Encrypt-Augment-Recover: Computationally Function Private Predicate Encryption in the Public-Key Setting

Blinded for Review

Abstract. We solve the open problem of constructing *computationally function private* public-key predicate encryption schemes. Existing public-key constructions for predicate encryption satisfy a *statistical* notion of function privacy, that was introduced for equality predicates by Boneh, Raghunathan and Segev in CRYPTO'13, and was generalized for subspace-membership predicates in ASIACRYPT'13. The secret-keys in these constructions are *statistically indistinguishable* from random as long the underlying predicates are sampled from sufficiently unpredictable distributions. The alternative notion of computational function privacy, where the secret-keys are *computationally indistinguishable* from random, has only been concretely realized in the private-key setting, to the best of our knowledge.

In this paper, we present the first computationally function private constructions for public-key predicate encryption. Our framework for computational function privacy requires that a secret-key corresponding to a predicate sampled from a distribution with min-entropy super logarithmic in the security parameter λ, is *computationally indistinguishable* from another secret-key corresponding to a uniformly and independently sampled predicate. Within this framework, we develop a novel approach, denoted as *encrypt-augment-recover*, that takes an existing predicate encryption scheme and transforms it into a computationally function private one while retaining its original data privacy guarantees. Our approach leads to public-key constructions for identity-based encryption and inner-product encryption that are fully data private and computationally function private under a family of weaker variants of the DLIN assumption. Our constructions, in fact, satisfy an *enhanced* notion of function privacy, requiring that an adversary learns nothing more than the minimum necessary from a secret-key, even given corresponding ciphertexts with attributes that allow successful decryption.

Keywords: Predicate Encryption, Public-Key, Function Privacy, Computational Indistinguishability, Min-Entropy, Identity-Based Encryption, Inner-Product Encryption

1 Introduction

Predicate encryption systems [1–3] in the public-key setting allow a single public-key to be associated with multiple secret-keys, where each secret-key corresponds to a boolean predicate $f : \Sigma \longrightarrow \{0,1\}$ over a pre-defined set of attributes Σ. A plaintext message in a predicate encryption system is an attribute-payload message pair $(I, M) \in \Sigma \times \mathcal{M}$, with $\mathcal{M}$ being the payload message space. A secret-key sk_f associated with a predicate f successfully decrypts a ciphertext C corresponding to a plaintext (I, M) and recovers the payload message M if and only if $f(I) = 1$. On the other hand, if $f(I) = 0$, attempting to decrypt C using sk_f returns the failure symbol $\perp$. A predicate encryption is said to be *attribute hiding* if the ciphertext C leaks no information about the underlying plaintext (I, M) to an adversary possessing benign secret-keys corresponding to predicates that do not trivially identify the attribute I.

Identity-Based Encryption. Identity-based encryption (IBE) [4–6] is the simplest sub-class of public-key predicate encryption. IBE supports a set of equality predicates of the form $f_{\mathsf{id}} : \Sigma \longrightarrow \{0,1\}$ defined as $f_{\mathsf{id}}(x) = 1$ if and only if $x = \mathsf{id}$. The attribute space in this case is a set of identities $\mathcal{ID}$, and each identity $\mathsf{id} \in \mathcal{ID}$ is associated with its own secret-key $\mathsf{sk}_{\mathsf{id}}$.

Inner-Product Encryption. Inner-product encryption (IPE) [2, 3, 7, 8] is the most expressive sub-class of predicate encryption, supporting a set of predicates $f_{\overrightarrow{v}} : \Sigma \longrightarrow \{0,1\}$ over a vector space of attributes $\Sigma = \mathbb{F}_q^n$ (q being a λ-bit prime). Of particular interest is a specific form of IPE called zero-IPE [3] where for $\overrightarrow{v}, \overrightarrow{x} \in \Sigma$, we have $f_{\overrightarrow{v}}(\overrightarrow{x}) = 1$ if and only if $\langle \overrightarrow{v}, \overrightarrow{x} \rangle = 0$ (here $\langle \overrightarrow{v}, \overrightarrow{x} \rangle$ denotes the standard inner-product of two vectors $\overrightarrow{v}$ and $\overrightarrow{x}$). IPE is powerful enough to encompass IBE and many other predicate encryption systems [3].

Searchable Encryption and Function Privacy. Predicate encryption provides a generic framework for searchable encryption supporting a wide range of query predicates including conjunctive, disjunctive, range and comparison queries [9, 1–3]. For instance, a predicate encryption system can be used to realize a mail gateway that follows some special instructions to route encrypted mails based on their header information (e.g. if the mail is from the boss and needs to be treated as urgent). The mail gateway is given the secret-key corresponding to the predicate *is-urgent*, the mail header serves as the attribute, while the routing instructions can be used as the payload message. Another application could be a payment gateway that flags encrypted payments if they correspond to amounts beyond some pre-defined threshold X. The payment gateway is given the secret-key corresponding to the predicate *greater-than-*X, the payment amount itself serves as the attribute, while the flag signal is encoded as the payload message. The attribute hiding property of the predicate encryption scheme ensures that neither gateway learns any information about the plaintext data from the entire operation.

