Adaptively Simulation-Secure Attribute-Hiding Predicate Encryption

Pratish Datta[1], Tatsuaki Okamoto[1], and Katsuyuki Takashima[2]

[1] NTT Secure Platform Laboratories
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
pratish.datta.yg@hco.ntt.co.jp, tatsuaki.okamoto@gmail.com
[2] Mitsubishi Electric
5-1-1 Ofuna, Kamakura, Kanagawa, 247-8501 Japan
Takashima.Katsuyuki@aj.MitsubishiElectric.co.jp

Abstract. This paper demonstrates how to achieve *simulation-based strong attribute hiding* against *adaptive* adversaries for *predicate encryption* (PE) schemes supporting *expressive* predicate families under *standard* computational assumptions in bilinear groups. Our main result is a *simulation-based adaptively strongly partially-hiding* PE (PHPE) scheme for predicates computing *arithmetic branching programs* (ABP) on *public* attributes, followed by an *inner-product predicate* on *private* attributes. This simultaneously generalizes attribute-based encryption (ABE) for boolean formulas and ABP's as well as strongly attribute-hiding PE schemes for inner products. The proposed scheme is proven secure for *any a priori bounded* number of ciphertexts and an *unbounded* (polynomial) number of decryption keys, which is the *best possible* in the simulation-based adaptive security framework. This directly implies that our construction also achieves *indistinguishability-based strongly partially-hiding* security against adversaries requesting an *unbounded* (polynomial) number of ciphertexts and decryption keys. The security of the proposed scheme is derived under (asymmetric version of) the *well-studied decisional linear* (DLIN) assumption. Our work resolves an open problem posed by Wee in TCC 2017, where his result was limited to the *semi-adaptive* setting. Moreover, our result advances the current state of the art in both the fields of simulation-based and indistinguishability-based strongly attribute-hiding PE schemes. Our main technical contribution lies in extending the strong attribute hiding methodology of Okamoto and Takashima [EUROCRYPT 2012, ASIACRYPT 2012] to the framework of simulation-based security and beyond inner products.

Keywords: predicate encryption, partially-hiding, simulation-based adaptive security, arithmetic branching programs, inner products

1 Introduction

Functional encryption (FE) is a new vision of modern cryptography that aims to overcome the potential limitation of the traditional encryption schemes, namely,

the all or nothing control over decryption capabilities. FE supports restricted decryption keys which enable decrypters to learn specific functions of encrypted messages, and nothing else. More precisely, a (public-key) FE scheme for a function family $\mathcal{F}$ involves a setup authority which holds a master secret key and publishes public system parameters. An encrypter uses the public parameters to encrypt its message M belonging to some supported message space $\mathbb{M}$, creating a ciphertext CT. A decrypter may obtain a private decryption key $\mathrm{SK}(F)$ for some function $F \in \mathcal{F}$ from the setup authority, provided the authority deems that the decrypter is entitled for that key. Such a decryption key $\mathrm{SK}(F)$ can be used to decrypt CT to recover $F(M)$, but nothing more about M.

The most intuitive security requirement for an FE scheme is *collusion resistance*, i.e., a group of decrypters cannot jointly retrieve any more information about an encrypted message beyond the union of what each of them is allowed to learn individually. This intuitive notion has been formalized by Boneh et al. [12] and O'Neill [37] in two distinct frameworks, namely, (a) *indistinguishability-based security* and (b) *simulation-based security*. The former stipulates that distinguishing encryptions of any two messages is infeasible for a group of colluders which do not have a decryption key that decrypts the ciphertext to distinct values. The latter, on the other hand, stipulates the existence of a polynomial-time simulator that given $F_1(M), \ldots, F_{q_{\mathrm{KEY}}}(M)$ for any message $M \in \mathbb{M}$ and functions $F_1, \ldots, F_{q_{\mathrm{KEY}}} \in \mathcal{F}$, outputs the view of the colluders which are given an encryption of M together with decryption keys for $F_1, \ldots, F_{q_{\mathrm{KEY}}}$. Both of the above notions can be further refined, depending on how the queries of the adversary to the decryption key generation and encryption oracles depend on one another as well as on the public parameters of the system, as *adaptive* vs *semi-adaptive* vs *selective*. Boneh et al. [12] and O'Neill [37] showed that in general, simulation-based security provides a stronger guarantee than indistinguishability-based security, i.e., simulation-based security of some kind, e.g., adaptive, semi-adaptive, or selective, implies indistinguishability-based security of the same kind; but the converse does not hold in general. In fact, Boneh et al. pointed out that indistinguishability-based security is vacuous for certain circuit families, which indicates that we should opt for simulation-based security whenever possible. On the other hand, it is known that while security for single and multiple ciphertexts are equivalent in the indistinguishability-based setting [12], this is not the case in the simulation-based setting [12,9,3,16]. In particular, it has been demonstrated by Boneh et al. [12] that in the adaptive or semi-adaptive simulation-based setting, where the adversary is allowed to make decryption key queries even after receiving the queried ciphertexts, achieving security for an unbounded number of ciphertexts is impossible.

An important subclass of FE is *predicate encryption* (PE). In recent years, with the rapid advancement of Internet communication and cloud technology, there has been an emerging trend among individuals and organizations to outsource potentially sensitive private data to external untrusted servers, and to perform selective computations on the outsourced data by remotely querying the server at some later point in time, or to share specific portions of the out-

sourced data to other parties of choice. PE is an indispensable tool for performing such operations on outsourced sensitive data without compromising the confidentiality of the data.

Consider a predicate family $R = \{R(Y, \cdot) : \mathcal{X} \to \{0,1\} \mid Y \in \mathcal{Y}\}$, where $\mathcal{X}$ and $\mathcal{Y}$ are two collections of indices or attributes. In a PE scheme for some predicate family R, the associated message space $\mathbb{M}$ is of the form $\mathcal{X} \times \mathcal{M}$, where $\mathcal{M}$ contains the actual payloads. The functionality F_{R_Y} associated with a predicate $R(Y, \cdot) \in R$ is defined as $F_{R_Y}(X, \mathsf{msg}) = \mathsf{msg}$ if $R(Y, X) = 1$, or in other words, Y is authorized for X, and $F_{R_Y}(X, \mathsf{msg}) = \perp$ (a special empty string) if $R(Y, X) = 0$, or in other words, Y is not authorized for X for all $(X, \mathsf{msg}) \in \mathbb{M} = \mathcal{X} \times \mathcal{M}$.

The standard security notion for FE described above, when adopted in the context of PE, stipulates that recovering the payload from a ciphertext generated with respect to some attribute $X \in \mathcal{X}$ should be infeasible for a group of colluders none of which possesses a decryption key corresponding to an attribute authorized for X, also referred to as an authorized decryption key; and moreover, the ciphertext should conceal X from any group of colluders, even those in possession of authorized decryption keys. In the context of PE, this security notion is referred to as *strongly attribute-hiding* security. A weakening of the above notion, called *weakly attribute-hiding* security requires that X should only remain hidden to colluders in possession of unauthorized keys. An even weaker notion, which only demands the payload to remain hidden to colluders with unauthorized keys, is known as *payload-hiding* security, and a payload-hiding PE scheme is often referred to as an *attribute-based encryption* (ABE) scheme in the literature.

Over the last decade, a long sequence of works have developed extremely powerful techniques for realizing indistinguishability-based ABE and weakly attribute-hiding PE schemes supporting more and more expressive predicate families under well-studied computational assumptions in bilinear groups and lattices, culminating into schemes that can now support general polynomial-size circuits [22,30,33,34,26,31,41,7,14,20,11,19]. However, very little is known for strongly attribute-hiding PE schemes, even in the indistinguishability-based setting. The situation is even worse when security against an unbounded (polynomial) number of authorized-key-possessing colluders under standard computational assumption is considered. In fact, until very recently, the known candidates were restricted to only inner products or even simpler predicates [13, 29, 32, 34, 36, 16], out of which the schemes designed in the more efficient and secure prime order bilinear groups being only the works of Okamoto and Takashima [32, 34, 36]. One big reason for this state of the art is that unlike payload-hiding or weakly attribute-hiding, for proving strongly attribute-hiding security, one must argue about an adversary that gets hold of authorized decryption keys, something cryptographers do not have a good understanding of so far. Moreover, there are indeed reasons to believe that constructing strongly attribute-hiding PE schemes for sufficiently expressive predicate classes such as NC^1 under standard computational assumptions could be very difficult. In fact, it is known that a strongly

attribute-hiding PE scheme for NC^1 predicates, even in the weakest selective setting, can lead all the way to indistinguishability obfuscation (IO) for general circuits, the new holy grail of modern cryptography [5,10,6]. In view of this state of affairs, it is natural to ask the following important question:

Can we realize "the best of both worlds", i.e., can we design PE *scheme for some sufficiently expressive predicate family (e.g.,* NC^1*) that is secure against an unbounded (polynomial) number of colluders under standard computational assumptions (without* IO*), such that the strongly attribute-hiding guarantee holds for a limited segment (e.g., one belonging to some subclass of* NC^1*) of each predicate in the predicate family?*

Towards answering this question, in TCC 2017, Wee [40] put forward a new PE scheme for an NC^1 predicate family in bilinear groups of prime order that is secure against an unbounded (polynomial) number of colluders under the well-studied *k-linear* (*k*-LIN) assumption, where the strongly attribute-hiding property is achieved only for an inner product evaluating segment of each predicate of the predicate class. More precisely, in his proposed PE system, the ciphertext attribute set $\mathcal{X}$ is given by $\mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ for some finite field $\mathbb{F}_q$ and $n', n \in \mathbb{N}$, while the decryption key attribute set $\mathcal{Y}$ is given by the function family $\mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$. Any function $f \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ operates on a pair $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ by first computing n arithmetic branching programs (ABP) $f_1, \ldots, f_n : \mathbb{F}_q^{n'} \to \mathbb{F}_q$ on $\vec{x}$ to obtain a vector $(f_1(\vec{x}), \ldots, f_n(\vec{x})) \in \mathbb{F}_q^n$, and then evaluating the inner product of the computed vector and $\vec{z}$. The predicate family $R^{\text{ABP}\circ\text{IP}}$ associated with the PE scheme is defined as $R^{\text{ABP}\circ\text{IP}} = \{R^{\text{ABP}\circ\text{IP}}(f, (\cdot,\cdot)) : \mathbb{F}_q^{n'} \times \mathbb{F}_q^n \to \{0,1\} \mid f \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}\}$, where $R^{\text{ABP}\circ\text{IP}}(f, (\vec{x}, \vec{z})) = 1$ if $f(\vec{x}, \vec{z}) = 0$, and 0 if $f(\vec{x}, \vec{z}) \neq 0$ for any $f \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ and $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$. The security property of Wee's PE scheme guarantees that other than hiding the payload, a ciphertext generated for some attribute pair $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ also conceals the attribute $\vec{z}$ (but not the attribute $\vec{x}$). Moreover, the concealment of the attribute $\vec{z}$ is strong, i.e., even against colluders possessing authorized keys. Wee termed this security notion as *strongly partially-hiding* security, while the attributes $\vec{x} \in \mathbb{F}_q^{n'}$ and $\vec{z} \in \mathbb{F}_q^n$ as the *public* and *private* attributes respectively.

This PE scheme simultaneously generalizes ABE for boolean formulas and ABP's, as well as strongly attribute-hiding inner-product PE (IPE). For instance, unlike standard IPE schemes, where an inner-product predicate is evaluated between the (private) attribute vector $\vec{z}$ associated with a ciphertext and the attribute vector $\vec{y}$ hardwired within a decryption key, this PE scheme evaluates inner-product predicate between $\vec{z}$ and $\vec{y}$ obtained as the result of complicated ABP computations on a public attribute string $\vec{x}$, which is now associated in addition to the private attribute vector $\vec{z}$ with the ciphertext. This in turn means that this PE scheme can be deployed in richer variants of the applications captured by IPE schemes. For example, it is well-known that inner-product predicates can be used to evaluate conjunctive comparison predicates of the form $R^{\text{COMP}}((c_1, \ldots, c_n), (z_1, \ldots, z_n)) = \bigwedge_{j \in [n]} [z_j \geq c_j]$, where c_j's and z_j's lie in polynomial-size domains [13]. In case of standard IPE schemes, $c_1, \ldots, c_n$ are

fixed constants which are specified within the decryption key. On the contrary, in case of a PE scheme for $R^{\text{ABP}\circ\text{IP}}$, we can carry out more complex computation, where instead of being fixed constants, $c_1, \ldots, c_n$ can be derived as the outputs of ABP evaluations on public ciphertext attributes. Of course, fixed $c_1, \ldots, c_n$ is a special case of this more expressive computation, since one can have ABP's that ignore the public ciphertext attributes, and simply output hardwired constants. Similarly, standard IPE schemes can be employed for evaluating polynomials with constant coefficients, where the coefficients are specified within the decryption keys [29]. In contrast, in case of a PE scheme for $R^{\text{ABP}\circ\text{IP}}$, the polynomial coefficients can be generated as outputs of ABP computations on public ciphertext attributes.

Partially-hiding PE (PHPE) schemes for similar type of predicate families were considered in [19, 2] in the lattice setting, and those PHPE schemes are in fact capable of evaluating general polynomial-size circuits, as opposed to ABP's in Wee's construction, over public ciphertext attributes prior to evaluating inner-product predicates over private ciphertext attributes. However, those constructions are either only weakly partially-hiding, i.e., the security of the private attributes of the ciphertexts are only guaranteed against unauthorized colluders [19], or strongly partially-hiding against a priori bounded number of authorized colluders [2]. In contrast, Wee's PHPE scheme is strongly partially-hiding against an unbounded (polynomial) number of authorized colluders. Another strong aspect of the PHPE construction of Wee is that its security is proven in the (unbounded) simulation-based framework [3], while except [16], all prior PE constructions with strongly attribute-hiding security against an unbounded (polynomial) number of authorized colluders were proven in the weaker indistinguishability-based framework.

However, the PHPE scheme proposed by Wee [40] only achieves semi-adaptive security [15], i.e., against an adversary that is restricted to submit its ciphertext queries immediately after viewing the public parameters, and can make decryption key queries only after that. While semi-adaptive security seems somewhat stronger, it has recently been shown by Goyal et al. [21] that it is essentially equivalent to the selective security, the weakest notion of security in which the adversary is bound to declare its ciphertext queries even before the system is setup. Their result also indicates that the gap between semi-adaptive and adaptive security, the strongest and most reasonable notion in which the adversary is allowed to make ciphertext and decryption key queries at any point during the security experiment, is in fact much wider than was previously thought. While Ananth et al. [4] have demonstrated how to generically transform an FE scheme that supports arbitrary polynomial-size circuits from selective security to one that achieves adaptive security, their conversion does not work for ABE or PE schemes which fall below this threshold in functionality. In view of this state of affairs, it is interesting to explore *whether it is possible to construct an efficient adaptively simulation-secure strongly partially-hiding* PE *scheme for the predicate family* $R^{\text{ABP}\circ\text{IP}}$ *that is secure against an unbounded* (*polynomial*) *number of colluders under well-studied computational assumption.* Note that while several

impossibility results exist against the achievability of simulation-based security in certain settings [12,9,3,16], those results do not overrule the existence of such a construction, provided of course we bound the number of allowed ciphertext queries by the adversary. In fact, Wee has posed the realization of such a PHPE construction as an open problem in his paper [40].

Our Contributions

In this paper, we resolve the above open problem. Specifically, our main result is a PE scheme for the predicate family $R^{\mathsf{ABP} \circ \mathsf{IP}}$ that achieves *simulation-based adaptively strongly partially hiding* security against adversaries making *any a priori bounded* number of ciphertext queries while requesting an *unbounded* (polynomial) number of decryption keys both before and after the ciphertext queries, which is the *best* one could hope for in the simulation-based framework when the adversary is allowed to make decryption key queries even after making the ciphertext queries [12]. From the relation between simulation-based and indistinguishability-based security as well as that between single and multiple ciphertext security in the indistinguishability-based setting as mentioned above, it is immediate that the proposed scheme is also *adaptively strongly partially-hiding* in the *indistinguishability-based* framework against adversaries making an *unbounded* number of queries to both the encryption and the decryption key generation oracles. Thus, our work advances the state of the art in both the fields of simulation-based and indistinguishability-based strongly attribute-hiding PE schemes. Our construction is built in *asymmetric* bilinear groups of *prime* order. The security of our PHPE scheme is derived under the *simultaneous external decisional linear* (SXDLIN) assumption [1,38], which is a natural extension of the *well-studied decisional linear* (DLIN) assumption in asymmetric bilinear group setting, and as noted in [1], the two assumptions are in fact *equivalent* in the generic bilinear group model. Nevertheless, our scheme can be readily generalized to one that is secure under the k-LIN assumption.

