^0|\mathbf{K})$ whereas the rightmost distribution corresponds to $(w_0, u, \vec{w}_1, \ldots, \vec{w}_\ell)^\top(\mathbf{C}^1|\mathbf{K})$.

C.3 Fully-hiding Orthogonality Testing with Inner-Product Functional Encryption

Each function is described by a predicate vector $\widetilde{\vec{y}} \in \mathbb{Z}_p^\ell$ and a message vector $\vec{y} \in [0,B]^d$, takes as input a pair $\left(\widetilde{\vec{x}} \in \mathbb{Z}_p^\ell \setminus \{\mathbf{0}\}, \vec{x} \in [0,B]^d\right)$, and returns $\vec{x}^\top \vec{y}$ if $\widetilde{\vec{x}}^\top \widetilde{\vec{y}} = 0$. There is no partial information: $\mathsf{part}(\widetilde{\vec{x}}, \vec{x}) = \emptyset$.

- Param: returns the parameters $(2d + d\ell, |\mathsf{ct}| = 1, |\mathsf{sk}| = n + 1)$.
- $\mathsf{EncCt}(\vec{x}, \widetilde{\vec{x}})$: returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{(2d+d\ell+1)\times(d\ell+d)}$ such that $\mathbf{C}^\top(w_0, \vec{u}, \vec{v}, \mathsf{vect}(\mathbf{W})) = (\mathsf{vect}(\vec{u}\widetilde{\vec{x}}^\top + \mathbf{W}), \vec{u} + w_0\vec{x}) \in \mathbb{Z}_p^d$, where for any matrix $\mathbf{M} \in \mathbb{Z}_p^{n\times m}$, we denote by $\mathsf{vect}(\mathbf{M}) \in \mathbb{Z}_p^{nm}$ the vector such that for all $\vec{x} \in \mathbb{Z}_p^n$, $\vec{y} \in \mathbb{Z}_p^m$, $\mathsf{vect}(\mathbf{M})^\top(\vec{x} \otimes \vec{y}) = \vec{x}^\top \mathbf{M} \vec{y}$. Here, $\vec{u}, \vec{v} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$, and $\mathbf{W} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{d\times\ell}$.
- $\mathsf{EncKey}(\vec{y}, \widetilde{\vec{y}})$: returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{(2d+d\ell+1)\times 1}$ such that $\mathbf{K}^\top(w_0, \vec{u}, \vec{v}, \mathsf{vect}(\mathbf{W})) = \vec{y}^\top \mathbf{W} \widetilde{\vec{y}} + \vec{y}^\top \vec{v} \in \mathbb{Z}_p$.
- $\mathsf{Decode}(\widetilde{\vec{y}}, \vec{y})$: it returns the vector $\vec{d} := (\vec{y} \otimes \widetilde{\vec{y}}, \vec{y}, -1) \in \mathbb{Z}_p^{d\ell+d+1}$.

Correctness. Let $\widetilde{\vec{x}}, \widetilde{\vec{y}} \in \mathbb{Z}_p^\ell$ such that $\widetilde{\vec{x}}^\top \widetilde{\vec{y}} = 0$, $\vec{x}, \vec{y} \in [0,B]^d$, $\mathbf{C} := \mathsf{EncCt}(\vec{x}, \widetilde{\vec{x}})$, $\mathbf{K} := \mathsf{EncKey}(\vec{y}, \widetilde{\vec{y}})$ and $\vec{d} := \mathsf{Decode}(\widetilde{\vec{y}}, \vec{y})$. We have $(w_0, \vec{u}, \vec{v}, \mathsf{vect}(\mathbf{W}))^\top(\mathbf{C}|\mathbf{K})\vec{d} = \vec{y}^\top(\vec{u}\widetilde{\vec{x}}^\top + \mathbf{W})\widetilde{\vec{y}} + (\vec{v} + w_0\vec{x})^\top \vec{y} - \vec{y}^\top \mathbf{W}\widetilde{\vec{y}} + \vec{y}^\top \vec{v} = w_0 \vec{x}^\top \vec{y}$, where the last equality uses the fact that $\widetilde{\vec{x}}^\top \widetilde{\vec{y}} = 0$.

Security. Let $\widetilde{\vec{x}}^0, \widetilde{\vec{x}}^1, \widetilde{\vec{y}} \in \mathbb{Z}_p^\ell$, $\vec{x}^0, \vec{x}^1, \vec{y} \in [0,B]^d$, for all $b \in \{0,1\}$, $\mathbf{C}^b := \mathsf{EncCt}(\vec{x}^b, \widetilde{\vec{x}}^b)$, $\mathbf{K} := \mathsf{EncKey}(\vec{y}, \widetilde{\vec{y}})$. If $\widetilde{\vec{y}}^\top \widetilde{\vec{x}}^0 = \widetilde{\vec{y}}^\top \widetilde{\vec{x}}^1 = 0$, then it must be that $\vec{y}^\top \vec{x}^0 = \vec{y}^\top \vec{x}^1$. In that case, security relies on the fact that the following are identically distributed: $\vec{v}$ and $\vec{v} + w_0(\vec{x}^1 - \vec{x}^0)$, where $\vec{v} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$.