A natural question now arises: should the gateways in the aforementioned examples be able to learn the underlying predicate from the secret-keys given to them? The answer in most scenarios is *no* - the secret-key sk_f should ideally reveal nothing about the predicate f beyond the absolute minimum. This notion of predicate hiding security is commonly referred to as *function privacy*, and predicate encryption scheme satisfying this notion of security are described as *function private.*

Function Privacy in the Public-Key Setting. As pointed out by Boneh, Raghunathan and Segev in [10, 11], formalizing a realistic notion of function privacy in the context of public-key predicate encryption is, in general, not straightforward. Consider, for example, an adversary against an IBE scheme who is given a secret-key $\mathsf{sk}_{\mathsf{id}}$ corresponding to an identity id and has access to an encryption oracle. As long as the adversary has some apriori information that the identity id belongs to a small set $\mathcal{S}$,(e.g. id is sampled distribution with min-entropy at most polynomial in the security parameter λ), it can fully recover id from $\mathsf{sk}_{\mathsf{id}}$: it can simply resort to encrypting a random message M under each identity in $\mathcal{S}$, and decrypting using $\mathsf{sk}_{\mathsf{id}}$ to check for a correct recovery. Consequently, [10, 11] consider a framework for function privacy under the minimal assumption that any predicate is sampled from a distribution with min-entropy at least super logarithmic in the security parameter λ.

Existing Function Private Constructions. Existing public-key constructions for predicate encryption satisfy a *statistical* notion of function privacy, that was introduced for equality predicates in [10], and was subsequently generalized for subspace-membership predicates in [11]. In particular, the secret-keys in these constructions are *statistically indistinguishable* from random, as long the underlying predicates are sampled from sufficiently unpredictable distributions. The alternative notion of computational function privacy, where the secret-keys are *computationally indistinguishable* from random, has only been concretely realized in the private-key setting [12–15] to the best of our knowledge. In fact, designing public-key predicate encryption schemes whose function privacy can be based on well-known computational assumptions, was left as an open problem in [11].

1.1 Our Contributions

In this paper, we present the first computationally function private constructions for public-key predicate encryption. Our framework for computational function privacy, presented formally in Section 3, requires that a secret-key corresponding to a predicate sampled from a distribution with min-entropy super logarithmic in the security parameter λ, is *computationally indistinguishable* from another secret-key corresponding to a uniformly and independently sampled predicate. Within this framework, we develop a novel approach, denoted as *encrypt-augment-recover*, that takes an existing predicate encryption scheme and transforms it into a computationally function private one. Our approach leads to the following constructions:

- In the random-oracle model, we present a family of identity-based encryption schemes from bilinear pairings based on the scheme of Boneh and Franklin [4]. Our schemes retain the adaptive data privacy of the underlying scheme (based on the same complexity assumption), and are computationally function private under progressively weaker variants of the well-known DLIN assumption. The detailed construction of these schemes, along with the proofs of data and function privacy, is presented in Section 4.

- In the standard model, we present a family of inner-product encryption schemes from bilinear pairings based on the scheme of Katz, Sahai and Waters [3]. Once again, our schemes retain the selectively attribute hiding property of the underlying scheme, and are computationally function private under progressively weaker variants of the DLIN assumption. The detailed construction of these schemes, along with the proofs of data and function privacy, is presented in Section 5.

- Our constructions, in fact, satisfy an *enhanced* notion of function privacy, requiring that an adversary learns nothing more than the minimum necessary from a secret-key, even given corresponding ciphertexts with attributes that allow successful decryption.

1.2 Overview of Our Approach: *Encrypt-Augment-Remove*

Our approach for achieving computationally function private predicate encryption schemes consists of three main steps - *encrypt*, *augment* and *recover*. We briefly describe the main ideas underlying each step, and exemplify them subsequently using a simple IBE scheme. Given a public-key predicate encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, and a CPA-secure public-key encryption algorithm $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, we create a function private predicate encryption scheme $\Pi' = \left(\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}'\right)$ as follows:

- The modified setup algorithm Setup' invokes $\mathsf{PKE.KeyGen}$ and obtains the key pair (PK, SK). It also invokes $\Pi.\mathsf{Setup}$ and obtains the public parameters pp, along with master secret-key

msk. It outputs the modified public parameter $\mathsf{pp}' = (\mathsf{pp}, g(SK))$ and the modified master secret-key $\mathsf{msk}' = (\mathsf{msk}, PK)$, where g is a suitably chosen one way function.

- On input a predicate f and the augmented master secret-key $\mathsf{msk}' = (\mathsf{msk}, PK)$, the modified key-generation algorithm KeyGen' invokes $\Pi.\mathsf{KeyGen}$ to obtain the original secret-key sk_f. It then outputs an *encrypted* secret-key sk'_f as $\mathsf{PKE.Enc}(PK, \mathsf{sk}_f)$.

  This step automatically guarantees computational function privacy - any adversary that can distinguish the augmented secret-key from random must break the CPA security guarantee of the PKE scheme. More specifically, it ensures *adaptive* function privacy - the inherently random nature of the augmented key generation algorithm ensures that the function privacy guarantees hold even when the adversary is allowed to specify predicate distributions in an adaptive manner after seeing the public parameters of the scheme. We assume that any adversarially-chosen distribution of predicates is sufficiently unpredictable, so as to rule out a trivial breach of function privacy as mentioned earlier. This minimal assumption is thus sufficient to transform the original predicate encryption scheme into a computationally function private one.

- An even greater challenge is to synchronize the encryption and decryption algorithms in the modified scheme. This is achieved as follows. On input the public parameter $\mathsf{pp}' = (\mathsf{pp}, g(SK))$, and a message M corresponding to an attribute I, the modified encryption algorithm Enc' first obtains $C = \Pi.\mathsf{Enc}(\mathsf{pp}, I, M)$. It then outputs the *augmented* ciphertext $C' = (C, \sigma(g(SK)))$, where σ is a randomized function sharing a source of randomness with $\Pi.\mathsf{Enc}$.

- Finally, Dec' cleverly uses the additional ciphertext component $\sigma(g(SK))$ in C' to remove the effect of PKE from the *encrypted* secret-key sk'_f, and *recovers* the message M. Note that *removal* here is not same as decryption, since Dec' has access to only a one-way function of SK and not SK itself. It is, in fact, impossible to provide SK to Dec' in the clear without trivially compromising function privacy. The challenge is thus to ensure that Dec' can recover M without a complete decryption of sk'_f.