Similar to [40], we only consider security against a single ciphertext query for the construction presented in this paper to keep the exposition simple. However, we explain in Remark 3.1 how our techniques can be readily extended to design a PHPE scheme that is secure for any a priori bounded number of ciphertexts. Following [16], here we present our construction in the attribute-only mode (i.e., without any actual payload). However, in the full version of this paper we also provide a key-encapsulation mechanism (KEM) version (i.e., one that uses a symmetric session key as the payload) of our scheme similar to [40]. For the attribute-only version, we design a simulator that runs in polynomial time, and thus this version of our scheme is secure in the standard simulation-based security framework. On the other hand, for the KEM version, similar to Wee [40], our simulator needs to perform a brute force discrete log computation, and thus requires super-polynomial (e.g., sub-exponential) computational power. Nonetheless, this is still stronger than the indistinguishability-based framework [3, 40].

In terms of efficiency, our PHPE scheme is fairly practical. The length of ciphertexts and decryption keys of our scheme grow linearly with the total length of the associated attribute strings and the ABP-size of the associated functions

respectively. This is the same as that of [40] except for a constant blow-up, which is common in the literature for semi-adaptive vs adaptive security. Moreover, asymmetric bilinear groups of prime order, which are used for implementing our scheme, are now considered to be both faster and more secure in the cryptographic community following the recent progress in analysing bilinear groups of composite order [17, 23] and symmetric bilinear groups instantiated with elliptic curves of small characteristics [8, 18, 27, 28].

As a byproduct of our main result, we also obtain the *first simulation-based* adaptively strongly attribute-hiding IPE scheme in asymmetric bilinear groups of prime order under the SXDLIN assumption. The only prior simulation-based strongly attribute-hiding IPE scheme, also due to Wee [40], only achieves semi-adaptive security.

On the technical side, our approach is completely different from that of Wee [40]. More precisely, Wee's technique consists of two steps, namely, first building a private-key scheme, and then bootstrapping it to a public-key one by applying a private-key to public-key compiler similar to [41, 14], built on Water's dual system encryption methodology [39]. In contrast, we *directly* construct our scheme in the public-key setting by extending the technique of Okamoto and Takashima [32, 34, 36], a more sophisticated methodology than the dual system encryption originally developed for designing adaptively strongly attribute-hiding IPE schemes in the indistinguishability-based setting, to the scenario of simulation-based adaptively strongly attribute-hiding security for a much expressive predicate class. Also, in order to incorporate the information of the session keys within the ciphertexts in the KEM version of our scheme, which is presented in the full version of this paper, we adopt an idea along the lines of the works of Okamoto and Takashima [32, 34, 36], that deviates from that of Wee [40]. Thus, our work further demonstrates the power of the technique introduced by Okamoto and Takashima [32, 34, 36] in achieving very strong security for highly expressive predicate families. We also believe that our work will shed further light on one of the longstanding questions of modern cryptography:
What is the most expressive function or predicate family for which it is possible to construct FE *or strongly attribute-hiding* PE *schemes with adaptive security against adversaries making an unbounded* (*polynomial*) *number of decryption key queries under standard computational assumptions?*

Overview of Our Techniques

We now proceed to explain the key technical ideas underlying our construction. For simplicity, here we will only deal with the IPE scheme, which is a special case of our PHPE construction for $R^{\text{ABP}\circ\text{IP}}$. The proposed PHPE scheme for $R^{\text{ABP}\circ\text{IP}}$ is obtained via a more sophisticated application of the techniques described in this section, and is formally presented in full details in the sequel.

In this overview, we will consider IPE in the attribute-only mode. For IPE, the ciphertext attribute set $\mathcal{X} = \mathbb{F}_q^n$, the decryption key attribute set $\mathcal{Y} = \mathbb{F}_q^n$ for some finite field $\mathbb{F}_q$ and $n \in \mathbb{N}$, and the predicate family is given by $R^{\text{IP}} = \{R^{\text{IP}}(\vec{y}, \cdot) : \mathbb{F}_q^n \to \{0,1\} \mid \vec{y} \in \mathbb{F}_q^n\}$, where $R^{\text{IP}}(\vec{y}, \vec{z}) = 1$ if $\vec{z} \cdot \vec{y} = 0$, and 0, if $\vec{z} \cdot \vec{y} \neq 0$ for any $\vec{z}, \vec{y} \in \mathbb{F}_q^n$. Observe that the predicate family R^{IP} is subclass

of the predicate family $R^{\text{ABP}\circ\text{IP}}$, where we set $n' = 0$, and the component ABP's $f_1, \ldots, f_n$ of a function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$ to simply output hardwired constants. In the attribute-only mode, a ciphertext is associated with only a vector $\vec{z} \in \mathbb{F}_q^n$ but no payload, and decryption with a key for some vector $\vec{y} \in \mathbb{F}_q^n$ only reveals the predicate, i.e., whether $\vec{z} \cdot \vec{y} = 0$ or not, but not the exact value of $\vec{z} \cdot \vec{y}$.

Just like [32,34,36], we make use of the machinery of the *dual pairing vector spaces* (DPVS) [35,33]. A highly powerful feature of DPVS is that one can completely or partially hide a linear subspace of the whole vector space by concealing the basis of that subspace or the basis of its dual from the public parameters respectively. In DPVS-based constructions, a pair of mutually dual vector spaces $\mathbb{V}_1$ and $\mathbb{V}_2$, along with a bilinear pairing $e : \mathbb{V}_1 \times \mathbb{V}_2 \to \mathbb{G}_T$ constructed from a standard bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ of prime order q is used. Typically a pair of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ of the vector spaces $(\mathbb{V}_1, \mathbb{V}_2)$ are generated during setup, using random linear transformations, and a portion of $\mathbb{B}$, say $\widehat{\mathbb{B}}$, is used as the public parameters. Thus, the corresponding segment of $\mathbb{B}^*$, say $\widehat{\mathbb{B}}^*$ remains partially hidden (its dual subspace is disclosed), while the part $\mathbb{B}\backslash\widehat{\mathbb{B}}$ of the basis $\mathbb{B}$ and the corresponding portion $\mathbb{B}^*\backslash\widehat{\mathbb{B}}^*$ of the basis $\mathbb{B}^*$ remain completely hidden to an adversary that is given the public parameters, ciphertexts, and decryption keys. This provides a strong framework for various kinds of information-theoretic tricks in the public-key setting by exploiting various nice properties of linear transformations.

In the proposed IPE scheme, we consider a $(4n+1)$-dimensional DPVS. During setup, we generate a random pair of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$, and use as the public parameters the subset $\widehat{\mathbb{B}}$ consisting of the first n and the last $n+1$ vectors of the basis $\mathbb{B}$, while as the master secret key the corresponding portion of the dual basis $\mathbb{B}^*$. Thus, the linear subspaces spanned by the remaining $2n$ vectors of the bases $\mathbb{B}$ and $\mathbb{B}^*$ are kept completely hidden. Intuitively, we will use the first n-dimensional subspaces of these $2n$-dimensional subspaces for simulating the post-ciphertext decryption key queries, while the latter n-dimensional subspaces for simulating the pre-ciphertext decryption key queries in the ideal experiment. A ciphertext for some vector $\vec{z} \in \mathbb{F}_q^n$ in the proposed scheme has the form $\text{CT} = \boldsymbol{c}$ such that

$$\boldsymbol{c} = (\omega\vec{z}, \vec{0}^n, \vec{0}^n, \vec{0}^n, \varphi)_{\mathbb{B}},$$

where $\omega, \varphi \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and $(\vec{v})_{\mathbb{W}}$ represents the linear combination of the elements of $\mathbb{W}$ with the entries of $\vec{v}$ as coefficients for any $\vec{v} \in \mathbb{F}_q^n$ and any basis $\mathbb{W}$ of DPVS. Similarly, a decryption key corresponding to some vector $\vec{y} \in \mathbb{F}_q^n$ is given by $\text{SK}(\vec{y}) = (\vec{y}, \boldsymbol{k})$ such that

$$\boldsymbol{k} = (\zeta\vec{y}, \vec{0}^n, \vec{0}^n, \vec{\kappa}, 0)_{\mathbb{B}^*},$$

where $\zeta \xleftarrow{\mathsf{U}} \mathbb{F}_q$ and $\vec{\kappa} \xleftarrow{\mathsf{U}} \mathbb{F}_q^n$. Decryption computes $e(\boldsymbol{c}, \boldsymbol{k})$ to obtain $g_T^{\omega\zeta(\vec{z}\cdot\vec{y})} \in \mathbb{G}_T$, which equals to the identity element of the group $\mathbb{G}_T$ if $\vec{z} \cdot \vec{y} = 0$, and a uniformly random element of $\mathbb{G}_T$ if $\vec{z} \cdot \vec{y} \neq 0$. Observe that this IPE construction is essentially the same as that presented by Okamoto and Takashima in [32]. However, they only proved the strongly attribute-hiding security of this construction in the indistinguishability-based framework, while we prove this construction to

be strongly attribute-hiding in the simulation-based framework, by extending their techniques. Let us start with describing our simulation strategy.

In the semi-adaptive case, as considered in [40], the simulation strategy is relatively simple. In fact, for designing an IPE in the semi-adaptive setting, only a $(3n+1)$-dimensional DPVS with n-dimensional hidden subspace would suffice. Note that in the semi-adaptive setting, the adversary is restricted to make the ciphertext query immediately after seeing the public parameters, and there is no pre-ciphertext decryption key query. So, in the semi-adaptive setting, when the adversary makes a ciphertext query, the simulator has no constraint arising from the pre-ciphertext queries of the adversary, and can simply simulate the ciphertext as $\mathrm{CT} = \boldsymbol{c}$ such that

$$\boldsymbol{c} = (\vec{0}^n, (\vec{0}^{n-1}, \tau), \vec{0}^n, \varphi)_{\mathbb{B}},$$

where $\tau, \varphi \xleftarrow{\mathsf{U}} \mathbb{F}_q$, i.e., the simulator puts nothing in the subspace spanned by the public segment of the basis $\mathbb{B}$, and merely puts a random value in a one-dimensional subspace spanned by the hidden segment of the basis. Later, when the adversary queries a decryption key for some vector $\vec{y} \in \mathbb{F}_q^n$, the simulator gets $\vec{y}$ along with the inner product relation of $\vec{y}$ with $\vec{z}$, and the simulator can simply hardwire this information in the corresponding hidden subspace of the decryption key. More precisely, it can simply generate the decryption key as $\mathrm{SK}(\vec{y}) = (\vec{y}, \boldsymbol{k})$ such that

$$\boldsymbol{k} = (\zeta\vec{y}, (\vec{\eta}, \nu), \vec{\kappa}, 0)_{\mathbb{B}^*},$$

where $\zeta \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\vec{\eta} \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n-1}$, $\vec{\kappa} \xleftarrow{\mathsf{U}} \mathbb{F}_q^n$, and $\nu = 0$ if $\vec{z}\cdot\vec{y} = 0$, and $\nu \xleftarrow{\mathsf{U}} \mathbb{F}_q$ if $\vec{z}\cdot\vec{y} \neq 0$. Observe that when the simulated ciphertext is decrypted using this simulated decryption key, one obtains the identity element of $\mathbb{G}_T$, or a random element of $\mathbb{G}_T$ according as the inner product relation is satisfied or not, i.e., decryption correctness clearly holds. At this point, please note that the simulator cannot put anything in the subspace of the ciphertext corresponding to the public segment of $\mathbb{B}$, since it must put the actual attribute vectors in the corresponding dual subspace of the decryption keys to ensure correct decryption with other honestly generated ciphertexts.

In the adaptive setting, the situation is much more complex, and we need a $(4n+1)$-dimensional DPVS with $2n$-dimensional hidden subspace. Now, the simulator should also correctly simulate the pre-ciphertext decryption key queries of the adversary. The difference between the pre-ciphertext and post-ciphertext decryption key queries is that unlike the post-ciphertext ones, the information about whether the inner product relation between the associated attribute vector and the attribute vector $\vec{z}$ corresponding to the ciphertext query of the adversary is not supplied when the decryption key is queried. In fact, $\vec{z}$ is not even declared at that time. On the contrary, the information about predicate satisfaction for all the pre-ciphertext decryption key vectors become available to the simulator when the ciphertext query is made by the adversary. The main hurdle for the simulator is to compactly embed this huge amount of information (note that we are considering an unbounded number of pre-ciphertext decryption key queries) in the simulated ciphertext, so that when the simulated ciphertext is decrypted

using any pre-ciphertext decryption key, one should get the proper information about predicate satisfaction.

Towards overcoming this difficulty, we observe that it has already been demonstrated by O'Neill [37] that the inner-product predicate family is *pre-image samplable*, i.e., given a set of vectors and their inner-product relation with another fixed vector (but not the vector itself), one can efficiently sample a vector that satisfies all those inner-product relations with high probability. To simulate the ciphertext queried by the adversary, our simulator does exactly this, i.e., it samples a vector $\vec{s} \in \mathbb{F}_q^n$ that has the same inner-product relations as the original queried ciphertext attribute vector $\vec{z}$ with all the attribute vectors corresponding to the pre-ciphertext decryption key queries of the adversary. However, $\vec{s}$ may not have the same inner-product relation as $\vec{z}$ with the attribute vectors corresponding to the post-ciphertext decryption key queries. Therefore, it cannot be embedded in the hidden subspace of the ciphertext devoted for handling the post-ciphertext decryption key queries. Therefore, the simulator needs another n-dimensional subspace to embed $\vec{s}$. Thus, the simulator simulates the queried ciphertext as $\mathrm{CT} = \boldsymbol{c}$ such that

$$\boldsymbol{c} = (\vec{0}^n, (\vec{0}^{n-1}, \tau), \theta\vec{s}, \vec{0}^n, \varphi)_{\mathbb{B}},$$

where $\theta \xleftarrow{\mathsf{U}} \mathbb{F}_q$. On the other hand, it simulates a decryption key corresponding to some vector $\vec{y} \in \mathbb{F}_q^n$ as $\mathrm{SK}(\vec{y}) = (\vec{y}, \boldsymbol{k})$ such that

$$\boldsymbol{k} = \begin{cases} (\zeta\vec{y}, \vec{0}^n, \widehat{\zeta}\vec{y}, \vec{\kappa}, 0)_{\mathbb{B}^*} & \text{(pre-ciphertext)}, \\ (\zeta\vec{y}, (\vec{\eta}, \nu), \vec{0}^n, \vec{\kappa}, 0)_{\mathbb{B}^*} & \text{(post-ciphertext)}, \end{cases}$$

where $\zeta, \widehat{\zeta} \xleftarrow{\mathsf{U}} \mathbb{F}_q$.

Here, we would like to emphasize that while we make use of the pre-image samplability property introduced by O'Neill [37] to design our simulator, our result is not a mere special case of the result that O'Neill obtained using that property. Specifically, O'Neill showed that indistinguishability-based and simulation-based security notions are equivalent in case of FE schemes for function families which are pre-image samplable, provided the adversary is constrained from making any decryption key query after making a ciphertext query. His result does not apply if the adversary is allowed to make decryption key queries even after making ciphertext queries, as is the case in this paper. Moreover, note that there is no known PE scheme for the predicate family $R^{\mathrm{ABP}\circ\mathrm{IP}}$, the actual focus of this paper, even with indistinguishability-based strongly partially-hiding security against adversaries that are allowed to make decryption key queries prior to making ciphertext queries.

Let us continue with the technical overview. It remains to argue that the above simulated forms of ciphertexts and decryption keys are indistinguishable from their real forms. In order to accomplish these changes, we design elaborate hybrid transitions over different forms of ciphertext and decryption keys. In fact, the $2n$-dimensional hidden subspace not only allows us to simulate the pre-ciphertext and post-ciphertext queries differently, but are also crucially leveraged to realize the various forms of ciphertext and decryption keys throughout our hybrid transitions. The hybrid transitions are alternatively computational

and information-theoretic. Also, note that not only our simulation strategy for pre-ciphertext and post-ciphertext decryption key queries are different, rather in our hybrid transitions, we handle the pre-ciphertext and post-ciphertext decryption key queries differently, and thereby achieve a security loss that is only proportional to the number of pre-ciphertext decryption key queries.