If $\widetilde{\vec{y}}^\top \widetilde{\vec{x}}^0 \neq 0$ and $\widetilde{\vec{y}}^\top \widetilde{\vec{x}}^1 \neq 0$, then we show that $(w_0, \vec{u}, \vec{v}, \mathsf{vect}(\mathbf{W}))^\top(\mathbf{C}^b|\mathbf{K})$ is independent of b, using the fact that the following are identically distributed: $(\mathbf{W}, \vec{v})$ and $(\mathbf{W} + \vec{u}(\vec{x}^1 - \vec{x}^0)^\top, \vec{v} + w_0(\vec{x}^1 - \vec{x}^0))$, with $\mathbf{W} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{d\times\ell}$ and $\vec{v} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$. The left most distribution corresponds to $(w_0, u, \vec{w}_1, \ldots, \vec{w}_\ell)^\top(\mathbf{C}^b|\mathbf{K})^\top(w_0, u, \vec{w}_1, \ldots, \vec{w}_\ell) := (\mathsf{vect}(\vec{u}\widetilde{\vec{x}}^{b\top} + \mathbf{W}), \vec{v} + w_0\vec{x}^b, \vec{y}^\top \mathbf{W}\widetilde{\vec{y}} + \vec{y}^\top \vec{v})$ whereas the rightmost distribution corresponds to $(\mathsf{vect}(\mathbf{W}), \vec{v}, \vec{y}^\top \mathbf{W}\widetilde{\vec{y}} + \vec{y}^\top \vec{v} - \boxed{w_0 \vec{y}^\top \vec{x}^b - \vec{y}^\top \vec{u} \underbrace{\widetilde{\vec{y}}^\top \cdot \widetilde{\vec{x}}^b}_{\neq 0}})$. If $\vec{y} = \mathbf{0}$, then the gray term in the last distribution disappears, which means the distribution is independent of b. If $\vec{y} \neq \mathbf{0}$, the value $\vec{y}^\top \vec{u}$ is uniformly random over $\mathbb{Z}_p$, and since the value $\widetilde{\vec{y}}^\top \cdot \widetilde{\vec{x}}^b$ is different from 0, that means the gray term itself is uniformly random over $\mathbb{Z}_p$, independent of b.

C.4 Ciphertext-policy ABE for Read-once Monotone Span Programs with Inner-Product Functional Encryption

Each function is described by a characteristic vector $\vec{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$, and a vector $\vec{y} \in [O,B]^d$. It takes as input an access structure, which is a matrix $\mathbf{M} \in \mathbb{Z}_p^{n\times\ell}$, and a vector $\vec{x} \in [0,B]^d$ and outputs $\vec{x}^\top \vec{y}$ if the vector $(1,0,\ldots,0)^\top \in \mathbb{Z}_p^{1\times\ell}$ is in the row span of $\mathbf{M}_{\vec{a}}$, which denotes the collection of vectors $\{\mathbf{M}_j : a_j = 1\}$ where $\mathbf{M}_j$ denotes the j'th row of $\mathbf{M}$. That is, $\vec{a}$ satisfies $\mathbf{M}$ if there exists constants $\omega_1, \ldots, \omega_n \in \mathbb{Z}_p$ such that

$$\sum_{j:a_j=1} \omega_j \mathbf{M}_j = (1,0,\ldots,0) \tag{3}$$

Observe that the constants $\{\omega_j\}_{j\in[n]}$ can be computed in time polynomial in the size of the matrix $\mathbf{M}$ via Gaussian elimination.

The partial information is: $\mathsf{part}(\vec{a}, \vec{x}) = \vec{a}$.

- Param: returns the parameters $(2d + d\ell, |\mathsf{ct}| = 1, |\mathsf{sk}| = n + 1)$.
- $\mathsf{EncCt}(\vec{x}, \mathbf{M})$: returns a matrix $\mathbf{C} \in \mathbb{Z}_p^{2dn\times d(n+1)}$ such that $\mathbf{C}^\top(w_0, \vec{v}, \mathsf{vect}(\mathbf{U}), \vec{w}_1, \ldots, \vec{w}_n) = ((\vec{v}|\mathbf{U})\mathbf{M}_1 + \vec{w}_1, \ldots, (\vec{v}|\mathbf{U})\mathbf{M}_n + \vec{w}_n, \vec{v} + w_0\vec{x}) \in \mathbb{Z}_p^{d(n+1)}$. Here, $\vec{v} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$, $\vec{w}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$ for all $i \in [n]$, and $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{d\times n}$.
- $\mathsf{EncKey}(\vec{y}, \vec{a})$: returns a matrix $\mathbf{K} \in \mathbb{Z}_p^{2dn\times dn}$ such that $\mathbf{K}^\top(w_0, \vec{v}, \mathsf{vect}(\mathbf{U})) = (\vec{w}_1 a_1, \ldots, \vec{w}_n a_n) \in \mathbb{Z}_p^{dn}$.
- $\mathsf{Decode}(\widetilde{\vec{y}}, \vec{y})$: it returns the vector $\vec{d} := (-\vec{y}\omega_1 a_1, \ldots, -\vec{y}\omega_n a_n, \vec{y}, \vec{y}\omega_1, \ldots \vec{y}\omega_n) \in \mathbb{Z}_p^{2dn+d}$.