Comparison with a deterministic public-key encryption-based approach. An alternative approach for designing computationally function private identity-based encryption schemes, suggested in [10], is as follows: encrypt all identities using a DPKE scheme, and use any existing anonymous IBE scheme that treats the corresponding ciphertexts of the DPKE scheme as its identities. Since the security of any deterministic public-key encryption (DPKE) algorithm is also based on the minimal assumption that its plaintexts are sampled from a distribution with a certain amount of min-entropy, the above approach seems quite natural. However, this approach suffers from two inherent drawbacks:

1. In the setting of DPKE, the dependency of plaintexts on the public-key of the scheme is essentially limited. Intuitively, the reason is as follows: in a deterministic encryption setting, plaintext distributions can be chosen depending on the public-key such that the encryption algorithm acts as a subliminal channel for leaking information, thus trivially violating all security guarantees [16]. However, this is too restrictive a security notion to be adopted in the context of IBE, where the key-generation process is allowed to be randomized. In particular, any realistic function privacy framework for IBE must allow adversaries to *adaptively*

specify challenge identity distributions, after they have seen the public parameters of the scheme.

2. The DPKE-based approach is not directly generalizable for functionally richer predicates unless the underlying encryption scheme is somewhat function-preserving. For example, if the above approach were to be extended in the context of IPE, it would require the ciphertexts of the DPKE scheme to preserve the orthogonality of the underlying plaintext vectors. This could potentially weaken the function privacy guarantees even further.

Our approach overcomes these limitations by focusing on encrypting the secret-key sk_f of the original predicate encryption scheme rather than the underlying predicate f. The augmented key-generation process uses a PKE instead of a DPKE, and is hence allowed to be non-deterministic. This makes our notion of function privacy more realistic since it does not restrict adaptive choice of predicate distributions on part of the adversary. Finally, our approach is generally applicable to a large class of predicate encryption schemes, including IBE and IPE, without any additional constraints on the properties of the underlying PKE.

An Example of Our Approach. We present an example of a computationally function private IBE scheme in the random-oracle model achieved using our *encrypt-augment-decrypt* approach. A generalization of this scheme is presented in greater detail in Section 4, along with proofs for data and function privacy. Consider a public-key encryption scheme PKE with the key generation, encryption and decryption algorithms as described below:

- **KeyGen:** The key-generation algorithm samples $x_1, x_2, x_3 \xleftarrow{R} \mathbb{Z}_q^*$, where q is a λ-bit prime, and $g_1, g_2, g_3 \xleftarrow{R} \mathbb{G}$, where $\mathbb{G}$ is a cyclic group of prime order q. It outputs the secret-key $SK = (x_1, x_2, x_3)$ and the public-key $PK = (g_1, g_2, g_3, (g_1^{x_1} \cdot g_3^{x_3}), (g_2^{x_2} \cdot g_3^{x_3}))$.

- **Enc:** The ciphertext C corresponding to a message $M \in \mathbb{G}$ is a tuple of the form:

$$C = \left(g_1^{y_1}, g_2^{y_2}, g_3^{y_1+y_2}, (g_1^{x_1} \cdot g_3^{x_3})^{y_1} \cdot (g_2^{x_2} \cdot g_3^{x_3})^{y_2} \cdot M\right)$$

  where $y_1, y_2 \xleftarrow{R} \mathbb{Z}_q^*$.

- **Dec:** The decryption algorithm, on input the ciphertext $C = (c_1, c_2, c_3, c_4)$ and the secret-key (x_1, x_2, x_3), recovers the message M as:

$$M = c_4 \Big/ (c_1^{x_1} \cdot c_2^{x_2} \cdot c_3^{x_3})$$

The above scheme is a simple variant of the Cramer-Shoup cryptosystem [17], and is CPA-secure under the DLIN assumption. We now present a computationally function private IBE scheme that is obtained by applying our *encrypt-augment-recover* approach to the anonymous IBE scheme based on bilinear maps proposed by Boneh and Franklin [4].

- **Setup:** The setup algorithm in the scheme of Boneh and Franklin samples $s \xleftarrow{R} \mathbb{Z}_q^*$, where q is a λ-bit prime. The public parameters are g and g^s, where g is the generator of a bilinear group $\mathbb{G}$ of prime order q, while the master secret-key is s. Our scheme additionally samples $x_1, x_2, x_3 \xleftarrow{R} \mathbb{Z}_q^*$ and $g_1, g_2, g_3 \xleftarrow{R} \mathbb{G}$. The augmented public parameter pp and master secret-key msk for our scheme are as follows:

$$\mathsf{pp} = \left(g, g^s, g^{x_1}, g^{x_2}, g^{x_3}\right)$$
$$\mathsf{msk} = \left(s, g_1, g_2, g_3, \left(g_1^{x_1} \cdot g_3^{x_3}\right), \left(g_2^{x_2} \cdot g_3^{x_3}\right)\right)$$

Observe that the additional components in pp are one-way functions of x_1, x_2, x_3 - the secret-key SK of the PKE scheme. Additionally, the modified msk contains the public-key PK of the PKE scheme. This is exactly in accordance with our proposed approach.