We start by changing the pre-ciphertext decryption keys to their simulated form. For making these changes, we use the first n-dimensional subspace of the $2n$-dimensional hidden subspace as the *working space*, where we generate the simulated components, and the next n-dimensional subspace as the *storing space*, where we transfer and store the simulated components once they are generated. Note that in the simulated pre-ciphertext decryption keys, the additional simulated components are placed in the second n-dimensions subspace of the $2n$-dimensional hidden subspace. For the hybrid transitions of this part, we make use of the first two of the three types of information-theoretic tricks, namely, Type I, Type II, and Type III introduced in [32,34,36], in conjunction with the three types of computational tricks based on the SXDLIN assumption also used in those works. The Type I trick is to apply a linear transformation inside a hidden subspace on the ciphertext side, while the more complex Type II trick is to apply a linear transformation inside a hidden subspace on the ciphertext side preserving the predicate relation with the entries in the corresponding dual subspace of a specific decryption key.

After the transformation of the pre-ciphertext queries is completed, we turn our attention to vanish the component of the ciphertext in the subspace spanned by the public portion of the basis $\mathbb{B}$. For doing this, we apply one of the three computational tricks followed by a Type III information-theoretic trick, which amounts to applying a linear transformation across a hidden and a partially public subspace on both the ciphertext and decryption key sides. While, this enables us to achieve our target for the ciphertext, the forms of the pre-ciphertext decryption keys get distorted. To bring the pre-ciphertext decryption keys to their correct simulated form, we then apply an extension of one of the computational tricks mentioned above.

Once the component in the public subspace of the ciphertext is vanished and pre-ciphertext decryption keys are brought back to their correct simulated form, we turn our attention to the post-cipertext decryption keys. Note that the Type III trick applied for the ciphertext, has already altered the forms of the post-ciphertext queries to something else. Starting with these modified forms, we apply a more carefully crafted variant of the Type II information-theoretic trick, followed by another computational trick based on the SXDLIN assumption to alter the post-ciphertext decryption keys to their simulated forms. This step is reminiscent of the *one-dimensional localization of the inner-product values* used in [36]. This step also alters the ciphertext to its simulated form. At this point we arrive at the simulated experiment, and our security analysis gets complete.

2 Preliminaries

In this section we present the backgrounds required for the rest of this paper.

2.1 Notations

Let $\lambda \in \mathbb{N}$ denotes the security parameter and 1^λ be its unary encoding. Let $\mathbb{F}_q$ for any prime $q \in \mathbb{N}$, denotes the finite field of integers modulo q. For $d \in \mathbb{N}$ and $c \in \mathbb{N} \cup \{0\}$ (with $c < d$), we let $[d] = \{1, \ldots, d\}$ and $[c, d] = \{c, \ldots, d\}$. For any set Z, $z \xleftarrow{\mathsf{U}} Z$ represents the process of uniformly sampling an element z from the set Z, and $\sharp Z$ signifies the size or cardinality of Z. For a probabilistic algorithm $\mathcal{U}$, we denote by $\Pi = \mathcal{U}(\Theta; \Phi)$ the output of $\mathcal{U}$ on input Θ with the content of the random tape being Φ, while by $\Pi \xleftarrow{\mathsf{R}} \mathcal{U}(\Theta)$ the process of sampling Π from the output distribution of $\mathcal{U}$ with a uniform random tape on input Θ. Similarly, for any deterministic algorithm $\mathcal{V}$, we write $\Pi = \mathcal{V}(\Theta)$ to denote the output of $\mathcal{V}$ on input Θ. We use the abbreviation PPT to mean probabilistic polynomial-time. We assume that all the algorithms are given the unary representation 1^λ of the security parameter λ as input and will not write 1^λ explicitly as input of the algorithms when it is clear from the context. For any finite field $\mathbb{F}_q$ and $d \in \mathbb{N}$, let $\vec{v}$ denotes the (row) vector $(v_1, \ldots, v_d) \in \mathbb{F}_q^d$, where $v_i \in \mathbb{F}_q$ for all $i \in [d]$. The all zero vectors in $\mathbb{F}_q^d$ will be denoted by $\vec{0}^d$. For any two vectors $\vec{v}, \vec{w} \in \mathbb{F}_q^d$, $\vec{v} \cdot \vec{w}$ stands for the inner product of the vectors $\vec{v}$ and $\vec{w}$, i.e., $\vec{v} \cdot \vec{w} = \sum_{i \in [d]} v_i w_i \in \mathbb{F}_q$. For any multiplicative cyclic group $\mathbb{G}$ of order q and any generator $g \in \mathbb{G}$, let $\boldsymbol{v}$ represents a d-dimensional (row) vector of group elements, i.e., $\boldsymbol{v} = (g^{v_1}, \ldots, g^{v_d}) \in \mathbb{G}^d$ for some $d \in \mathbb{N}$, where $\vec{v} = (v_1, \ldots, v_d) \in \mathbb{F}_q^d$. We use $\boldsymbol{M} = (m_{i,k})$ to represent a matrix with entries $m_{i,k} \in \mathbb{F}_q$. By $\boldsymbol{M}^\intercal$ we will signify the transpose of the matrix $\boldsymbol{M}$. The determinant of a matrix $\boldsymbol{M}$ is denoted by $\det(\boldsymbol{M})$. Let $\mathsf{GL}(d, \mathbb{F}_q)$ denotes the set of all $d \times d$ invertible matrices over $\mathbb{F}_q$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is said to be *negligible* if for every $c \in \mathbb{N}$, there exists $T \in \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$ with $\lambda > T$, $|\mathsf{negl}(\lambda)| < 1/\lambda^c$.

2.2 Arithmetic Branching Programs

A *branching program* (BP) Γ is defined by a 5-tuple $\Gamma = (V, E, v_0, v_1, \phi)$, where (V, E) is a directed acyclic graph, $v_0, v_1 \in V$ are two special vertices called the source and the sink respectively, and ϕ is a labeling function for the edges in E. An *arithmetic branching program* (ABP) Γ over a finite field $\mathbb{F}_q$ computes a function $f : \mathbb{F}_q^d \to \mathbb{F}_q$ for some $d \in \mathbb{N}$. In this case, the labeling function ϕ assigns to each edge in E either a degree one polynomial function in one of the input variables with coefficients in $\mathbb{F}_q$ or a constant in $\mathbb{F}_q$. Let $\wp$ be the set of all v_0-v_1 paths in Γ. The output of the function f computed by the ABP Γ on some input $\vec{w} = (w_1, \ldots, w_d) \in \mathbb{F}_q^d$ is defined as $f(\vec{w}) = \sum_{P \in \wp} \left[\prod_{e \in P} \phi(e)|_{\vec{w}} \right]$, where for any $e \in E$, $\phi(e)|_{\vec{w}}$ represents the evaluation of the function $\phi(e)$ at $\vec{w}$. We refer to $\sharp V + \sharp E$ as the size of the ABP Γ. Ishai and Kushilevitz [25, 24] showed how to relate the computation performed by an ABP to the computation of the determinant of a matrix.

Lemma 2.1 ([24]): *Given an* ABP $\Gamma = (V, E, v_0, v_1, \phi)$ *computing a function* $f : \mathbb{F}_q^d \to \mathbb{F}_q$*, we can efficiently and deterministically compute a function* $\boldsymbol{L}$

mapping an input $\vec{w} \in \mathbb{F}_q^d$ *to a* $(\sharp V - 1) \times (\sharp V - 1)$ *matrix* $\boldsymbol{L}(\vec{w})$ *over* $\mathbb{F}_q$ *such that the following holds:*

- $\det(\boldsymbol{L}(\vec{w})) = f(\vec{w})$.
- *Each entry of* $\boldsymbol{L}(\vec{w})$ *is either a degree one polynomial in a single input variable* w_i $(i \in [d])$ *with coefficients in* $\mathbb{F}_q$ *or a constant in* $\mathbb{F}_q$.
- $\boldsymbol{L}(\vec{w})$ *contains only* -1*'s in the second diagonal, i.e., the diagonal just below the main diagonal, and 0's below the second diagonal.*

Specifically, $\boldsymbol{L}$ *is obtained by removing the column corresponding to* v_0 *and the row corresponding to* v_1 *in the matrix* $\boldsymbol{A}_\Gamma - \boldsymbol{I}$*, where* $\boldsymbol{A}_\Gamma$ *is the adjacency matrix for* Γ *and* $\boldsymbol{I}$ *is the identity matrix.*

Note that there is a linear-time algorithm that converts any Boolean formula, Boolean branching program, or arithmetic formula to an ABP with a constant blow-up in the representation size. Thus, ABP's can be viewed as a stronger computational model than all the others mentioned above.

2.3 The Function Family $\mathcal{F}_{\mathsf{ABP}\circ\mathsf{IP}}^{(q,n',n)}$ and the Algorithm PGB

Here, we formally describe the function family $\mathcal{F}_{\mathrm{ABP}\circ\mathrm{IP}}^{(q,n',n)}$ which our PHPE scheme supports, and an algorithm PGB for this function class that will be used as a sub-routine in our PHPE construction. Parts of this section is taken verbatim from [26, 40].

■ The Function Family $\mathcal{F}_{\mathsf{ABP}\circ\mathsf{IP}}^{(q,n',n)}$

The function class $\mathcal{F}_{\mathrm{ABP}\circ\mathrm{IP}}^{(q,n',n)}$, parameterized by a prime q and $n', n \in \mathbb{N}$, contains functions of the form $f : \mathbb{F}_q^{n'} \times \mathbb{F}_q^n \to \mathbb{F}_q$ defined by $f(\vec{x}, \vec{z}) = \sum_{j \in [n]} f_j(\vec{x}) z_j$ for all $\vec{x} = (x_1, \ldots, x_{n'}) \in \mathbb{F}_q^{n'}$ and $\vec{z} = (z_1, \ldots, z_n) \in \mathbb{F}_q^n$, where $f_1, \ldots, f_n : \mathbb{F}_q^{n'} \to \mathbb{F}_q$ are functions computed by some ABP's $\Gamma_1, \ldots, \Gamma_n$ respectively. We will view the input $\vec{x} = (x_1, \ldots, x_{n'})$ as the *public attribute string*, while $\vec{z} = (z_1, \ldots, z_n)$ as the *private attribute string*. Please refer to [40] for some illustrative examples. A simple but crucial property of the function f is that for any $\zeta \in \mathbb{F}_q$ and any $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$, we have $f(\vec{x}, \zeta\vec{z}) = \zeta f(\vec{x}, \vec{z})$.

Observe that the function f can itself be realized by an ABP Γ constructed as follows: First, marge the source vertices of all the component ABP's $\{\Gamma_j\}_{j \in [n]}$ together to form a single vertex, and designate it as the source vertex of the ABP Γ. Next, generate a new sink vertex for Γ, and for each $j \in [n]$, connect the sink vertex of the component ABP Γ_j to that newly formed sink vertex with a directed edge labeled with z_j. For ease of notations, we will denote the size of the ABP Γ computing the function f as $m + n + 1$, where 1 corresponds to the sink vertex of Γ, n accounts for the number of edges directed to that sink vertex, and m accounts for the number of other vertices and edges in Γ. Also, note that the ABP Γ can be further modified to another ABP Γ' in which each vertex has at most one outgoing edge having a label of degree one, by replacing each edge e in Γ with a pair of edges labeled 1 and $\phi(e)$ respectively, where ϕ is the labeling

function of the ABP Γ. It is clear that the number of vertices in this modified ABP Γ' is $m+n+1$, since Γ' is obtained by adding a fresh vertex for each edge in Γ as a result of replacing each edge in Γ with a pair of edges. Throughout this paper, whenever we will talk about the ABP computing the function f, we will refer to the ABP Γ' just described, unless otherwise specified.

■ **The Algorithm PGB**

► **Syntax and Properties**:

$\mathsf{PGB}(f;\vec{r})$: PGB is a PPT algorithm takes as input a function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, uses randomness $\vec{r} \in \mathbb{F}_q^{m+n-1}$, and outputs a collection of constants $(\{\sigma_j\}_{j\in[n]}, \{\alpha_{j'}, \gamma_{j'}\}_{j'\in[m]}) \in \mathbb{F}_q^n \times (\mathbb{F}_q^2)^m$ along with a function $\rho : [m] \to [n']$. Together with some $\vec{x} \in \mathbb{F}_q^{n'}$ and $\vec{z} \in \mathbb{F}_q^n$, this specifies a collection of $n+m$ *shares*

$$(\{z_j + \sigma_j\}_{j\in[n]}, \{\alpha_{j'} x_{\rho(j')} + \gamma_{j'}\}_{j'\in[m]}). \tag{2.1}$$

Here, $m+n+1$ is the number of vertices in the ABP computing f and ρ is deterministically derived from f.

The algorithm PGB satisfies the following properties:

- **Linearity**: For a fixed $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, $\mathsf{PGB}(f;\cdot)$ computes a linear function of its randomness over $\mathbb{F}_q$.
- **Reconstruction**: There exists a deterministic polynomial-time algorithm REC that on input any $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$ and any $\vec{x} \in \mathbb{F}_q^{n'}$, outputs a collection of coefficients $(\{\Omega_j\}_{j\in[n]}, \{\Omega'_{j'}\}_{j'\in[m]}) \in \mathbb{F}_q^n \times \mathbb{F}_q^{n'}$. These coefficients can be used in combination with any set of shares of the form as in Eq. (2.1), computed by combining the output of $\mathsf{PGB}(f)$ with $\vec{x}$ and any $\vec{z} \in \mathbb{F}_q^n$, to recover $f(\vec{x}, \vec{z})$. Moreover, the recovery procedure is linear in the shares used.
- **Privacy**: There exists a PPT simulator SIM such that for all $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, $\vec{x} \in \mathbb{F}_q^{n'}$, $\vec{z} \in \mathbb{F}_q^n$, the output of SIM on input f, $\vec{x}$, and $f(\vec{x}, \vec{z})$ is identically distributed to the shares obtained by combining the output of $\mathsf{PGB}(f;\vec{r})$ for uniformly random $\vec{r}$, with $\vec{x}$ and $\vec{z}$ as in Eq. (2.1).

► **Instantiation of the Algorithm**: We now sketch an instantiation of the algorithm PGB following [26, 40]. This instantiation will be utilized in our PHPE construction.

$\mathsf{PGB}(f)$: The algorithm takes as input a function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, and proceeds as follows:

1. Let Γ' denotes the ABP computing f as described above. Recall that in the ABP Γ', there are $m+n+1$ vertices, the variables z_j's only appear on edges leading into the sink vertex, and any vertex has at most one outgoing edge with a label of degree one. It first computes the matrix representation $\boldsymbol{L} \in \mathbb{F}_q^{(m+n)\times(m+n)}$ of the ABP Γ' using the efficient algorithm of Lemma 2.1. Then as per Lemma 2.1, the matrix $\boldsymbol{L}$ satisfies the following properties:
   - $\det(\boldsymbol{L}(\vec{x}, \vec{z})) = f(\vec{x}, \vec{z})$ for all $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$.

- For $j' \in [m]$, each entry in the j'^{th} row of $\boldsymbol{L}$ is either a degree one polynomial function in one (and the same) input variable $x_{\iota'}$ ($\iota' \in [n']$), with coefficients in $\mathbb{F}_q$ or a constant in $\mathbb{F}_q$.
- $\boldsymbol{L}$ contains only -1's in the second diagonal, and 0's below the second diagonal.
- The last column of $\boldsymbol{L}$ is $(0, \ldots, 0, z_1, \ldots, z_n)^\intercal$.
- $\boldsymbol{L}$ has 0's everywhere else in the last n rows.

It defines the function $\rho : [m] \to [n']$ as $\rho(j') = \iota'$ if the entries of the j'^{th} row of $\boldsymbol{L}$ involves the variable $x_{\iota'}$ for $j' \in [m]$.

2. Next, it chooses $\vec{r} \xleftarrow{\mathsf{U}} \mathbb{F}_q^{m+n-1}$, and computes
$$\boldsymbol{L} \begin{pmatrix} \vec{r}^\intercal \\ 1 \end{pmatrix} = (\alpha_1 x_{\rho(1)} + \gamma_1, \ldots, \alpha_m x_{\rho(m)} + \gamma_m, z_1 + \sigma_1, \ldots, z_n + \sigma_n)^\intercal.$$
3. It outputs $\big((\{\sigma_j\}_{j\in[n]}, \{\alpha_{j'}, \gamma_{j'}\}_{j'\in[m]}), \rho : [m] \to [n']\big)$.

It is straightforward to verify that each of $\{\sigma_j\}_{j\in[n]}, \{\alpha_{j'}, \gamma_{j'}\}_{j'\in[m]}$ are indeed linear functions of the randomness $\vec{r}$.