Correctness. Let $\vec{a} \in \{0,1\}^n$ ant $\mathbf{M} \in \mathbb{Z}_p^{n\times\ell}$ such that $\vec{a}$ satisfies $\mathbf{M}$, $\vec{x}, \vec{y} \in [0,B]^d$, $\mathbf{C} := \mathsf{EncCt}(\vec{x}, \mathbf{M})$, $\mathbf{K} := \mathsf{EncKey}(\vec{y}, \vec{a})$ and $\vec{d} := \mathsf{Decode}(\vec{a}, \mathbf{M}, \vec{y})$. We have $(w_0, \vec{v}, \mathsf{vect}(\mathbf{U}), \vec{w}_1, \ldots, \vec{w}_n)^\top (\mathbf{C}|\mathbf{K})\vec{d} = -\sum_{j:a_j=1} \omega_j (\vec{y}^\top (\vec{v}|\mathbf{U})\mathbf{M}_j + \vec{y}^\top \vec{w}_j) + w_0 \vec{x}^\top \vec{y} + \vec{y}^\top \vec{v} + \vec{y}^\top \mathbf{W}\widetilde{\vec{y}} + \sum_{j:a_j=1} \omega_j \vec{y}^\top (\vec{v}|\mathbf{U})\mathbf{M}_j = w_0 \vec{x}^\top \vec{y}$, where the last equality uses (3).

Security. Let $\vec{a} \in \{0,1\}^n$, $\vec{x}^0, \vec{x}^1, \vec{y} \in [0,B]^d$, $\mathbf{M} \in \mathbb{Z}_p^{n\times\ell}$, for all $b \in \{0,1\}$, $\mathbf{C}^b := \mathsf{EncCt}(\vec{x}^b, \mathbf{M})$, $\mathbf{K} := \mathsf{EncKey}(\vec{y}, \vec{a})$. If $\vec{a}$ satisfies $\mathbf{M}$, then $\vec{x}^0 = \vec{x}^1$ and there is nothing to prove.

If $\vec{a}$ doesn't satisfies $\mathbf{M}$ then we show that $(w_0, \vec{v}, \mathsf{vect}(\mathbf{U}), \vec{w}_1, \ldots, \vec{w}_n)^\top (\mathbf{C}^b|\mathbf{K})$ is independent of b, using the fact that for all $\vec{a} \in \{0,1\}^n$, the following are identically distributed : $\{\vec{w}_i\}_{i\in[n]}$ and $\{\vec{w}_i + (a_i - 1)(\vec{v}|\mathbf{U})\mathbf{M}_i\}_{i\in[n]}$, with $\vec{w}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$ for all $i \in [n]$, $\vec{v} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^d$, and $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{d\times(n-1)}$. The leftmost distribution corresponds to

$$\begin{aligned}&(\mathbf{C}^b|\mathbf{K})^\top (w_0, \vec{v}, \mathsf{vect}(\mathbf{U}), \vec{w}_1, \ldots, \vec{w}_n) := \\ &\left((\vec{v}|\mathbf{U})\mathbf{M}_1 + \vec{w}_1, \ldots, (\vec{v}|\mathbf{U})\mathbf{M}_n + \vec{w}_n, \vec{v} + w_0\vec{x}^b\right)\end{aligned}$$

whereas the rightmost distribution corresponds to:

$$\left(a_1(\vec{v}|\mathbf{U})\mathbf{M}_1 + \vec{w}_1, \ldots, a_n(\vec{v}|\mathbf{U})\mathbf{M}_n + \vec{w}_n, \vec{v} + w_0\vec{x}^b\right).$$

Since $\vec{a}$ does not satisfies $\mathbf{M}$, we have: $(a_1(\vec{v}|\mathbf{U})\mathbf{M}_1, \ldots, a_n(\vec{v}|\mathbf{U})\mathbf{M}_n)$ is independent of $\vec{v}$, which can be used to mask $w_0\vec{x}^0$. Formally, that means the following are identically distributed: $(a_1(\vec{v}|\mathbf{U})\mathbf{M}_1 + \vec{w}_1, \ldots, a_n(\vec{v}|\mathbf{U})\mathbf{M}_n + \vec{w}_n, \vec{v} + w_0\vec{x}^0)$ and $(a_1(\vec{v}|\mathbf{U})\mathbf{M}_1 + \vec{w}_1, \ldots, a_n(\vec{v}|\mathbf{U})\mathbf{M}_n + \vec{w}_n, \vec{v} + w_0\vec{x}^1)$.