- **KeyGen:** The key-generation algorithm in the scheme of Boneh and Franklin computes a secret-key for an identity id as $\mathsf{sk}_{\mathsf{id}} = \left(H\left(\mathsf{id}\right)\right)^s$, where H is a random oracle mapping identities onto the group $\mathbb{G}$. In our scheme, we augment the key generation process as follows. We sample $y_1, y_2 \xleftarrow{R} \mathbb{Z}_q^*$, and output:

  $$\mathsf{sk}_{\mathsf{id}} = \left(g_1^{y_1}, g_2^{y_2}, g_3^{y_1+y_2}, \left(g_1^{x_1} \cdot g_3^{x_3}\right)^{y_1} \cdot \left(g_2^{x_2} \cdot g_3^{x_3}\right)^{y_2} \cdot \left(H\left(\mathsf{id}\right)\right)^s\right)$$

  Observe that $\mathsf{sk}_{\mathsf{id}} = \mathsf{PKE.Enc}\left(PK, \left(H\left(\mathsf{id}\right)\right)^s\right)$, which is a direct exemplification of the *encrypt* step in our approach described above. The reader is referred to Section 4 for the detailed proof of function privacy.

- **Enc:** An encryption of a message M for an identity id in the scheme of Boneh and Franklin is a tuple of the form $\left(g^r, M \cdot e\left(H\left(\mathsf{id}\right), g^s\right)^r\right)$, where $r \xleftarrow{R} \mathbb{Z}_q^*$. In our scheme, we augment the encryption process to produce the ciphertext:

  $$C = \left(g^r, \left(g^{x_1}\right)^r, \left(g^{x_2}\right)^r, \left(g^{x_3}\right)^r, M \cdot e\left(H\left(\mathsf{id}\right), g^s\right)^r\right)$$

  Note that the augmented ciphertext in our scheme retains unaltered the original ciphertext. The main technical challenge is to prove that such an augmented ciphertext still provides the same data privacy guarantees as the original scheme of Boneh and Franklin (the reader is referred to Section 4 for the detailed proof).

- **Dec:** Our decryption algorithm, on input of a ciphertext $C = (c_0, c_1, c_2, c_3, c_4)$, and a secret-key $\mathsf{sk}_{\mathsf{id}} = (d_0, d_1, d_2, d_3)$, recovers the encrypted message M as:

  $$M = c_4 \cdot \frac{e\left(d_0, c_1\right) \cdot e\left(d_1, c_2\right) \cdot e\left(d_2, c_3\right)}{e\left(d_3, c_0\right)}$$

  Observe that at the core of the above computation is the original decryption procedure in the scheme of Boneh and Franklin, with the additional components in the ciphertext and the secret-key canceling out each other to *recover the effect of the* PKE(the reader is referred to Section 4 for the detailed proof of correctness). It is important to note that this removal is different from directly decrypting $\mathsf{sk}_{\mathsf{id}}$, and in particular, does not require the knowledge of the secret-key of the PKE.

1.3 Other Related Work

Computational function privacy for predicate encryption has been studied in the private-key setting [18]. The inherent difficulty of achieving function privacy in the public-key setting does not apply to the private-key setting, where the encryptor and decryptor have a shared secret-key. In this setting, an adversary with access to a searching key cannot test the same on ciphertexts of its choice since it does not have access to the secret-key. Function privacy in the private-key setting is thus more natural to achieve. A general solution in this direction was proposed by Goldreich and Ostrovsky [19] in their construction of an oblivious RAM. More efficient constructions have been subsequently proposed for equality testing [20, 21, 12, 22, 23] and, more recently, for inner product testing [13–15].

The first meaningful notion of predicate privacy in the public-key setting was put forth by Boneh, Raghunathan and Segev for equality predicates [10], and subsequently for more general subspace-membership predicates [11]. Prior to their work, achieving predicate privacy in the public-key setting was considered impossible. Their notion of predicate privacy is based on the statistical indistinguishability of secret-keys from random, subject to the condition that the predicates are sampled from sufficiently unpredictable distributions. Prior to this work, no public-key constructions satisfying computational predicate privacy have been proposed to the best of our knowledge.

1.4 Paper Organization

The remainder of this paper is organized as follows. Section 2 presents background material on predicate encryption, and introduces several computational assumptions in bilinear groups. In Section 3, we formally define our framework for the computational function privacy of public-key predicate encryption. In Section 4, we present a family of adaptively data private and computationally function private IBE schemes in the random-oracle model. In Section 5, we present a family of selectively attribute hiding and computationally function private IPE schemes in the standard model. Finally, Section 6 concludes the paper and enumerates several open problems.

1.5 Notations Used

We write $x \xleftarrow{R} \chi$ to represent that an element x is sampled uniformly at random from a set $\mathcal{X}$. The output a of a deterministic algorithm $\mathcal{A}$ is denoted by $x \leftarrow \mathcal{A}$ and the output a' of a randomized algorithm $\mathcal{A}'$ is denoted by $x' \xleftarrow{R} \mathcal{A}'$. We refer to $\lambda \in \mathbb{N}$ as the security parameter, and denote by $\mathsf{exp}(\lambda)$, $\mathsf{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ any generic (unspecified) exponential function, polynomial function and negligible function in λ respectively. Note that a function $f : \mathbb{N} \to \mathbb{N}$ is said to be negligible in λ if for every positive polynomial p, $f(\lambda) < 1/p(\lambda)$ when λ is sufficiently large. Finally, for $a, b \in \mathbb{Z}$ such that $a \leq b$, we denote by $[a, b]$ the set of integers lying between a and b (both inclusive).