$\mathsf{REC}(f, \vec{x})$: This algorithm takes as input a function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$ and a vector $\vec{x} \in \mathbb{F}_q^{n'}$. It proceeds as follows:

1. It first executes Step 1 of the algorithm PGB described above to generate the matrix representation $\boldsymbol{L}$ of f.
2. Next, it computes the cofactors of each entry in the last column of $\boldsymbol{L}$. Let $(\{\Omega'_{j'}\}_{j'\in[m]}, \{\Omega_j\}_{j\in[n]}) \in \mathbb{F}_q^{m+n}$ be the collection of all the cofactors in the order of the entries. Note that the first $m+n-1$ columns of $\boldsymbol{L}$ involve only the variables $\{x_{\iota'}\}_{\iota'\in[n']}$. Hence, it can compute all the cofactors using the input $\vec{x}$.
3. It outputs $(\{\Omega_j\}_{j\in[n]}, \{\Omega'_{j'}\}_{j'\in[m]})$.

The output of $\mathsf{REC}(f, \vec{x})$ can be used in conjunction with a collection of shares $(\{z_j + \sigma_j\}_{j\in[n]}, \{\alpha_{j'} x_{\rho(j')} + \gamma_{j'}\}_{j'\in[m]})$ for any $\vec{z} \in \mathbb{F}_q^n$, to compute $f(\vec{x}, \vec{z})$ as
$$f(\vec{x}, \vec{z}) = \sum_{j'\in[m]} \Omega'_{j'}(\alpha_{j'} x_{\rho(j')} + \gamma_{j'}) + \sum_{j\in[n]} \Omega_j(z_j + \sigma_j). \tag{2.2}$$
Observe that the RHS of Eq. (2.2) corresponds to computing $\det(\boldsymbol{L}'(\vec{x}, \vec{z}))$, where the matrix $\boldsymbol{L}'$ is obtained by replacing the last column of the matrix $\boldsymbol{L}$ with the column $(\alpha_1 x_{\rho(1)} + \gamma_1, \ldots, \alpha_m x_{\rho(m)} + \gamma_m, z_1 + \sigma_1, \ldots, z_n + \sigma_n)^\intercal$, where $\boldsymbol{L}$ is the matrix representation of the ABP Γ' computing the function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, obtained by applying the algorithm of Lemma 2.1. Hence, the correctness of Eq. (2.2) follows from the fact that
$$\det(\boldsymbol{L}'(\vec{x}, \vec{z})) = \det(\boldsymbol{L}(\vec{x}, \vec{z})) \begin{vmatrix} 1 & & r_1 \\ & \ddots & \vdots \\ & 1 & r_{m+n-1} \\ & & 1 \end{vmatrix} = \det(\boldsymbol{L}(\vec{x}, \vec{z})) \cdot 1 = f(\vec{x}, \vec{z}).$$
Here, $\vec{r} = (r_1, \ldots, r_{m+n-1}) \in \mathbb{F}_q^{m+n-1}$ is the randomness used by PGB while generating the constants $(\{\sigma_j\}_{j\in[n]}, \{\alpha_{j'}, \gamma_{j'}\}_{j'\in[m]})$. In fact, an augmented version of Eq. (2.2) also holds. More precisely, for any $\Upsilon, \widetilde{\Upsilon} \in \mathbb{F}_q$, we have
$$\Upsilon f(\vec{x}, \vec{z}) = \sum_{j'\in[m]} \Omega'_{j'} \widetilde{\Upsilon}(\alpha_{j'} x_{\rho(j')} + \gamma_{j'}) + \sum_{j\in[n]} \Omega_j(\Upsilon z_j + \widetilde{\Upsilon}\sigma_j). \tag{2.3}$$

This follows by observing that

$$\det(\boldsymbol{L}(\vec{x},\vec{z})) = \sum_{j\in[n]} \Omega_j z_j, \text{(since the first } m \text{ entries in the last column is 0)}$$

and hence, the RHS of Eq. (2.3) can be written as

$$\widetilde{\Upsilon}\Big[\sum_{j'\in[m]} \Omega'_{j'}(\alpha_{j'} x_{\rho(j')} + \gamma_{j'}) + \sum_{j\in[n]} \Omega_j(z_j + \sigma_j)\Big] + (\Upsilon - \widetilde{\Upsilon}) \sum_{j\in[n]} \Omega_j z_j$$

$$= \widetilde{\Upsilon} \det(\boldsymbol{L}'(\vec{x},\vec{z})) + (\Upsilon - \widetilde{\Upsilon}) \det(\boldsymbol{L}(\vec{x},\vec{z})) = \Upsilon \det(\boldsymbol{L}(\vec{x},\vec{z})),$$

as $\det(\boldsymbol{L}'(\vec{x},\vec{z})) = \det(\boldsymbol{L}(\vec{x},\vec{z}))$. This fact will be used to justify the correctness of our PHPE construction.

$\mathsf{SIM}(f, \vec{x}, \epsilon)$: The simulator takes as input a function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, a vector $\vec{x} \in \mathbb{F}_q^{n'}$, and a value $\epsilon \in \mathbb{F}_q$. It proceeds as follows:

1. At first, it executes Step 1 of the algorithm PGB described above to obtain the matrix representation $\boldsymbol{L}$ of f together with the function $\rho : [m] \to [n']$.
2. Next, it constructs a matrix $\widehat{\boldsymbol{L}}$ from the matrix $\boldsymbol{L}$ by replacing its last column with $(\epsilon, 0, \ldots, 0)^\intercal$.
3. Next, it samples $\vec{r} \xleftarrow{\mathsf{U}} \mathbb{F}_q^{m+n-1}$, and computes
$$\widehat{\boldsymbol{L}} \begin{pmatrix} \vec{r}^\intercal \\ 1 \end{pmatrix} = (\mu_1, \ldots, \mu_m, \nu_1, \ldots, \nu_n)^\intercal.$$
4. It outputs $\big((\{\nu_j\}_{j\in[n]}, \{\mu_{j'}\}_{j'\in[m]}), \rho : [m] \to [n']\big)$.

It readily follows from Theorem 3, Corollary 1 of [26] that for all $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, $\vec{x} \in \mathbb{F}_q^{n'}$, and $\vec{z} \in \mathbb{F}_q^n$, the output of $\mathsf{SIM}(f, \vec{x}, f(\vec{x},\vec{z}))$ is identically distributed to the shares obtained by combining with $(\vec{x},\vec{z})$ the output of $\mathsf{PGB}(f)$ with uniform randomness, thereby establishing the privacy property of the algorithm PGB described above. We omit the details here. Clearly the determinant value of the matrix $\widehat{\boldsymbol{L}}(\vec{x},\vec{z})$ generated by SIM on input any $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, $\vec{x} \in \mathbb{F}_q^{n'}$, and $f(\vec{x},\vec{z})$ for any $\vec{z} \in \mathbb{F}_q^n$ is $f(\vec{x},\vec{z})$.

2.4 Bilinear Groups and Dual Pairing Vector Spaces

In this section, we will provide the necessary backgrounds on bilinear groups and dual pairing vector spaces, which are the primary building blocks of our PHPE construction.

Definition 2.1 (Bilinear Group): A bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a tuple of a prime integer $q \in \mathbb{N}$; cyclic multiplicative groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order q each with polynomial-time computable group operations; generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$; and a polynomial-time computable non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, i.e., e satisfies the following two properties:

- *Bilinearity*: $e(g_1^\delta, g_2^{\hat{\delta}}) = e(g_1, g_2)^{\delta\hat{\delta}}$ for all $\delta, \hat{\delta} \in \mathbb{F}_q$.
- *Non-degeneracy*: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ denotes the identity element of the group $\mathbb{G}_T$.

A bilinear group is said to be asymmetric if no efficiently computable isomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $\mathcal{G}_{\text{BPG}}$ be an algorithm that on input the unary encoding 1^λ of the security parameter λ, outputs a description $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ of a bilinear group.

Definition 2.2 (Dual Pairing Vector Spaces: DPVS [35, 33]): A dual pairing vector space (DPVS) $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e)$ formed by the direct product of a bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a tuple of a prime integer q; d-dimensional vector spaces $\mathbb{V}_t = \mathbb{G}_t^d$ over $\mathbb{F}_q$ for $t \in [2]$ under vector addition and scalar multiplication defined componentwise in the usual manner; canonical bases $\mathbb{A}_t = \{\boldsymbol{a}^{(t,\ell)} = (\overbrace{1_{\mathbb{G}_t}, \ldots, 1_{\mathbb{G}_t}}^{\ell-1}, g_t, \overbrace{1_{\mathbb{G}_t}, \ldots, 1_{\mathbb{G}_t}}^{d-\ell})\}_{\ell \in [d]}$ of $\mathbb{V}_t$ for $t \in [2]$, where $1_{\mathbb{G}_t}$ is the identity element of the group $\mathbb{G}_t$ for $t \in [2]$; and a pairing $e : \mathbb{V}_1 \times \mathbb{V}_2 \to \mathbb{G}_T$ defined by $e(\boldsymbol{v}, \boldsymbol{w}) = \prod_{\ell \in [d]} e(g_1^{v_\ell}, g_2^{w_\ell}) \in \mathbb{G}_T$ for all $\boldsymbol{v} = (g_1^{v_1}, \ldots, g_1^{v_d}) \in \mathbb{V}_1$, $\boldsymbol{w} = (g_2^{w_1}, \ldots, g_2^{w_d}) \in \mathbb{V}_2$. Observe that the newly defined map e is also non-degenerate bilinear, i.e., e also satisfies the following two properties:

- *Bilinearity*: $e(\delta \boldsymbol{v}, \widehat{\delta} \boldsymbol{w}) = e(\boldsymbol{v}, \boldsymbol{w})^{\delta\widehat{\delta}}$ for all $\delta, \widehat{\delta} \in \mathbb{F}_q$, $\boldsymbol{v} \in \mathbb{V}_1$, and $\boldsymbol{w} \in \mathbb{V}_2$.
- *Non-degeneracy*: If $e(\boldsymbol{v}, \boldsymbol{w}) = 1_{\mathbb{G}_T}$ for all $\boldsymbol{w} \in \mathbb{V}_2$, then $\boldsymbol{v} = (\overbrace{1_{\mathbb{G}_1}, \ldots, 1_{\mathbb{G}_1}}^{d})$. Similar statement also holds with the vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ interchanged.

For any ordered basis $\mathbb{W} = \{\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(d)}\}$ of $\mathbb{V}_t$ for $t \in [2]$, and any vector $\vec{v} \in \mathbb{F}_q^d$, let $(\vec{v})_{\mathbb{W}}$ represents the vector in $\mathbb{V}_t$ formed by the linear combination of the members of $\mathbb{W}$ with the components of $\vec{v}$ as the coefficients, i.e., $(\vec{v})_{\mathbb{W}} = \sum_{\ell \in [d]} v_\ell \boldsymbol{w}^{(\ell)} \in \mathbb{V}_t$. The DPVS generation algorithm $\mathcal{G}_{\mathrm{DPVS}}$ takes as input the unary encoded security parameter 1^λ, a dimension value $d \in \mathbb{N}$, along with a bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{BPG}}()$, and outputs a description $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e)$ of DPVS with d-dimensional $\mathbb{V}_1$ and $\mathbb{V}_2$.

We now describe random *dual orthonormal basis* generator $\mathcal{G}_{\mathrm{OB}}$ [35,33] in Fig. 2.1. This algorithm will be utilized as a sub-routine in our PHPE construction.

2.5 Complexity Assumption

For realizing our PHPE construction in asymmetric bilinear groups, we rely on the natural extension of the well-studied decisional linear (DLIN) assumption to the asymmetric bilinear group setting, called the external decisional linear (XDLIN) assumption.

Assumption (External Decisional Linear: XDLIN [1, 38]): For $t \in [2]$, the XDLIN_t problem is to guess the bit $\widehat{\beta} \xleftarrow{\mathsf{U}} \{0,1\}$ given $\varrho_{\widehat{\beta}}^{\mathrm{XDLIN}_t} = (\mathsf{params}_{\mathbb{G}}, g_1^{\varpi}, g_1^{\xi}, g_1^{\varkappa\varpi}, g_1^{\varsigma\xi}, g_2^{\varpi}, g_2^{\xi}, g_2^{\varkappa\varpi}, g_2^{\varsigma\xi}, \Re_{t,\widehat{\beta}})$, where

$$\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{BPG}}();$$
$$\varpi, \xi, \varkappa, \varsigma, \varepsilon \xleftarrow{\mathsf{U}} \mathbb{F}_q;$$
$$\Re_{t,0} = g_t^{(\varkappa+\varsigma)}, \Re_{t,1} = g_t^{(\varkappa+\varsigma)+\varepsilon}.$$

The XDLIN_t assumption states that for any PPT algorithm $\mathcal{E}$, for any security parameter λ, the advantage of $\mathcal{E}$ in deciding the XDLIN_t problem, defined as

$$\mathsf{Adv}_{\mathcal{E}}^{\mathrm{XDLIN}_t}(\lambda) = |\mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{E}(\varrho_0^{\mathrm{XDLIN}_t}) - \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{E}(\varrho_1^{\mathrm{XDLIN}_t})]|,$$

$\mathcal{G}_{\mathrm{OB}}(N, (d_0, \ldots, d_N))$: This algorithm takes as input the unary encoded security parameter 1^λ, a number $N \in \mathbb{N}$, and the respective dimensions $d_0, \ldots, d_N \in \mathbb{N}$ of the $N+1$ pairs of bases to be generated. It executes the following operations:

1. It first generates $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{BPG}}()$.
2. Next, it samples $\psi \xleftarrow{\mathsf{U}} \mathbb{F}_q \backslash \{0\}$ and computes $g_T = e(g_1, g_2)^\psi$.
3. Then, for $\imath \in [0, N]$, it performs the following:
   (a) It constructs $\mathsf{params}_{\mathbb{V}_\imath} = (q, \mathbb{V}_{\imath,1}, \mathbb{V}_{\imath,2}, \mathbb{G}_T, \mathbb{A}_{\imath,1}, \mathbb{A}_{\imath,2}, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{DPVS}}(d_\imath, \mathsf{params}_{\mathbb{G}})$.
   (b) It samples $\boldsymbol{B}^{(\imath)} = (b^{(\imath)}_{\ell,k}) \xleftarrow{\mathsf{U}} \mathsf{GL}(d_\imath, \mathbb{F}_q)$.
   (c) It computes $\boldsymbol{B}^{*(\imath)} = (b^{*(\imath)}_{\ell,k}) = \psi((\boldsymbol{B}^{(\imath)})^{-1})^\intercal$.
   (d) For all $\ell \in [d_\imath]$, let $\vec{b}^{(\imath,\ell)}$ and $\vec{b}^{*(\imath,\ell)}$ represent the ℓ^{th} rows of $\boldsymbol{B}^{(\imath)}$ and $\boldsymbol{B}^{*(\imath)}$ respectively. It computes $\boldsymbol{b}^{(\imath,\ell)} = (\vec{b}^{(\imath,\ell)})_{\mathbb{A}_{\imath,1}}, \boldsymbol{b}^{*(\imath,\ell)} = (\vec{b}^{*(\imath,\ell)})_{\mathbb{A}_{\imath,2}}$ for $\ell \in [d_\imath]$, and sets
   $$\mathbb{B}_\imath = \{\boldsymbol{b}^{(\imath,1)}, \ldots, \boldsymbol{b}^{(\imath,d_\imath)}\}, \mathbb{B}^*_\imath = \{\boldsymbol{b}^{*(\imath,1)}, \ldots, \boldsymbol{b}^{*(\imath,d_\imath)}\}.$$
   Clearly $\mathbb{B}_\imath$ and $\mathbb{B}^*_\imath$ form bases of the vector spaces $\mathbb{V}_{\imath,1}$ and $\mathbb{V}_{\imath,2}$ respectively. Also, note that $\mathbb{B}_\imath$ and $\mathbb{B}^*_\imath$ are dual orthonormal in the sense that for all $\ell, \ell' \in [d_\imath]$,
   $$e(\boldsymbol{b}^{(\imath,\ell)}, \boldsymbol{b}^{*(\imath,\ell')}) = \begin{cases} g_T & \text{if } \ell = \ell', \\ 1_{\mathbb{G}_T} & \text{otherwise.} \end{cases}$$
4. Next, it sets $\mathsf{params} = (\{\mathsf{params}_{\mathbb{V}_\imath}\}_{\imath \in [0,N]}, g_T)$.
5. It returns $(\mathsf{params}, \{\mathbb{B}_\imath, \mathbb{B}^*_\imath\}_{\imath \in [0,N]})$.

Fig. 2.1: Dual Orthonormal Basis Generator $\mathcal{G}_{\mathrm{OB}}$

is negligible in λ, i.e., $\mathsf{Adv}^{\mathrm{XDLIN}_t}_{\mathcal{E}}(\lambda) \leq \mathsf{negl}(\lambda)$, where negl is some negligible function. The simultaneous XDLIN (SXDLIN) assumption states that both XDLIN_1 and XDLIN_2 assumptions hold at the same time. For any security parameter λ, we denote the advantage of any probabilistic algorithm $\mathcal{E}$ against SXDLIN as $\mathsf{Adv}^{\mathrm{SXDLIN}}_{\mathcal{E}}(\lambda)$.

2.6 The Notion of Partially-Hiding Predicate Encryption

Here, we formally present the syntax and simulation-based security notion of a partially-hiding predicate encryption (PHPE) scheme for the function family $\mathcal{F}^{(q,n',n)}_{\mathrm{ABP} \circ \mathrm{IP}}$ for some prime q and $n', n \in \mathbb{N}$. Following [40], we define the ABP∘IP predicate family $R^{\mathrm{ABP} \circ \mathrm{IP}}$ as $R^{\mathrm{ABP} \circ \mathrm{IP}} = \{R^{\mathrm{ABP} \circ \mathrm{IP}}(f, (\cdot, \cdot)) : \mathbb{F}_q^{n'} \times \mathbb{F}_q^n \to \{0,1\} \mid f \in \mathcal{F}^{(q,n',n)}_{\mathrm{ABP} \circ \mathrm{IP}}\}$, where $R^{\mathrm{ABP} \circ \mathrm{IP}}(f, (\vec{x}, \vec{z})) = 1$ if $f(\vec{x}, \vec{z}) = 0$, and $R^{\mathrm{ABP} \circ \mathrm{IP}}(f, (\vec{x}, \vec{z})) = 0$ if $f(\vec{x}, \vec{z}) \neq 0$ for all $f \in \mathcal{F}^{(q,n',n)}_{\mathrm{ABP} \circ \mathrm{IP}}$ and $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$.

► **Syntax**: An *attribute-only/key-encapsulation mechanism* (KEM) *partially-hiding predicate encryption* (PHPE) scheme for the function family $\mathcal{F}^{(q,n',n)}_{\mathrm{ABP} \circ \mathrm{IP}}$ consists of the following polynomial-time algorithms:

$\mathsf{PHPE.Setup}(1^{n'}, 1^n)$: The setup algorithm takes as input the security parameter λ along with the public and private attribute lengths n' and n respectively

(all encoded in unary). It outputs the public parameters MPK and the master secret key MSK.

$\mathsf{PHPE.Encrypt}(\text{MPK}, (\vec{x}, \vec{z}))$: The encryption algorithm takes as input the public parameters MPK, a pair of public-private attribute strings $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$. It outputs a ciphertext CT. In the KEM mode, it additionally outputs a session key KEM.

$\mathsf{PHPE.KeyGen}(\text{MPK}, \text{MSK}, f)$: On input the public parameters MPK, the master secret key MSK, along with a function $f \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$, the key generation algorithm outputs a decryption key $\text{SK}(f)$.

$\mathsf{PHPE.Decrypt}(\text{MPK}, (f, \text{SK}(f)), (\vec{x}, \text{CT}))$: The decryption algorithm takes as input the public parameters MPK, a pair of a function $f \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ and a decryption key $\text{SK}(f)$ for f, along with a pair of a public attribute $\vec{x} \in \mathbb{F}_q^{n'}$ and a ciphertext CT associated with $\vec{x}$ and some private attribute string. In the attribute-only mode, it outputs either 1 or 0, while in the KEM mode, it outputs a session key $\widetilde{\text{KEM}}$. For notational convenience, we will think of f and $\vec{x}$ as parts of $\text{SK}(f)$ and CT respectively, and will not write them explicitly in the argument of PHPE.Decrypt.

The algorithm PHPE.Decrypt is deterministic, while all the others are probabilistic.

► **Correctness**: A PHPE scheme for the function family $\mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ is said to be correct if for any security parameter λ, any $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$, any $f \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$, any $(\text{MPK}, \text{MSK}) \xleftarrow{\mathsf{R}} \mathsf{PHPE.Setup}(1^{n'}, 1^n)$, and any $\text{SK}(f) \xleftarrow{\mathsf{R}} \mathsf{PHPE.KeyGen}(\text{MPK}, \text{MSK}, f)$, the following holds:

– (*Authorized*) If $R^{\text{ABP}\circ\text{IP}}(f, (\vec{x}, \vec{z})) = 1$, then
$\mathsf{Pr}[\mathsf{PHPE.Decrypt}(\text{MPK}, \text{SK}(f), \text{CT}) = 1 :$
$\text{CT} \xleftarrow{\mathsf{R}} \mathsf{PHPE.Encrypt}(\text{MPK}, (\vec{x}, \vec{z}))] \geq 1 - \mathsf{negl}(\lambda)$ (attribute-only mode),
$\mathsf{Pr}[\mathsf{PHPE.Decrypt}(\text{MPK}, \text{SK}(f), \text{CT}) = \text{KEM} :$
$(\text{CT}, \text{KEM}) \xleftarrow{\mathsf{R}} \mathsf{PHPE.Encrypt}(\text{MPK}, (\vec{x}, \vec{z}))] \geq 1 - \mathsf{negl}(\lambda)$ (KEM mode).
– (*Unauthorized*) If $R^{\text{ABP}\circ\text{IP}}(f, (\vec{x}, \vec{z})) = 0$, then
$\mathsf{Pr}[\mathsf{PHPE.Decrypt}(\text{MPK}, \text{SK}(f), \text{CT}) = 0 :$
$\text{CT} \xleftarrow{\mathsf{R}} \mathsf{PHPE.Encrypt}(\text{MPK}, (\vec{x}, \vec{z}))] \geq 1 - \mathsf{negl}(\lambda)$ (attribute-only mode),
$\mathsf{Pr}[\mathsf{PHPE.Decrypt}(\text{MPK}, \text{SK}(f), \text{CT}) \neq \text{KEM} :$
$(\text{CT}, \text{KEM}) \xleftarrow{\mathsf{R}} \mathsf{PHPE.Encrypt}(\text{MPK}, (\vec{x}, \vec{z}))] \geq 1 - \mathsf{negl}(\lambda)$ (KEM mode).

Here, negl is some negligible function, and the probabilities are taken over the random coins of PHPE.Encrypt.

► **Simulation-Based Security**: The simulation-based adaptively strongly partially-hiding security notion for a PHPE scheme is formulated by considering the following two experiments involving a stateful probabilistic adversary $\mathcal{A}$ and a stateful probabilistic simulator $\mathcal{S}$:

$\mathsf{Exp}^{\text{PHPE,REAL}}_{\mathcal{A}}(\lambda)$:

1. $(\text{MPK}, \text{MSK}) \xleftarrow{\mathsf{R}} \mathsf{PHPE.Setup}(1^{n'}, 1^{n})$.
2. $\{(\vec{x}^{(i)}, \vec{z}^{(i)})\}_{i\in[q_{\text{CT}}]} \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathsf{PHPE.KeyGen}(\text{MSK},\cdot)}(\text{MPK})$.
3. (a) (attribute-only case) $\text{CT}^{(i)} \xleftarrow{\mathsf{R}} \mathsf{PHPE.Encrypt}(\text{MPK}, (\vec{x}^{(i)}, \vec{z}^{(i)}))$ for $i \in [q_{\text{CT}}]$.
   (b) (KEM case) $(\text{CT}^{(i)}, \text{KEM}^{(i)}) \xleftarrow{\mathsf{R}} \mathsf{PHPE.Encrypt}(\text{MPK}, (\vec{x}^{(i)}, \vec{z}^{(i)}))$ for $i \in [q_{\text{CT}}]$.
4. (a) (attribute-only case) $\Im \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathsf{PHPE.KeyGen}(\text{MSK},\cdot)}(\text{MPK}, \{\text{CT}^{(i)}\}_{i\in[q_{\text{CT}}]})$.
   (b) (KEM case) $\Im \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathsf{PHPE.KeyGen}(\text{MSK},\cdot)}(\text{MPK}, \{(\text{CT}^{(i)}, \text{KEM}^{(i)})\}_{i\in[q_{\text{CT}}]})$.
5. Output $\varrho^{\text{PHPE,REAL}}_{\mathcal{A}} = \big(\text{MPK}, \{(\vec{x}^{(i)}, \vec{z}^{(i)})\}_{i\in[q_{\text{CT}}]}, \Im\big)$.

$\mathsf{Exp}^{\text{PHPE,IDEAL}}_{\mathcal{A},\mathcal{S}}(\lambda)$:

1. $\text{MPK} \xleftarrow{\mathsf{R}} \mathcal{S}(1^{n'}, 1^{n})$.
2. $\{(\vec{x}^{(i)}, \vec{z}^{(i)})\}_{i\in[q_{\text{CT}}]} \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{S}(\cdot)}(\text{MPK})$.
3. (a) (attribute-only case) $\{\text{CT}^{(i)}\}_{i\in[q_{\text{CT}}]} \xleftarrow{\mathsf{R}} \mathcal{S}(q_{\text{CT}}, \{(\vec{x}^{(i)}, R^{\text{ABP}\circ\text{IP}}(f_h, (\vec{x}^{(i)}, \vec{z}^{(i)})))\}_{i\in[q_{\text{CT}}],h\in[q_{\text{KEY-PRE}}]})$.
   (b) (KEM case) $\{\text{KEM}^{(i)}\}_{i\in[q_{\text{CT}}]} \xleftarrow{\mathsf{U}} \mathbb{K}$, where $\mathbb{K}$ = session key space $\{\text{CT}^{(i)}\}_{i\in[q_{\text{CT}}]} \xleftarrow{\mathsf{R}} \mathcal{S}(q_{\text{CT}}, \{(\vec{x}^{(i)}, \overline{\text{KEM}}^{(i,h)})\}_{i\in[q_{\text{CT}}],h\in[q_{\text{KEY-PRE}}]})$, where for all $i \in [q_{\text{CT}}], h \in [q_{\text{KEY-PRE}}]$, $\overline{\text{KEM}}^{(i,h)} = \text{KEM}^{(i)}$ if $R^{\text{ABP}\circ\text{IP}}(f_h, (\vec{x}^{(i)}, \vec{z}^{(i)})) = 1$, and $\perp$ if $R^{\text{ABP}\circ\text{IP}}(f_h, (\vec{x}^{(i)}, \vec{z}^{(i)})) = 0$.
4. (a) (attribute-only case) $\Im \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{S}^{\mathcal{O}_{R^{\text{ABP}\circ\text{IP}}(\{(\vec{x}^{(i)},\vec{z}^{(i)})\}_{i\in[q_{\text{CT}}]},\cdot)}(\cdot)}}(\text{MPK}, \{\text{CT}^{(i)}\}_{i\in[q_{\text{CT}}]})$.
   (b) (KEM case) $\Im \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{S}^{\mathcal{O}_{R^{\text{ABP}\circ\text{IP}}(\{((\vec{x}^{(i)},\vec{z}^{(i)}),\text{KEM}^{(i)})\}_{i\in[q_{\text{CT}}]},\cdot)}(\cdot)}}(\text{MPK}, \{(\text{CT}^{(i)}, \text{KEM}^{(i)})\}_{i\in[q_{\text{CT}}]})$.
5. Output $\varrho^{\text{PHPE,IDEAL}}_{\mathcal{A},\mathcal{S}} = \big(\text{MPK}, \{(\vec{x}^{(i)}, \vec{z}^{(i)})\}_{i\in[q_{\text{CT}}]}, \Im\big)$.

Here, the simulator $\mathcal{S}$ accepts as input a function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$ when it acts as an oracle to $\mathcal{A}$. Also, q_{CT} and $q_{\text{KEY-PRE}}$ respectively denotes the number of ciphertext queries made by $\mathcal{A}$ and number of decryption key queries made by $\mathcal{A}$ prior to submitting the ciphertext queries. Further, in the attribute-only case, the oracle $\mathcal{O}_{R^{\text{ABP}\circ\text{IP}}}$ receives as its second argument a function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, and outputs $\{R^{\text{ABP}\circ\text{IP}}(f, (\vec{x}^{(i)}, \vec{z}^{(i)}))\}_{i\in[q_{\text{CT}}]}$. On the other hand, in the KEM case, the oracle $\mathcal{O}_{R^{\text{ABP}\circ\text{IP}}}$ takes as its second argument a function $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$, and outputs $\text{KEM}^{(i)}$ if $R^{\text{ABP}\circ\text{IP}}(f, (\vec{x}^{(i)}, \vec{z}^{(i)})) = 1$, and $\perp$ if $R^{\text{ABP}\circ\text{IP}}(f, (\vec{x}^{(i)}, \vec{z}^{(i)})) = 0$ for $i \in [q_{\text{CT}}]$. A simulator $\mathcal{S}$ is said to be admissible if on each decryption key query $f \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$ of $\mathcal{A}$ in the post-ciphertext query phase, $\mathcal{S}$ makes just a single query to the oracle $\mathcal{O}_{R^{\text{ABP}\circ\text{IP}}}$ on f itself. Let the number of decryption key queries made by $\mathcal{A}$ after receiving the queried ciphertexts be $q_{\text{KEY-POST}}$.

For any security parameter λ, for any probabilistic distinguisher $\mathcal{D}$, the advantage of $\mathcal{D}$ in distinguishing the above two experiments is defined as

$$\mathsf{Adv}^{\text{PHPE,SIM-AH}}_{\mathcal{D}}(\lambda) = |\mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{D}(\varrho^{\text{PHPE,REAL}}_{\mathcal{A}})] - \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{D}(\varrho^{\text{PHPE,IDEAL}}_{\mathcal{A},\mathcal{S}})]|.$$

Definition 2.3: A PHPE scheme is called $(q_{\text{KEY-PRE}}, q_{\text{CT}}, q_{\text{KEY-POST}})$-simulation-based adaptively strongly partially hiding if there exists an admissible stateful PPT simulator $\mathcal{S}$ such that for any stateful PPT adversary $\mathcal{A}$ making at most q_{CT} ciphertext queries, $q_{\text{KEY-PRE}}$ decryption key queries in the pre-ciphertext query phase, while $q_{\text{KEY-POST}}$ decryption key queries in the post-ciphertext query phase, any PPT distinguisher $\mathcal{D}$, and any security parameter λ, $\mathsf{Adv}_{\mathcal{D}}^{\text{PHPE,SIM-AH}}(\lambda) \leq \mathsf{negl}(\lambda)$, where negl is some negligible function. Further, a PHPE scheme is said to be $(\mathsf{poly}, q_{\text{CT}}, \mathsf{poly})$-simulation-based adaptively strongly partially hiding if it is $(q_{\text{KEY-PRE}}, q_{\text{CT}}, q_{\text{KEY-POST}})$-simulation-based adaptively strongly partially hiding as well as $q_{\text{KEY-PRE}}$ and $q_{\text{KEY-POST}}$ are unbounded polynomials in the security parameter λ.

Remark 2.1: Consider an adversary $\mathcal{H}$ that first invokes $\mathcal{A}$ and then invokes $\mathcal{D}$ once the transcript ($\varrho_{\mathcal{A}}^{\text{PHPE,REAL}}$ or $\varrho_{\mathcal{A},\mathcal{S}}^{\text{PHPE,IDEAL}}$) of the experiment is obtained. Consider the experiments $\mathsf{Exp}_{\mathcal{H}}^{\text{PHPE,REAL}}(\lambda)$ and $\mathsf{Exp}_{\mathcal{H},\mathcal{S}}^{\text{PHPE,IDEAL}}(\lambda)$ which are obtained from the experiments $\mathsf{Exp}_{\mathcal{A}}^{\text{PHPE,REAL}}(\lambda)$ and $\mathsf{Exp}_{\mathcal{A},\mathcal{S}}^{\text{PHPE,IDEAL}}(\lambda)$ respectively by applying the corresponding augmentations. Let us define the outputs of the augmented experiments as the output of $\mathcal{H}$ in those experiments, and the advantage of $\mathcal{H}$ as

$$\mathsf{Adv}_{\mathcal{H}}^{\text{PHPE,SIM-AH}}(\lambda) = |\mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathsf{Exp}_{\mathcal{H}}^{\text{PHPE,REAL}}(\lambda)] - \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathsf{Exp}_{\mathcal{H},\mathcal{S}}^{\text{PHPE,IDEAL}}(\lambda)]|.$$

Then, clearly $\mathsf{Adv}_{\mathcal{H}}^{\text{PHPE,SIM-AH}}(\lambda) = \mathsf{Adv}_{\mathcal{D}}^{\text{PHPE,SIM-AH}}(\lambda)$. We make use of this combined adversary $\mathcal{H}$ as well as the associated augmented experiments $\mathsf{Exp}_{\mathcal{H}}^{\text{PHPE,REAL}}(\lambda)$ and $\mathsf{Exp}_{\mathcal{H},\mathcal{S}}^{\text{PHPE,IDEAL}}(\lambda)$ in the security proof of our PHPE construction, both the attribute-only and KEM versions.