The min-entropy of a random variable Y is denoted as $\mathbf{H}_\infty(Y) = -\log(\max_y \Pr[Y = y])$; a random variable Y is said to be a k-source if $\mathbf{H}_\infty(Y) \geq k$. A (T, k)-block-source is a random variable $\mathbf{Y} = (Y_1, \cdots, Y_T)$ where for each $i \in [1, T]$ and $y_1, \cdots, y_{i-1}$, it holds that:

$$\mathbf{H}_\infty(Y_i | Y_1 = y_1, \cdots, Y_{i-1} = y_{i-1}) \geq k$$

2 Preliminaries

2.1 Public-key Predicate Encryption

A public-key predicate encryption scheme for a class of predicates $\mathcal{F}$ over an attribute space Σ and a payload-message space $\mathcal{M}$ is a quadruple $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic polynomial time algorithms. The Setup algorithm takes as input the security parameter λ, and generates the public parameter pp and the master secret-key msk for the system. The key-generation algorithm, KeyGen takes as input the master secret-key msk and a predicate $f \in \mathcal{F}$, and generates a secret-key sk_f corresponding to f. The Enc algorithm takes as input the public parameter pp, an attribute $I \in \Sigma$ and a payload-message $M \in \mathcal{M}$, and outputs the ciphertext $C = \mathsf{Enc}(\mathsf{pp}, I, M)$. The Dec algorithm takes as input the public parameter pp, a ciphertext C and a secret-key sk_f, and outputs either a payload-message $M \in \mathcal{M}$ or the symbol $\perp$.

Functional Correctness. A predicate encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be functionally correct if for any security parameter λ, for any predicate $f \in \mathcal{F}$, for any attribute $I \in \Sigma$ and any payload-message $M \in \mathcal{M}$, the following hold with probability at least $1 - \mathsf{negl}(\lambda)$:

1. If $f(I) = 1$, we have $\mathsf{Dec}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, I, M), \mathsf{KeyGen}(\mathsf{msk}, f)) = M$.
2. If $f(I) = 0$, we have $\mathsf{Dec}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, I, M), \mathsf{KeyGen}(\mathsf{msk}, f)) = \perp$.

where the probability is taken over the internal randomness of the algorithms $\mathsf{Setup}, \mathsf{KeyGen}$, Enc, and Dec.

Data Privacy. We briefly recall the notion of data privacy for a predicate encryption scheme under an *adaptive* chosen-attribute chosen-payload-message attack. Data privacy of a functional encryption scheme guarantees that any probabilistic polynomial-time adversary can gain no information about either the attribute I nor the payload-message M associated with a ciphertext C from the knowledge of the public parameters pp. We denote this notion of security by DP throughout the rest of the paper.

Definition 2.1 (Adaptively Data Private Predicate Encryption). A predicate encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be *adaptively data private* if for any probabilistic polynomial-time adversary $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathsf{DP}}_{\Pi,\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{DP},\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{DP},\Pi,\mathcal{A}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where for each $\lambda \in \mathbb{N}$ and each $b \in \{0,1\}$, the experiment $\mathsf{Expt}^{(b)}_{\mathsf{DP},\Pi,\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pp}, \mathsf{msk}) \overset{R}{\leftarrow} \mathsf{Setup}\left(1^\lambda\right)$.
2. $\left((I_0^*, M_0^*), (I_1^*, M_1^*), \mathsf{state}\right) \overset{R}{\leftarrow} \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{state})$, where $I_0^*, I_1^* \in \Sigma$ and $M_0^*, M_1^* \in \mathcal{M}$, subject to the restriction that for each predicate f_i with which $\mathcal{A}$ queries $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$, we have $f_i(I_0^*) = f_i(I_1^*)$.
3. $C^* \overset{R}{\leftarrow} \mathsf{Enc}(\mathsf{pp}, I_b^*, M_b^*)$.
4. $b' \overset{R}{\leftarrow} \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(C^*, \mathsf{state})$, once again subject to the restriction that for each predicate f_i with which $\mathcal{A}$ queries $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$, we have $f_i(I_0^*) = f_i(I_1^*)$.
5. Output b'.

We also consider a *selective* variant of the above security notion that requires the adversary to commit to the challenge pair of attributes before seeing the public parameters of the scheme. We denote this notion of security by sDP throughout the rest of the paper.

Definition 2.2 (Selectively Data private Predicate Encryption). A predicate encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be *selectively data private* if for any probabilistic polynomial-time adversary $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathsf{sDP}}_{\Pi,\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{sDP},\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{sDP},\Pi,\mathcal{A}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where for each $\lambda \in \mathbb{N}$ and each $b \in \{0,1\}$, the experiment $\mathsf{Expt}^{(b)}_{\mathsf{sDP},\Pi,\mathcal{A}}(\lambda)$ is defined as follows*:*

1. $(I_0^*, I_1^*, \mathsf{state}) \overset{R}{\longleftarrow} \mathcal{A}\left(1^\lambda\right)$, where $I_0^*, I_1^* \in \Sigma$.
2. $(\mathsf{pp}, \mathsf{msk}) \overset{R}{\longleftarrow} \mathsf{Setup}\left(1^\lambda\right)$.
3. $(M_0^*, M_1^*, \mathsf{state}) \overset{R}{\longleftarrow} \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{state})$, where $M_0^*, M_1^* \in \mathcal{M}$, subject to the restriction that for each predicate f_i with which $\mathcal{A}$ queries $\mathsf{KeyGen}(\mathsf{msk},\cdot)$, we have $f_i\left(I_0^*\right) = f_i\left(I_1^*\right)$.
4. $C^* \overset{R}{\longleftarrow} \mathsf{Enc}\left(\mathsf{pp}, I_b^*, M_b^*\right)$.
5. $b' \overset{R}{\longleftarrow} \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}\left(C^*, \mathsf{state}\right)$, once again subject to the restriction that for each predicate f_i with which $\mathcal{A}$ queries $\mathsf{KeyGen}(\mathsf{msk},\cdot)$, we have $f_i\left(I_0^*\right) = f_i\left(I_1^*\right)$.
6. Output b'.