3 The Proposed PHPE Scheme

3.1 Construction

In this section, we will present our PHPE scheme for the function family $\mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$. This construction is presented in the attribute-only mode, i.e., without any actual payload. A key-encapsulation mechanism (KEM) version of this construction is presented in the full version of this paper. In the proposed scheme, we assume that the function ρ outputted by $\mathsf{PGB}(f)$ for any $f \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ is injective. This restriction can be readily overcome using standard techniques along the lines of [30, 34].

$\mathsf{PHPE.Setup}(1^{n'}, 1^n)$: The setup algorithm takes as input the security parameter λ together with the lengths n' and n of the public and private attribute strings respectively. It proceeds as follows:

1. It first generates $(\mathsf{params}, \{\mathbb{B}_\imath, \mathbb{B}_\imath^*\}_{\imath\in[n'+n]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\text{OB}}(n'+n, (0, \overbrace{9, \ldots, 9}^{n'+n}))$.
2. For $\imath \in [n'+n]$, it sets $\widehat{\mathbb{B}}_\imath = \{\boldsymbol{b}^{(\imath,1)}, \boldsymbol{b}^{(\imath,2)}, \boldsymbol{b}^{(\imath,9)}\}, \widehat{\mathbb{B}}_\imath^* = \{\boldsymbol{b}^{*(\imath,1)}, \boldsymbol{b}^{*(\imath,2)}, \boldsymbol{b}^{*(\imath,7)}, \boldsymbol{b}^{*(\imath,8)}\}$.

3. It outputs the public parameters $\mathrm{MPK} = (\mathsf{params}, \{\widehat{\mathbb{B}}_\iota\}_{\iota\in[n'+n]})$ and the master secret key $\mathrm{MSK} = \{\widehat{\mathbb{B}}^*_i\}_{\iota\in[n'+n]}$.

$\mathsf{PHPE.Encrypt}(\mathrm{MPK}, (\vec{x}, \vec{z}))$: The encryption algorithm takes as input the public parameters MPK and a pair of public-private attribute strings $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$. It executes the following:

1. First, it samples $\omega \xleftarrow{\mathsf{U}} \mathbb{F}_q$.
2. Next, for $\iota' \in [n']$, it samples $\varphi'_{\iota'} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and computes
$$\boldsymbol{c}'^{(\iota')} = (\omega(1, x_{\iota'}), \vec{0}^4, \vec{0}^2, \varphi'_{\iota'})_{\mathbb{B}_{\iota'}}.$$
3. Then, for $\iota \in [n]$, it samples $\varphi_\iota \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and computes
$$\boldsymbol{c}^{(\iota)} = (\omega(1, z_\iota), \vec{0}^4, \vec{0}^2, \varphi_\iota)_{\mathbb{B}_{n'+\iota}}.$$
4. It outputs the ciphertext $\mathrm{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota'\in[n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota\in[n]})$.

$\mathsf{PHPE.KeyGen}(\mathrm{MPK}, \mathrm{MSK}, f)$: The key generation algorithm takes as input the public parameters MPK, the master secret key MSK, along with a function $f \in \mathcal{F}^{(q,n',n)}_{\mathrm{ABP}\circ\mathrm{IP}}$. It operates as follows:

1. It first generates $\big((\{\sigma_j\}_{j\in[n]}, \{\alpha_{j'}, \gamma_{j'}\}_{j'\in[m]}), \rho : [m] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{PGB}(f)$.
2. Next, it samples $\zeta \xleftarrow{\mathsf{U}} \mathbb{F}_q$.
3. Then, for $j' \in [m]$, it samples $\vec{\kappa}'^{(j')} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and computes
$$\boldsymbol{k}'^{(j')} = ((\gamma_{j'}, \alpha_{j'}), \vec{0}^4, \vec{\kappa}'^{(j')}, 0)_{\mathbb{B}^*_{\rho(j')}}.$$
4. Then, for $j \in [n]$, it samples $\vec{\kappa}^{(j)} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and computes
$$\boldsymbol{k}^{(j)} = ((\sigma_j, \zeta), \vec{0}^4, \vec{\kappa}^{(j)}, 0)_{\mathbb{B}^*_{n'+j}}.$$
5. It outputs the decryption key $\mathrm{SK}(f) = (f, \{\boldsymbol{k}'^{(j')}\}_{j'\in[m]}, \{\boldsymbol{k}^{(j)}\}_{j\in[n]})$.

$\mathsf{PHPE.Decrypt}(\mathrm{MPK}, \mathrm{SK}(f), \mathrm{CT})$: The decryption algorithm takes in the public parameters MPK, a decryption key $\mathrm{SK}(f) = (f, \{\boldsymbol{k}'^{(j')}\}_{j'\in[m]}, \{\boldsymbol{k}^{(j)}\}_{j\in[n]})$, and a ciphertext $\mathrm{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota'\in[n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota\in[n]})$. It proceeds as follows:

1. It first computes $\Lambda'_{j'} = e(\boldsymbol{c}'^{(\rho(j'))}, \boldsymbol{k}'^{(j')})$ for $j' \in [m]$, and $\Lambda_j = e(\boldsymbol{c}^{(j)}, \boldsymbol{k}^{(j)})$ for $j \in [n]$.
2. Next, it determines the coefficients $(\{\Omega_j\}_{j\in[n]}, \{\Omega'_{j'}\}_{j'\in[m]}) = \mathsf{REC}(f, \vec{x})$.
3. Then, it computes $\Lambda = \left(\prod_{j'\in[m]} \Lambda'^{\Omega'_{j'}}_{j'}\right)\left(\prod_{j\in[n]} \Lambda^{\Omega_j}_j\right)$.
4. It outputs 1 if $\Lambda = 1_{\mathbb{G}_T}$, and 0 otherwise, where $1_{\mathbb{G}_T}$ is the identity element in $\mathbb{G}_T$.

► **Correctness**: For any decryption key $\mathrm{SK}(f) = (f, \{\boldsymbol{k}'^{(j')}\}_{j'\in[m]}, \{\boldsymbol{k}^{(j)}\}_{j\in[n]})$ for a function $f \in \mathcal{F}^{(q,n',n)}_{\mathrm{ABP}\circ\mathrm{IP}}$, and any ciphertext $\mathrm{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota'\in[n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota\in[n]})$ encrypting a pair of public-private attribute strings $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$, we have
$$\Lambda'_{j'} = g_T^{\omega(\alpha_{j'} x_{\rho(j')} + \gamma_{j'})} \text{ for } j' \in [m],$$
$$\Lambda_j = g_T^{\omega(\zeta z_j + \sigma_j)} \text{ for } j \in [n].$$

The above follows from the expressions of $\{\boldsymbol{k}'^{(j')}\}_{j'\in[m]}, \{\boldsymbol{k}^{(j)}\}_{j\in[n]}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota'\in[n']}$, $\{\boldsymbol{c}^{(\iota)}\}_{\iota\in[n]}$, and the dual orthonormality property of $\{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota\in[n'+n]}$. Hence, from Eq. (2.3) it follows that

$$\Lambda = g_T^{\omega\zeta f(\vec{x},\vec{z})}.$$

Therefore, if $R^{\text{ABP}\circ\text{IP}}(f,(\vec{x},\vec{z})) = 1$, i.e., $f(\vec{x},\vec{z}) = 0$, then $\Lambda = 1_{\mathbb{G}_T}$, while if $R^{\text{ABP}\circ\text{IP}}(f,(\vec{x},\vec{z})) = 0$, i.e., $f(\vec{x},\vec{z}) \neq 0$, then $\Lambda \neq 1_{\mathbb{G}_T}$ with all but negligible probability $2/q$, i.e., except when $\omega = 0$ or $\zeta = 0$.

Remark 3.1 (On Multi-Ciphertext Scheme): The PHPE scheme described above is only secure against adversaries that are allowed to make a single ciphertext query. However, we can readily extend the above scheme to one that is secure for any a priori bounded number of ciphertext queries of the adversary. The extension is as follows: Suppose we want to design a scheme that is secure for q_{CT} number of ciphertext queries. Then, we would introduce a $4q_{\text{CT}}$-dimensional hidden subspace on each of the ciphertext and the decryption key sides, where each 4-dimensional hidden subspace on the ciphertext side and its corresponding 4-dimensional dual subspace on the decryption key side will be used to handle each ciphertext query in the security reduction. Clearly the size of ciphertexts, decryption keys, and public parameters would scale linearly with q_{CT}.

3.2 Security

We now present our main theorem:

Theorem 3.1: *The proposed* PHPE *scheme is* $(\mathsf{poly}, 1, \mathsf{poly})$*-simulation-based adaptively strongly partially hiding* (*as per the security model described in Section 2.6*) *under the* SXDLIN *assumption.*

Following corollary is immediate from the relation between indistinguishability-based and simulation-based security for FE, as mentioned in the Introduction as well as the equivalence of the single- and multi-ciphertext security in the indistinguishability-based setting for FE:

Corollary 3.1: *The proposed* PHPE *scheme is* $(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$*-indistinguishability-based adaptively strongly partially hiding* (*as per the security model described in [12] and the full version of this paper*) *under the* SXDLIN *assumption.*

In order to prove Theorem 3.1, we consider a sequence of hybrid experiments which differ from one another in the construction of the ciphertext and/or the decryption keys queried by the augmented adversary $\mathcal{H}$ (described in Remark 2.1). The first hybrid corresponds to the experiment $\mathsf{Exp}_{\mathcal{H}}^{\text{PHPE,REAL}}(\lambda)$ (described in Section 2.6), while the last one corresponds to the experiment $\mathsf{Exp}_{\mathcal{H},\mathcal{S}}^{\text{PHPE,IDEAL}}(\lambda)$ (also described in Section 2.6) with the simulator $\mathcal{S}$ described below. We argue that $\mathcal{H}$'s probability of outputting 1 changes only by a negligible amount in each successive hybrid experiment, thereby establishing Theorem 3.1. Note that we are considering only one ciphertext query made by the adversary $\mathcal{H}$. Let, $q_{\text{KEY-PRE}}, q_{\text{KEY-POST}}$ be respectively the number of decryption key queries made by $\mathcal{H}$ before and after making the ciphertext query, and $q_{\text{KEY}} = q_{\text{KEY-PRE}} + q_{\text{KEY-POST}}$. Note that we consider $q_{\text{KEY-PRE}}$ and $q_{\text{KEY-POST}}$ to be arbitrary polynomials in the security parameter λ.

■ **Description of the Simulator**

The simulator $\mathcal{S}$ is described below.

- In order to generate the public parameters, $\mathcal{S}$ proceeds as follows:
  1. It first generates $(\mathsf{params}, \{\mathbb{B}_\iota, \mathbb{B}_\iota^*\}_{\iota \in [n'+n]}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{OB}}(n'+n, (0, \overbrace{9, \ldots, 9}^{n'+n}))$.
  2. For $\iota \in [n'+n]$, it sets $\widehat{\mathbb{B}}_\iota = \{\boldsymbol{b}^{(\iota,1)}, \boldsymbol{b}^{(\iota,2)}, \boldsymbol{b}^{(\iota,9)}\}$.
  3. It outputs the public parameters $\mathrm{MPK} = (\mathsf{params}, \{\widehat{\mathbb{B}}_\iota\}_{\iota \in [n'+n]})$.
- For $h \in [q_{\text{KEY-PRE}}]$, $\mathcal{S}$ simulates the h^{th} decryption key queried by $\mathcal{H}$ corresponding to some function $f_h \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$ as follows:
  1. At first, it generates $\big((\{\sigma_{h,j}\}_{j\in[n]}, \{\alpha_{h,j'}, \gamma_{h,j'}\}_{j'\in[m_h]}), \rho_h : [m_h] \to [n']\big)$, $\big((\{\widehat{\sigma}_{h,j}\}_{j\in[n]}, \{\widehat{\alpha}_{h,j'}, \widehat{\gamma}_{h,j'}\}_{j'\in[m_h]}), \rho_h : [m_h] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{PGB}(f_h)$.
  2. Next, it samples $\zeta_h, \widehat{\zeta}_h \xleftarrow{\mathsf{U}} \mathbb{F}_q$.
  3. Then, for $j' \in [m_h]$, it samples $\vec{\kappa}'^{(h,j')} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and computes
  $$\boldsymbol{k}'^{(h,j')} = ((\gamma_{h,j'}, \alpha_{h,j'}), \vec{0}^2, (\widehat{\gamma}_{h,j'}, \widehat{\alpha}_{h,j'}), \vec{\kappa}'^{(h,j')}, 0)_{\mathbb{B}^*_{\rho_h(j')}}.$$
  4. Then, for $j \in [n]$, it samples $\vec{\kappa}^{(h,j)} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and computes
  $$\boldsymbol{k}^{(h,j)} = ((\sigma_{h,j}, \zeta_h), \vec{0}^2, (\widehat{\sigma}_{h,j}, \widehat{\zeta}_h), \vec{\kappa}^{(h,j)}, 0)_{\mathbb{B}^*_{n'+j}}.$$
  5. It outputs $\mathrm{SK}(f_h) = (f_h, \{\boldsymbol{k}'^{(h,j')}\}_{j'\in[m_h]}, \{\boldsymbol{k}^{(h,j)}\}_{j\in[n]})$.
- When $\mathcal{H}$ queries a ciphertext for some pair of public-private attribute strings $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$, $\mathcal{S}$ receives $\vec{x}$ and $\{R^{\text{ABP}\circ\text{IP}}(f_h, (\vec{x}, \vec{z}))\}_{h\in[q_{\text{KEY-PRE}}]}$. It simulates the ciphertext as follows:
  1. At first, it samples $\vec{s} \xleftarrow{\mathsf{U}} S = \{\vec{s} \in \mathbb{F}_q^n \mid R^{\text{ABP}\circ\text{IP}}(f_h, (\vec{x}, \vec{s})) = R^{\text{ABP}\circ\text{IP}}(f_h, (\vec{x}, \vec{z}))\forall h \in [q_{\text{KEY-PRE}}]\}$. Observe that the set S is exactly identical to the set $\widetilde{S} = \{\vec{s} \in \mathbb{F}_q^n \mid R^{\text{IP}}((f_{h,1}(\vec{x}), \ldots, f_{h,n}(\vec{x})), \vec{s}) = R^{\text{IP}}((f_{h,1}(\vec{x}), \ldots, f_{h,n}(\vec{x})), \vec{z})\forall h \in [q_{\text{KEY-PRE}}]\}$, where R^{IP} represents the inner-product predicate family defined as $R^{\text{IP}} = \{R^{\text{IP}}(\vec{w}, \cdot) : \mathbb{F}_q^n \to \{0,1\} \mid \vec{w} \in \mathbb{F}_q^n\}$ such that $R^{\text{IP}}(\vec{w}, \vec{v}) = 1$ if $\vec{v} \cdot \vec{w} = 0$, and 0 if $\vec{v} \cdot \vec{w} \neq 0$ for $\vec{v}, \vec{w} \in \mathbb{F}_q^n$, and $f_{h,j}$ is the j^{th} component ABP of f_h for $h \in [q_{\text{KEY-PRE}}], j \in [n]$. It has already been demonstrated by O'Neill [37] that the inner-product predicate family R^{IP} is *pre-image-samplable*, which essentially means that we can efficiently sample from the set $\widetilde{S}$. In fact, he provided an explicit algorithm for doing this. Thus, given $\{f_h\}_{h\in[q_{\text{KEY-PRE}}]}$ and $\vec{x}$, $\mathcal{S}$ can efficiently sample from S by first determining the vectors $\{(f_{h,1}(\vec{x}), \ldots, f_{h,n}(\vec{x}))\}_{h\in[q_{\text{KEY-PRE}}]}$ and then sampling from the set $\widetilde{S}$ using the algorithm described in [37].
  2. Then, it samples $\tau, \theta, \xleftarrow{\mathsf{U}} \mathbb{F}_q$.
  3. Next, for $\iota' \in [n']$, it samples $\varphi'_{\iota'} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and computes
  $$\boldsymbol{c}'^{(\iota')} = (\vec{0}^3, \tau, \theta(1, x_{\iota'}), \vec{0}^2, \varphi'_{\iota'})_{\mathbb{B}_{\iota'}}.$$
  4. Then, for $\iota \in [n]$, it samples $\varphi_\iota \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and computes
  $$\boldsymbol{c}^{(\iota)} = (\vec{0}^3, \tau, \theta(1, s_\iota), \vec{0}^2, \varphi_\iota)_{\mathbb{B}_{n'+\iota}}.$$
  5. It outputs the ciphertext $\mathrm{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota'\in[n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota\in[n]})$.

- For $h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}]$, in response to the h^{th} decryption key query of $\mathcal{H}$ corresponding to some function $f_h \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$, $\mathcal{S}$ executes the following steps:
  1. It first generates $\big((\{\sigma_{h,j}\}_{j\in[n]}, \{\alpha_{h,j'}, \gamma_{h,j'}\}_{j'\in[m_h]}), \rho_h : [m_h] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{PGB}(f_h)$.
  2. Next, it samples $\zeta_h \xleftarrow{\mathsf{U}} \mathbb{F}_q$.
  3. After that, it queries its oracle $\mathcal{O}_{R^{\text{ABP}\circ\text{IP}}}((\vec{x},\vec{z}),\cdot)$ with the function f_h, and receives back $R^{\text{ABP}\circ\text{IP}}(f_h,(\vec{x},\vec{z}))$. If $R^{\text{ABP}\circ\text{IP}}(f_h(\vec{x},\vec{z})) = 1$, i.e., $f_h(\vec{x},\vec{z}) = 0$, it forms $\big((\{\nu_{h,j}\}_{j\in[n]}, \{\mu_{h,j'}\}_{j'\in[m_h]}), \rho_h : [m_h] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{SIM}(f_h, \vec{x}, 0)$. Otherwise, if $R^{\text{ABP}\circ\text{IP}}(f_h,(\vec{x},\vec{z})) = 0$, i.e., $f_h(\vec{x},\vec{z}) \neq 0$, then it samples $\check{\zeta}_h \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and generates $\big((\{\nu_{h,j}\}_{j\in[n]}, \{\mu_{h,j'}\}_{j'\in[m_h]}), \rho_h : [m_h] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{SIM}(f_h, \vec{x}, \check{\zeta}_h)$.
  4. Then, for $j' \in [m_h]$, it samples $\eta'_{h,j'} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \vec{\kappa}'^{(h,j')} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and computes
$$\boldsymbol{k}'^{(h,j')} = ((\gamma_{h,j'}, \alpha_{h,j'}), (\eta'_{h,j'}, \mu_{h,j'}), \vec{0}^2, \vec{\kappa}'^{(h,j')}, 0)_{\mathbb{B}^*_{\rho_h(j')}}.$$
  5. Then, for $j \in [n]$, it samples $\eta_{h,j} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \vec{\kappa}^{(h,j)} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, and computes
$$\boldsymbol{k}^{(h,j)} = ((\sigma_{h,j}, \zeta_h), (\eta_{h,j}, \nu_{h,j}), \vec{0}^2, \vec{\kappa}^{(h,j)}, 0)_{\mathbb{B}^*_{n'+j}}.$$
  6. It outputs $\text{SK}(f_h) = (f_h, \{\boldsymbol{k}'^{(h,j')}\}_{j'\in[m_h]}, \{\boldsymbol{k}^{(h,j)}\}_{j\in[n]})$.

■ Sequence of Hybrid Experiments

The hybrid experiments are described below. In the description of these hybrids, a part framed by a box indicates coefficients that are altered in a transition from its previous hybrid.

Hyb_0: This experiment corresponds to the experiment $\mathsf{Exp}_{\mathcal{H}}^{\text{PHPE,REAL}}(\lambda)$ defined in Section 2.6. Thus, in this experiment, the ciphertext queried by $\mathcal{H}$ corresponding to a pair of public-private attribute strings $(\vec{x},\vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ is generated as $\text{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota'\in[n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota\in[n]})$ such that

$$\begin{aligned} \boldsymbol{c}'^{(\iota')} &= (\omega(1, x_{\iota'}), \vec{0}^2, \vec{0}^2, \vec{0}^2, \varphi'_{\iota'})_{\mathbb{B}_{\iota'}} \text{ for } \iota' \in [n'], \\ \boldsymbol{c}^{(\iota)} &= (\omega(1, z_\iota), \vec{0}^2, \vec{0}^2, \vec{0}^2, \varphi_\iota)_{\mathbb{B}_{n'+\iota}} \text{ for } \iota \in [n], \end{aligned} \tag{3.1}$$

where $\omega, \{\varphi'_{\iota'}\}_{\iota'\in[n']}, \{\varphi_\iota\}_{\iota\in[n]} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, while for $h \in [q_{\text{KEY}}]$, the h^{th} decryption key queried by $\mathcal{H}$ corresponding to the function $f_h \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ is generated as $\text{SK}(f_h) = (f_h, \{\boldsymbol{k}'^{(h,j')}\}_{j'\in[m_h]}, \{\boldsymbol{k}^{(h,j)}\}_{j\in[n]})$ such that

$$\begin{aligned} \boldsymbol{k}'^{(h,j')} &= ((\gamma_{h,j'}, \alpha_{h,j'}), \vec{0}^2, \vec{0}^2, \vec{\kappa}'^{(h,j')}, 0)_{\mathbb{B}^*_{\rho_h(j')}} \text{ for } j' \in [m_h], \\ \boldsymbol{k}^{(h,j)} &= ((\sigma_{h,j}, \zeta_h), \vec{0}^2, \vec{0}^2, \vec{\kappa}^{(h,j)}, 0)_{\mathbb{B}^*_{n'+j}} \text{ for } j \in [n], \end{aligned} \tag{3.2}$$

where $\zeta_h \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\{\vec{\kappa}'^{(h,j')}\}_{j'\in[m_h]}, \{\vec{\kappa}^{(h,j)}\}_{j\in[n]} \xleftarrow{\mathsf{U}} \mathbb{F}_q^2$, $m_h + n + 1$ is the number of vertices in the ABP Γ'_h computing the function f_h as described in Section 2.3, and $\big((\{\sigma_{h,j}\}_{j\in[n]}, \{\alpha_{h,j'}, \gamma_{h,j'}\}_{j'\in[m_h]}), \rho_h : [m_h] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{PGB}(f_h)$. Here, $\{\mathbb{B}_\iota, \mathbb{B}^*_\iota\}_{\iota\in[n'+n]}$ is the collection of dual orthonormal bases generated by executing $\mathcal{G}_{\text{OB}}(n'+n, (0, \overbrace{9, \ldots, 9}^{n'+n}))$ during setup.

Hyb_1: This experiment is analogous to Hyb_0 except that in this experiment, the ciphertext queried by $\mathcal{H}$ corresponding to the pair of public-private attribute strings $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ is generated as $\mathrm{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota' \in [n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota \in [n]})$ such that

$$\begin{aligned} \boldsymbol{c}'^{(\iota')} &= (\omega(1, x_{\iota'}), (\boxed{\vartheta}, 0), \vec{0}^2, \vec{0}^2, \varphi'_{\iota'})_{\mathbb{B}_{\iota'}} \text{ for } \iota' \in [n'], \\ \boldsymbol{c}^{(\iota)} &= (\omega(1, z_\iota), (\boxed{\vartheta}, 0), \vec{0}^2, \vec{0}^2, \varphi_\iota)_{\mathbb{B}_{n'+\iota}} \text{ for } \iota \in [n], \end{aligned} \tag{3.3}$$

where $\vartheta \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and all the other variables are generated as in Hyb_0.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ ($\chi \in [q_{\text{KEY-PRE}}]$): The experiment $\mathsf{Hyb}_{2\text{-}0\text{-}4}$ coincides with Hyb_1. This experiment is analogous to $\mathsf{Hyb}_{2\text{-}(\chi-1)\text{-}4}$ except that in this experiment, the ciphertext queried by $\mathcal{H}$ corresponding to the pair of public-private attribute strings $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ is generated as $\mathrm{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota' \in [n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota \in [n]})$ such that

$$\begin{aligned} \boldsymbol{c}'^{(\iota')} &= (\omega(1, x_{\iota'}), \boxed{\tau(1, x_{\iota'}), \theta(1, x_{\iota'})}, \vec{0}^2, \varphi'_{\iota'})_{\mathbb{B}_{\iota'}} \text{ for } \iota' \in [n'], \\ \boldsymbol{c}^{(\iota)} &= (\omega(1, z_\iota), \boxed{\tau(1, z_\iota), \theta(1, s_\iota)}, \vec{0}^2, \varphi_\iota)_{\mathbb{B}_{n'+\iota}} \text{ for } \iota \in [n], \end{aligned} \tag{3.4}$$

where $\tau, \theta \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\vec{s} \xleftarrow{\mathsf{U}} S = \{\vec{s} \in \mathbb{F}_q^n \mid R^{\mathrm{ABP} \circ \mathrm{IP}}(f_h, (\vec{x}, \vec{s})) = R^{\mathrm{ABP} \circ \mathrm{IP}}(f_h, (\vec{x}, \vec{z})) \forall h \in [q_{\text{KEY-PRE}}]\}$, and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}(\chi-1)\text{-}4}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$ ($\chi \in [q_{\text{KEY-PRE}}]$): This experiment is the same as $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$ with the only exception that the χ^{th} decryption key queried by $\mathcal{H}$ corresponding to the function $f_\chi \in \mathcal{F}_{\mathrm{ABP} \circ \mathrm{IP}}^{(q,n',n)}$ is formed as $\mathrm{SK}(f_\chi) = (f_\chi, \{\boldsymbol{k}'^{(\chi,j')}\}_{j' \in [m_\chi]}, \{\boldsymbol{k}^{(\chi,j)}\}_{j \in [n]})$ such that

$$\begin{aligned} \boldsymbol{k}'^{(\chi,j')} &= ((\gamma_{\chi,j'}, \alpha_{\chi,j'}), \boxed{(\widetilde{\gamma}_{\chi,j'}, \widetilde{\alpha}_{\chi,j'})}, \vec{0}^2, \vec{\kappa}'^{(\chi,j')}, 0)_{\mathbb{B}^*_{\rho_\chi(j')}} \text{ for } j' \in [m_\chi], \\ \boldsymbol{k}^{(\chi,j)} &= ((\sigma_{\chi,j}, \zeta_\chi), \boxed{(\widetilde{\sigma}_{\chi,j}, \widetilde{\zeta}_\chi)}, \vec{0}^2, \vec{\kappa}^{(\chi,j)}, 0)_{\mathbb{B}^*_{n'+j}} \text{ for } j \in [n], \end{aligned} \tag{3.5}$$

where $\widetilde{\zeta}_\chi \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\big((\{\widetilde{\sigma}_{\chi,j}\}_{j \in [n]}, \{\widetilde{\alpha}_{\chi,j'}, \widetilde{\gamma}_{\chi,j'}\}_{j' \in [m_\chi]}), \rho_\chi : [m_\chi] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{PGB}(f_\chi)$, and all the other variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}1}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}3$ ($\chi \in [q_{\text{KEY-PRE}}]$): This experiment is analogous to $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$ except that in this experiment, the ciphertext queried by $\mathcal{H}$ for the pair of public-private attribute strings $(\vec{x}, \vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ is formed as $\mathrm{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota' \in [n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota \in [n]})$ such that $\{\boldsymbol{c}'^{(\iota')}\}_{\iota' \in [n']}$ are given by Eq. (3.4) and

$$\boldsymbol{c}^{(\iota)} = (\omega(1, z_\iota), \tau(1, \boxed{s_\iota}), \theta(1, s_\iota), \vec{0}^2, \varphi_\iota)_{\mathbb{B}_{n'+\iota}} \text{ for } \iota \in [n], \tag{3.6}$$

where all the variables are generated as in $\mathsf{Hyb}_{2\text{-}\chi\text{-}2}$.

$\mathsf{Hyb}_{2\text{-}\chi\text{-}4}$ ($\chi \in [q_{\text{KEY-PRE}}]$): This experiment is identical to $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$ except that the χ^{th} decryption key queried by $\mathcal{H}$ corresponding to the function $f_\chi \in \mathcal{F}_{\mathrm{ABP} \circ \mathrm{IP}}^{(q,n',n)}$ is generated as $\mathrm{SK}(f_\chi) = (f_\chi, \{\boldsymbol{k}'^{(\chi,j')}\}_{j' \in [m_\chi]}, \{\boldsymbol{k}^{(\chi,j)}\}_{j \in [n]})$ such that

$$\begin{aligned} \boldsymbol{k}'^{(\chi,j')} &= ((\gamma_{\chi,j'}, \alpha_{\chi,j'}), \boxed{\vec{0}^2, (\widehat{\gamma}_{\chi,j'}, \widehat{\alpha}_{\chi,j'})}, \vec{\kappa}'^{(\chi,j')}, 0)_{\mathbb{B}^*_{\rho_\chi(j')}} \text{ for } j' \in [m_\chi], \\ \boldsymbol{k}^{(\chi,j)} &= ((\sigma_{\chi,j}, \zeta_\chi), \boxed{\vec{0}^2, (\widehat{\sigma}_{\chi,j}, \widehat{\zeta}_\chi)}, \vec{\kappa}^{(\chi,j)}, 0)_{\mathbb{B}^*_{n'+j}} \text{ for } j \in [n], \end{aligned} \tag{3.7}$$

where $\widehat{\zeta}_\chi \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\big((\{\widehat{\sigma}_{\chi,j}\}_{j \in [n]}, \{\widehat{\alpha}_{\chi,j'}, \widehat{\gamma}_{\chi,j'}\}_{j' \in [m_\chi]}), \rho_\chi : [m_\chi] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{PGB}(f_\chi)$, and all the other variables are generated in the same manner as that in $\mathsf{Hyb}_{2\text{-}\chi\text{-}3}$.

Hyb_3: This experiment is analogous to $\mathsf{Hyb}_{2\text{-}q_{\text{KEY-PRE}}\text{-}4}$ except that in this experiment, the ciphertext queried by $\mathcal{H}$ for the pair of public-private attribute strings $(\vec{x},\vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ is formed as $\text{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota' \in [n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota \in [n]})$, where $\{\boldsymbol{c}'^{(\iota')}\}_{\iota' \in [n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota \in [n]}$ are given by Eq. (3.4), while for $h \in [q_{\text{KEY}}]$, the h^{th} decryption key queried by $\mathcal{H}$ for $f_h \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ is generated as $\text{SK}(f_h) = (f_h, \{\boldsymbol{k}'^{(h,j')}\}_{j' \in [m_h]}, \{\boldsymbol{k}^{(h,j)}\}_{j \in [n]})$ such that

$$\boldsymbol{k}'^{(h,j')} = \begin{cases} ((\gamma_{h,j'}, \alpha_{h,j'}), \boxed{(\widetilde{\gamma}_{h,j'}, \widetilde{\alpha}_{h,j'})}, (\widehat{\gamma}_{h,j'}, \widehat{\alpha}_{h,j'}), \vec{\kappa}'^{(h,j')}, 0)_{\mathbb{B}^*_{\rho_h(j')}} \\ \qquad\qquad \text{for } j' \in [m_h] \text{ if } h \in [q_{\text{KEY-PRE}}], \\ ((\gamma_{h,j'}, \alpha_{h,j'}), \boxed{(\widetilde{\gamma}_{h,j'}, \widetilde{\alpha}_{h,j'})}, \vec{0}^2, \vec{\kappa}'^{(h,j')}, 0)_{\mathbb{B}^*_{\rho_h(j')}} \\ \qquad\qquad \text{for } j' \in [m_h] \text{ if } h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}], \end{cases}$$

$$\boldsymbol{k}^{(h,j)} = \begin{cases} ((\sigma_{h,j}, \zeta_h), \boxed{(\widetilde{\sigma}_{h,j}, \widetilde{\zeta}_h)}, (\widehat{\sigma}_{h,j}, \widehat{\zeta}_h), \vec{\kappa}^{(h,j)}, 0)_{\mathbb{B}^*_{n'+j}} \\ \qquad\qquad \text{for } j \in [n] \text{ if } h \in [q_{\text{KEY-PRE}}], \\ ((\sigma_{h,j}, \zeta_h), \boxed{(\widetilde{\sigma}_{h,j}, \widetilde{\zeta}_h)}, \vec{0}^2, \vec{\kappa}^{(h,j)}, 0)_{\mathbb{B}^*_{n'+j}} \\ \qquad\qquad \text{for } j \in [n] \text{ if } h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}], \end{cases} \tag{3.8}$$

where $\{\widetilde{\zeta}_h\}_{h \in [q_{\text{KEY}}]} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\big((\{\widetilde{\sigma}_{h,j}\}_{j \in [n]}, \{\widetilde{\alpha}_{h,j'}, \widetilde{\gamma}_{h,j'}\}_{j' \in [m_h]}), \rho_h : [m_h] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{PGB}(f_h)$ for $h \in [q_{\text{KEY}}]$, and all the other variables are formed as in $\mathsf{Hyb}_{2\text{-}q_{\text{KEY-PRE}}\text{-}4}$.