Identity-Based Encryption. An identity-based encryption scheme Π^{IBE} over an identity space $\mathcal{ID}$ and a message space $\mathcal{M}$ is a public-key predicate encryption scheme supporting the set of equality predicates $f_{\mathsf{id}} : \mathcal{ID} \longrightarrow \{0,1\}$ defined as $f_{\mathsf{id}}(\mathsf{id}') = 1$ if and only if $\mathsf{id}' = \mathsf{id}$. The secret-key associated with an identity $\mathsf{id} \in \mathcal{ID}$ is denoted as $\mathsf{sk}_{\mathsf{id}}$. The notions of anonymity and message indistinguishability security popularly associated with IBE are equivalent to the notion of adaptive data privacy as described above.

Inner-Product Encryption. An inner-product encryption scheme Π^{IPE} over an attribute space $\Sigma = \mathbb{F}_q^n$ (q being a λ-bit prime) and a payload message space $\mathcal{M}$ is a public-key predicate encryption scheme supporting the set of vector predicates $f_{\overrightarrow{v}} : \Sigma \longrightarrow \{0,1\}$. The secret-key associated with a vector $\overrightarrow{v} \in \Sigma$ is denoted as $\mathsf{sk}_{\overrightarrow{v}}$. Zero-IPE is a specific sub-class of IPE where for $\overrightarrow{v}, \overrightarrow{x} \in \Sigma$, we have $f_{\overrightarrow{v}}\left(\overrightarrow{x}\right) = 1$ if and only if $\langle \overrightarrow{v}, \overrightarrow{x} \rangle = 0$.

2.2 Computational Assumptions in Bilinear Groups

The decisional bilinear Diffie-Hellman assumption (DBDH). Let $\mathsf{GroupGen}(1^\lambda)$ be a probabilistic polynomial-time algorithm that takes as input a security parameter λ, and outputs the tuple $(\mathbb{G}, \mathbb{G}_T, q, g, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of order q (q being a λ-bit prime), g is a generator for $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map. The group $\mathbb{G}$ is popularly referred to as a *bilinear group* [4]. The decisional bilinear Diffie-Hellman assumption is that the distribution ensembles:

$$\left\{\left(g, g^{a_1}, g^{a_2}, g^{a_3}, e(g,g)^{a_1 \cdot a_2 \cdot a_3}\right)\right\}_{a_1,a_2,a_3 \overset{R}{\longleftarrow} \mathbb{Z}_q^*} \text{ and } \left\{\left(g, g^{a_1}, g^{a_2}, g^{a_3}, Z\right)\right\}_{a_1,a_2,a_3 \overset{R}{\longleftarrow} \mathbb{Z}_q^*, Z \overset{R}{\longleftarrow} \mathbb{G}_T}$$

are computationally indistinguishable, where $(\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathsf{GroupGen}(1^\lambda)$.

The decisional linear assumption (DLIN)[24]. Let $\mathbb{G}$ be a group of prime order q and let g_1, g_2, g_3 be arbitrary generators for $\mathbb{G}$. The decisional linear assumption is that the distribution ensembles:

$$\left\{\left(g_1, g_2, g_3, g_1^{a_1}, g_2^{a_2}, g_3^{a_1+a_2}\right)\right\}_{a_1,a_2 \xleftarrow{R} \mathbb{Z}_q^*} \text{ and } \left\{\left(g_1, g_2, g_3, g_1^{a_1}, g_2^{a_2}, g_3^{a_3}\right)\right\}_{a_1,a_2,a_3 \xleftarrow{R} \mathbb{Z}_q^*}$$

are computationally indistinguishable.

The DLIN assumption was introduced by Boneh, Boyen, and Shacham [24], and was intended to take the place of the more standard decisional Diffie Hellman (DDH) assumption in groups where the DDH assumption does not hold. In particular, for bilinear groups as defined above, the DLIN assumption holds even if the DDH assumption does not, at least in the generic group model.

The generalized decisional *k*-linear assumption (*k*-DLIN) [25]. Let $\mathbb{G}$ be a group of prime order q and let $g_1, \cdots, g_k, g_{k+1}$ be arbitrary generators for $\mathbb{G}$. The generalized decisional k-linear assumption is that the distribution ensembles:

$$\left\{\left(g_1, \cdots, g_k, g_{k+1}, g_1^{a_1}, \cdots, g_k^{a_k}, g_{k+1}^{\sum_{j=1}^{k} a_j}\right)\right\}_{a_1,\cdots,a_k \xleftarrow{R} \mathbb{Z}_q^*} \text{ and}$$
$$\left\{\left(g_1, \cdots, g_k, g_{k+1}, g_1^{a_1}, \cdots, g_k^{a_k}, g_{k+1}^{a_{k+1}}\right)\right\}_{a_1,\cdots,a_k,a_{k+1} \xleftarrow{R} \mathbb{Z}_q^*}$$

are computationally indistinguishable.

Quite evidently, this assumption is a generalization of the DLIN assumption stated above. Note that the k-DLIN assumption implies the $(k+1)$-DLIN assumption for all $k \geq 1$, but the reverse is not necessarily true, implying that the k-DLIN assumption family is a family of progressively weaker assumptions [25].

3 Computational Function Privacy of Public-Key Predicate Encryption

We present our definitions for the computational function privacy of predicate encryption in the public-key setting. We consider adversaries that have access to the public parameters of the scheme, as well as a secret-key generation oracle. The adversary can also adaptively interact with a *real-or-random* function-privacy oracle $\mathsf{RoR}^{\mathsf{FP}}$. This oracle takes as input any adversarially-chosen distribution over the class of predicates $\mathcal{F}$, and outputs a secret-key either for a predicate sampled from the given distribution, or for an independently and uniformly sampled predicate. At the end of the interaction, the adversary should be able to distinguish between these *real* and *random* modes of operation of $\mathsf{RoR}^{\mathsf{FP}}$ with only negligible probability.

Formal Definitions. We now formally present the computational function privacy definitions for public-key predicate encryption.