Hyb_4: This experiment is analogous to Hyb_3 except that in this experiment, the ciphertext queried by $\mathcal{H}$ for the pair of public-private attribute strings $(\vec{x},\vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ is generated as $\text{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota' \in [n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota \in [n]})$ such that

$$\begin{aligned} \boldsymbol{c}'^{(\iota')} &= (\boxed{\vec{0}^2}, \tau(1, x_{\iota'}), \theta(1, x_{\iota'}), \vec{0}^2, \varphi'_{\iota'})_{\mathbb{B}_{\iota'}} \text{ for } \iota' \in [n'], \\ \boldsymbol{c}^{(\iota)} &= (\boxed{\vec{0}^2}, \tau(1, z_\iota), \theta(1, s_\iota), \vec{0}^2, \varphi_\iota)_{\mathbb{B}_{n'+\iota}} \text{ for } \iota \in [n], \end{aligned} \tag{3.9}$$

where all the variables are generated as in Hyb_3.

Hyb_5: This experiment is identical to Hyb_4 except that in this experiment, for $h \in [q_{\text{KEY-PRE}}]$, the h^{th} decryption key queried by $\mathcal{H}$ corresponding to the function $f_h \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ is generated as $\text{SK}(f_h) = (f_h, \{\boldsymbol{k}'^{(h,j')}\}_{j' \in [m_h]}, \{\boldsymbol{k}^{(h,j)}\}_{j \in [n]})$ such that $\{\boldsymbol{k}'^{(h,j')}\}_{j' \in [m_h]}$ and $\{\boldsymbol{k}^{(h,j)}\}_{j \in [n]}$ are given by Eq. (3.7).

Hyb_6: This experiment is the same as Hyb_5 except that in this experiment, the ciphertext queried by $\mathcal{H}$ for the pair of public-private attribute strings $(\vec{x},\vec{z}) \in \mathbb{F}_q^{n'} \times \mathbb{F}_q^n$ is generated as $\text{CT} = (\vec{x}, \{\boldsymbol{c}'^{(\iota')}\}_{\iota' \in [n']}, \{\boldsymbol{c}^{(\iota)}\}_{\iota \in [n]})$ such that

$$\begin{aligned} \boldsymbol{c}'^{(\iota')} &= (\vec{0}^2, \boxed{(0,\tau)}, \theta(1, x_{\iota'}), \vec{0}^2, \varphi'_{\iota'})_{\mathbb{B}_{\iota'}} \text{ for } \iota' \in [n'], \\ \boldsymbol{c}^{(\iota)} &= (\vec{0}^2, \boxed{(0,\tau)}, \theta(1, s_\iota), \vec{0}^2, \varphi_\iota)_{\mathbb{B}_{n'+\iota}} \text{ for } \iota \in [n], \end{aligned} \tag{3.10}$$

while for $h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}]$, the h^{th} decryption key queried by $\mathcal{H}$ for $f_h \in \mathcal{F}_{\text{ABP}\circ\text{IP}}^{(q,n',n)}$ is generated as $\text{SK}(f_h) = (f_h, \{\boldsymbol{k}'^{(h,j')}\}_{j' \in [m_h]}, \{\boldsymbol{k}^{(h,j)}\}_{j \in [n]})$ such that

$$\begin{aligned} \boldsymbol{k}'^{(h,j')} &= ((\gamma_{h,j'}, \alpha_{h,j'}), \boxed{(\widetilde{\gamma}_{h,j'}, \widetilde{\alpha}_{h,j'})\boldsymbol{U}'^{(\rho_h(j'))}}, \vec{0}^2, \vec{\kappa}'^{(h,j')}, 0)_{\mathbb{B}^*_{\rho_h(j')}} \text{ for } j' \in [m_h], \\ \boldsymbol{k}^{(h,j)} &= ((\sigma_{h,j}, \zeta_h), \boxed{(\widetilde{\sigma}_{h,j}, \widetilde{\zeta}_h)\boldsymbol{U}^{(j)}}, \vec{0}^2, \vec{\kappa}^{(h,j)}, 0)_{\mathbb{B}^*_{n'+j}} \text{ for } j \in [n], \end{aligned} \tag{3.11}$$

where $\boldsymbol{Z}'^{(\iota')} \xleftarrow{\mathsf{U}} \{\boldsymbol{Z} \in \mathsf{GL}(2,\mathbb{F}_q) \mid (1,x_{\iota'})\boldsymbol{Z} = \vec{e}^{(2)} = (0,1)\}$, $\boldsymbol{U}'^{(\iota')} = ((\boldsymbol{Z}'^{(\iota')})^{-1})^{\intercal}$ for $\iota' \in [n']$, $\boldsymbol{Z}^{(\iota)} \xleftarrow{\mathsf{U}} \{\boldsymbol{Z} \in \mathsf{GL}(2,\mathbb{F}_q \mid (1,z_\iota)\boldsymbol{Z} = \vec{e}^{(2)} = (0,1)\}$, $\boldsymbol{U}^{(\iota)} = ((\boldsymbol{Z}^{(\iota)})^{-1})^{\intercal}$ for $\iota \in [n]$, and all the other variables are generated as in Hyb_5.

Hyb_7: This experiment is identical to Hyb_6 with the only exception that for $h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}]$, the h^{th} decryption key queried by $\mathcal{H}$ corresponding to the function $f_h \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$ is generated as $\text{SK}(f_h) = (f_h, \{\boldsymbol{k}'^{(h,j')}\}_{j' \in [m_h]}, \{\boldsymbol{k}^{(h,j)}\}_{j \in [n]})$ such that

$$\boldsymbol{k}'^{(h,j')} = ((\gamma_{h,j'}, \alpha_{h,j'}), \boxed{(\eta'_{h,j'}, \widetilde{\alpha}_{h,j'} x_{\rho_h(j')} + \widetilde{\gamma}_{h,j'})}, \vec{0}^2, \vec{\kappa}'^{(h,j')}, 0)_{\mathbb{B}^*_{\rho_h(j')}} \quad \text{for } j' \in [m_h], \tag{3.12}$$

$$\boldsymbol{k}^{(h,j)} = ((\sigma_{h,j}, \zeta_h), \boxed{(\eta_{h,j}, \widetilde{\zeta}_h z_j + \widetilde{\sigma}_{h,j})}, \vec{0}^2, \vec{\kappa}^{(h,j)}, 0)_{\mathbb{B}^*_{n'+j}} \text{ for } j \in [n],$$

where $\{\eta'_{h,j'}\}_{h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}], j' \in [m_h]}, \{\eta_{h,j}\}_{h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}], j \in [n]} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and all the other variables are generated as in Hyb_6.

Hyb_8: This experiment is analogous to Hyb_7 with the only exception that for $h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}]$, the h^{th} decryption key queried by $\mathcal{H}$ corresponding to the function $f_h \in \mathcal{F}^{(q,n',n)}_{\text{ABP}\circ\text{IP}}$ is formed as $\text{SK}(f_h) = (f_h, \{\boldsymbol{k}'^{(h,j')}\}_{j' \in [m_h]}, \{\boldsymbol{k}^{(h,j)}\}_{j \in [n]})$ such that

$$\begin{aligned} \boldsymbol{k}'^{(h,j')} &= ((\gamma_{h,j'}, \alpha_{h,j'}), (\eta'_{h,j'}, \boxed{\mu_{h,j'}}), \vec{0}^2, \vec{\kappa}'^{(h,j')}, 0)_{\mathbb{B}^*_{\rho_h(j')}} \text{ for } j' \in [m_h], \\ \boldsymbol{k}^{(h,j)} &= ((\sigma_{h,j}, \zeta_h), (\eta_{h,j}, \boxed{\nu_{h,j}}), \vec{0}^2, \vec{\kappa}^{(h,j)}, 0)_{\mathbb{B}^*_{n'+j}} \text{ for } j \in [n], \end{aligned} \tag{3.13}$$

where $\{\widetilde{\zeta}_h\}_{h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}]} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\big(\{\nu_{h,j}\}_{j \in [n]}, \{\mu_{h,j'}\}_{j' \in [m_h]}), \rho_h : [m_h] \to [n']\big) \xleftarrow{\mathsf{R}} \mathsf{SIM}(f_h, \vec{x}, f_h(\vec{x}, \widetilde{\zeta}_h \vec{z}))$ for $h \in [q_{\text{KEY-PRE}}+1, q_{\text{KEY}}]$, and all the other variables are generated as in Hyb_7. Observe that for any $h \in [q_{\text{KEY-PRE}}]+1, q_{\text{KEY}}]$ if $R^{\text{ABP}\circ\text{IP}}(f_h, (\vec{x}, \vec{z})) = 1$, i.e., $f_h(\vec{x}, \vec{z}) = 0$, then $f_h(\vec{x}, \widetilde{\zeta}_h \vec{z}) = \widetilde{\zeta}_h f_h(\vec{x}, \vec{z}) = 0$, while if $R^{\text{ABP}\circ\text{IP}}(f_h(\vec{x}, \vec{z})) = 0$, i.e., $f_h(\vec{x}, \vec{z}) \neq 0$, then due to the uniform and independent (of the other variables) choice of $\widetilde{\zeta}_h$, it follows that $f_h(\vec{x}, \widetilde{\zeta}_h \vec{z}) = \widetilde{\zeta}_h f_h(\vec{x}, \vec{z})$ is uniformly and independently (of the other variables) distributed in $\mathbb{F}_q$. Thus, this experiment coincides with the experiment $\mathsf{Exp}^{\text{PHPE,IDEAL}}_{\mathcal{H},\mathcal{S}}(\lambda)$ with the simulator $\mathcal{S}$ as described above.

■ **Analysis**

Let us now denote by $\mathsf{Adv}^{(\jmath)}_{\mathcal{H}}(\lambda)$ the probability that $\mathcal{H}$ outputs 1 in $\mathsf{Hyb}_\jmath$ for $\jmath \in \{0, 1, \{2\text{-}\chi\text{-}k\}_{\chi \in [q_{\text{KEY-PRE}}], k \in [4]}, 3, \ldots, 8\}$. By definition of the hybrids, we clearly have $\mathsf{Adv}^{\text{PHPE-SIM-AH}}_{\mathcal{H}}(\lambda) = |\mathsf{Adv}^{(0)}_{\mathcal{H}}(\lambda) - \mathsf{Adv}^{(8)}_{\mathcal{H}}(\lambda)|$. Hence, we have

$$\begin{aligned} \mathsf{Adv}^{\text{PHPE,SIM-AH}}_{\mathcal{H}}(\lambda) \leq\ & |\mathsf{Adv}^{(0)}_{\mathcal{H}}(\lambda) - \mathsf{Adv}^{(1)}_{\mathcal{H}}(\lambda)| + \\ & \sum_{\chi \in [q_{\text{KEY-PRE}}]} \Big[|\mathsf{Adv}^{(2\text{-}(\chi-1)\text{-}4)}_{\mathcal{H}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}1)}_{\mathcal{H}}(\lambda)| + \\ & \sum_{k \in [3]} |\mathsf{Adv}^{(2\text{-}\chi\text{-}k)}_{\mathcal{H}}(\lambda) - \mathsf{Adv}^{(2\text{-}\chi\text{-}(k+1))}_{\mathcal{H}}(\lambda)| \Big] + \\ & |\mathsf{Adv}^{(2\text{-}q_{\text{KEY-PRE}}\text{-}4)}_{\mathcal{H}}(\lambda) - \mathsf{Adv}^{(3)}_{\mathcal{H}}(\lambda)| + \sum_{\jmath \in [3,7]} |\mathsf{Adv}^{(\jmath)}_{\mathcal{H}}(\lambda) - \mathsf{Adv}^{(\jmath+1)}_{\mathcal{H}}(\lambda)|. \end{aligned}$$

It can be shown that each term on the RHS of the above equation is negligible in λ, and hence Theorem 3.1 follows. The details are provided in the full version.

References

1. Abe, M., Chase, M., David, B., Kohlweiss, M., Nishimaki, R., Ohkubo, M.: Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In: ASIACRYPT 2012. pp. 4–24. Springer
2. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: CRYPTO 2017. pp. 3–35. Springer
3. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: New perspectives and lower bounds. In: CRYPTO 2013. pp. 500–518. Springer
4. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: CRYPTO 2015. pp. 657–677. Springer
5. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: CRYPTO 2015. pp. 308–326. Springer
6. Ananth, P., Jain, A., Sahai, A.: Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730
7. Attrapadung, N.: Dual system encryption via doubly selective security : Framework, fully secure functional encryption for regular languages, and more. In: EUROCRYPT 2014. pp. 557–577. Springer
8. Barbulescu, R., Gaudry, P., Joux, A., Thomé, E.: A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In: EUROCRYPT 2014. pp. 1–16. Springer
9. Bellare, M., O'Neill, A.: Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In: CANS 2013. pp. 218–234. Springer
10. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: FOCS 2015. pp. 171–190. IEEE
11. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In: EUROCRYPT 2014. pp. 533–556. Springer
12. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: TCC 2011. pp. 253–273. Springer
13. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: TCC 2007. pp. 535–554. Springer
14. Chen, J., Gay, R., Wee, H.: Improved dual system abe in prime-order groups via predicate encodings. In: EUROCRYPT 2015. pp. 595–624. Springer
15. Chen, J., Wee, H.: Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In: SCN 2014. pp. 277–297. Springer
16. De Caro, A., Iovino, V., Jain, A., O'Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. In: CRYPTO 2013. pp. 519–535. Springer
17. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: EUROCRYPT 2010. pp. 44–61. Springer
18. Göloğlu, F., Granger, R., McGuire, G., Zumbrägel, J.: On the function field sieve and the impact of higher splitting probabilities. In: CRYPTO 2013, pp. 109–128. Springer
19. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from lwe. In: CRYPTO 2015. pp. 503–523. Springer

20. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. Journal of the ACM (JACM) 62(6) (2015)
21. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: TCC 2016. pp. 361–388. Springer
22. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006. pp. 89–98. ACM
23. Guillevic, A.: Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In: ACNS 2013. pp. 357–372. Springer
24. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: ICALP 2002. pp. 244–256. Springer
25. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: Theory of Computing and Systems, 1997, Proceedings of the Fifth Israeli Symposium on. pp. 174–184. IEEE
26. Ishai, Y., Wee, H.: Partial garbling schemes and their applications. In: ICALP 2014. pp. 650–662. Springer
27. Joux, A.: Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In: EUROCRYPT 2013. pp. 177–193. Springer
28. Joux, A.: A new index calculus algorithm with complexity $l(1/4+o(1))$ in small characteristic. In: SAC 2013. pp. 355–379. Springer
29. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: EUROCRYPT 2008. pp. 146–162. Springer
30. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: EUROCRYPT 2010. pp. 62–91. Springer
31. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: Achieving full security through selective techniques. In: CRYPTO 2012. pp. 180–198. Springer
32. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: EUROCRYPT 2012. pp. 591–608. Springer
33. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: CRYPTO 2010. pp. 191–208. Springer
34. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: ASIACRYPT 2012. pp. 349–366. Springer
35. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: ASIACRYPT 2009. pp. 214–231. Springer
36. Okamoto, T., Takashima, K.: Efficient (hierarchical) inner-product encryption tightly reduced from the decisional linear assumption. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 96(1), 42–52 (2013)
37. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556
38. Tomida, J., Abe, M., Okamoto, T.: Efficient functional encryption for inner-product values with full-hiding security. In: ISC 2016. pp. 408–425. Springer
39. Waters, B.: Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In: CRYPTO 2009. pp. 619–636. Springer
40. Wee, H.: Attribute-hiding predicate encryption in bilinear groups, revisited. In: TCC 2017. pp. 206–233. Springer
41. Wee, H.: Dual system encryption via predicate encodings. In: TCC 2014. pp. 616–637. Springer