Definition 3.1 (Real-or-Random Function Privacy Oracle for Predicate Encryption). The real-or-random function privacy oracle for predicate encryption $\mathsf{RoR}^{\mathsf{FP}}$ takes as input triplets of the form $(\mathsf{mode}, msk, \mathbf{F})$, where $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$, msk is the master secret-key, and $\mathbf{F}$ is a circuit representing a distribution over the class of predicates $\mathcal{F}$. If $\mathsf{mode} = \mathsf{real}$, the oracle samples $f \xleftarrow{R} \mathbf{F}$, while if $\mathsf{mode} = \mathsf{rand}$, it samples $f \xleftarrow{R} \mathcal{F}$. It then computes $\mathsf{sk}_f \xleftarrow{R} \mathsf{KeyGen}(msk, f)$ and responds with sk_f.

Definition 3.2 (Computational Function Privacy for Predicate Encryption). A predicate encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be *computationally function private* if for any probabilistic polynomial-time adversary $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathsf{FP}}_{\Pi,\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{FP},\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi,\mathcal{A}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where for each $\lambda \in \mathbb{N}$ and each $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$, the experiment $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi,\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pp}, \mathsf{msk}) \xleftarrow{R} \mathsf{Setup}\left(1^\lambda\right)$.
2. $b \xleftarrow{R} \mathcal{A}^{\mathsf{RoR}^{\mathsf{FP}}(\mathsf{mode},\mathsf{msk},\cdot),\mathsf{KeyGen}(\mathsf{msk},\cdot)}\left(1^\lambda, \mathsf{pp}\right)$, subject to the restriction that each $\mathbf{F}_i$ with which $\mathcal{A}$ queries $\mathsf{RoR}^{\mathsf{FP}}(\mathsf{mode}, \mathsf{msk}, \cdot)$ represents a distribution with min-entropy $k = \omega(\log \lambda)$.
3. Output b.

Note that our definitions are generic, and may be suitably adopted for IBE, IPE and other classes of predicate encryption.

Min-Entropy Requirements. In our definitions for computational function privacy, the adversary is allowed to adaptively issue a polynomial number of queries to the $\mathsf{RoR}^{\mathsf{FP}}$ oracle, as long as the queries correspond to distributions with min-entropy $k = \omega(\log \lambda)$. As discussed in Section 1.2, such a restriction is *necessary* for any definition of function privacy to be meaningful in the public-key setting. In the context of IBE, for example, the adversary is allowed to query the real-or-random oracle with $\mathbf{ID}^* \in \mathcal{ID}$ only if $\mathbf{ID}^*$ represents a k-source such that $k = \omega(\log \lambda)$. In the context of IPE, on the other hand, and adversary can query the real-or-random oracle with $\mathbf{V}^* = (V_1^*, \cdots, V_n^*) \in \mathbb{Z}_N^n$ only of $\mathbf{V}^*$ is an (n, k)-block source such that $k = \omega(\log \lambda)$. Additionally, each component-wise distribution V_i^* for $i \in [1, n]$ should be completely uncorrelated with each of the other distributions in $\mathbf{V}^*$. This restriction is necessary to ensure that the adversary cannot carefully craft vectorial distributions with arbitrary inter-component correlations that trivially compromise function privacy.

Multi-Shot v/s Single-Shot Adversaries. Definition 3.1 considers *multi-shot* adversaries that are allowed to query the $\mathsf{RoR}^{\mathsf{FP}}$ oracle polynomially many times. However, it is polynomially equivalent to consider *single-shot* adversaries that can query the $\mathsf{RoR}^{\mathsf{FP}}$ at most once. This is easily established by a hybrid argument, where the hybrids are constructed such that only one query is forwarded to the $\mathsf{RoR}^{\mathsf{FP}}$ oracle, while the rest are answered by the key generation oracle.

3.1 Computational Function Privacy of Existing Predicate Encryption Schemes

Identity-Based Encryption. To the best of our knowledge, there exist no IBE schemes in literature that can be proven to be computationally function private under well-known cryptographic assumptions. The constructions in [4] and [26] have deterministic trapdoors, and are hence trivially not function private under Definition 3.1. For such schemes, an adversary could easily manufacture a circuit that uniformly samples id such that some function of the public parameters pp and the secret-key $\mathsf{sk}_{\mathsf{id}}$ is already known to the adversary, thus leading to a straightforward attack breaking function privacy [10]. While such straightforward attacks cannot be demonstrated on the IBE constructions proposed in [5, 27–29], we are not aware if their function privacy can be based on standard computationally intractable problems. Finally, the IBE constructions presented in [10] are secure under a different notion of function privacy called statistical function privacy, that is based on the statistical closeness of adversarially-chosen and random distributions. Once again, to the best of our knowledge, these constructions are not computationally function private to the best of our knowledge. In Section 4, we present a family of IBE constructions that are computationally function private under well-known cryptographic assumptions.

Inner-Product Encryption. While computational functional privacy with respect to IPE in the private key setting is well-studied [13–15], there exist, to the best of our knowledge, no equivalent public-key counterparts in literature that can be proven to be function private under well-known cryptographic assumptions. The authors of [11] present a generic technique to achieve statistical function privacy in the context of inner-product encryption in the public-key setting; however, they leave the construction of computationally function private IPE schemes in the public-key setting as an open problem. It also seems that the function privacy of existing IPE constructions, such as in [2, 3], cannot directly be based on standard computational assumptions without suitable modifications. In Section 5, we present a family of IPE constructions that are computationally function private under well-known cryptographic assumptions.

3.2 Enhanced Computational Function Privacy for Predicate Encryption

Boneh, Raghunathan and Segev put forth a stronger notion of function privacy in [11] with respect to IBE, referred to as *enhanced* function privacy. Informally, this notion requires that the function privacy guarantees of an IBE scheme hold even if the adversary can obtain, in addition to the secret-key $\mathsf{sk}_{\mathsf{id}}$ corresponding to an identity id, an encryption of an arbitrary message M under id. We formally extend their notion of enhanced computational function privacy to address a more generalized class of predicates. We assume that the adversary, in addition to interacting with the key generation and real-or-random function privacy oracles, also interacts with a function-privacy encryption oracle $\mathsf{Enc}^{\mathsf{FP}}$. This oracle shares a state with the real-or-random function privacy oracle. For a multi-shot adversary, let f_j^* be the predicate sampled by the real-or-random function privacy oracle when responding to the j^{th} query made by the adversary to the real-or-random function privacy oracle. The $\mathsf{Enc}^{\mathsf{FP}}$ oracle takes as input a payload message M and j, uniformly samples an attribute I_j^* such that $f_j^*\left(I_j^*\right) = 1$, and responds with $C_j^* = \mathsf{Enc}\left(\mathsf{pp}, \left(I_j^*, M\right)\right)$. The equivalent definition for a single-shot adversary follows similarly.

Definition 3.3 (Enhanced Computational Function Privacy for Predicate Encryption). A predicate encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be *computationally enhanced function private* if for any probabilistic polynomial-time adversary $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathsf{EFP}}_{\Pi,\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{EFP},\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP},\Pi,\mathcal{A}}(\lambda) = 1\right] \right| \le \mathsf{negl}(\lambda)$$

where for each $\lambda \in \mathbb{N}$ and each $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$, the experiment $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi,\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pp}, \mathsf{msk}) \xleftarrow{R} \mathsf{Setup}\left(1^\lambda\right)$.
2. $b \xleftarrow{R} \mathcal{A}^{\mathsf{RoR}^{\mathsf{FP}}(\mathsf{mode},\mathsf{msk},\cdot),\mathsf{Enc}^{\mathsf{FP}}(\cdot),\mathsf{KeyGen}(\mathsf{msk},\cdot)}\left(1^\lambda, \mathsf{pp}\right)$, subject to the restriction that each $\mathbf{F}_i$ with which $\mathcal{A}$ queries $\mathsf{RoR}^{\mathsf{FP}}(\mathsf{mode}, \mathsf{msk}, \cdot)$ represents a distribution with min-entropy $k = \omega(\log \lambda)$.
3. Output b.

4 Computationally Function private Identity-Based Encryption

In this section, we apply our *encrypt-augment-recover* approach to the anonymous IBE scheme of Boneh and Franklin [4] to achieve a family of computationally function private IBE schemes $\{\Pi^{\mathrm{IBE}}_k\}_{k\geq 1}$. At the core of Π^{IBE}_k is the following generalized version of the PKE scheme introduced in Section 1.2, which is CPA-secure under the $(k+1)$-DLIN assumption:

- **KeyGen:** The key-generation algorithm samples $x_1, \cdots, x_{k+2} \xleftarrow{R} \mathbb{Z}^*_q$, where q is a λ-bit prime, and $g_1, \cdots, g_{k+2} \xleftarrow{R} \mathbb{G}$, where $\mathbb{G}$ is a cyclic group of prime order q. It outputs the secret-key SK and the public-key PK as:
$$SK = (x_1, \cdots, x_{k+2}) \ , \ PK = \left(g_1, \cdots, g_{k+2}, \{\left(g_j^{x_j} \cdot g_{k+2}^{x_{k+2}}\right)\}_{j\in[1,k+1]}\right)$$
- **Enc:** The ciphertext C corresponding to a message $M \in \mathbb{G}$ is a tuple of the form:
$$C = \left(g_1^{y_1}, \cdots, g_{k+1}^{y_{k+1}}, g_{k+2}^{\sum_{j=1}^{k+1} y_j}, \left(\prod_{j=1}^{k+1} \left(g_j^{x_j} \cdot g_{k+2}^{x_{k+2}}\right)^{y_j}\right) \cdot M\right)$$
where $y_1, \cdots, y_{k+1} \xleftarrow{R} \mathbb{Z}^*_q$.
- **Dec:** The decryption algorithm, on input the ciphertext $C = (c_1, \cdots, c_{k+3})$ and the secret-key $(x_1, \cdots, x_{k+2})$, recovers the message M as:
$$M = c_{k+3} \bigg/ \left(\prod_{j=1}^{k+1} c_j^{x_j}\right)$$

The IBE scheme Π^{IBE}_k, which is built using the above scheme, is adaptively data private under the DBDH assumption and function private under the $(k+1)$-DLIN assumption. We present the construction for the same next.

The Construction. Let $\mathsf{GroupGen}(1^\lambda)$ be a probabilistic polynomial-time algorithm that takes as input a security parameter λ, and outputs the tuple $(\mathbb{G}, \mathbb{G}_T, q, g, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order $q = \mathcal{O}(2^\lambda)$, g is a generator for $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map. The IBE scheme $\Pi_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined over the identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda\in\mathbb{N}}$ and the message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$. We assume that $\mathcal{M}$ is a small subset of $\mathbb{G}_T$, namely $|\mathcal{M}| < |\mathbb{G}_T|^{1/2}$. While the reason for this restriction will become clear eventually, we note that it is not very serious since the space of valid messages in reality is expected to be significantly smaller than $|\mathbb{G}_T|^{1/2}$. Finally, let $H : \mathcal{ID} \longrightarrow \mathbb{G}$ be a publicly available collision-resistant hash function.

- **Setup:** The setup algorithm samples $(\mathbb{G}, \mathbb{G}_T, q, g, e) \xleftarrow{R} \mathsf{GroupGen}(1^\lambda)$ on input the security parameter 1^λ. It also samples $s, x_1, \cdots, x_{k+2} \xleftarrow{R} \mathbb{Z}_q^*$ and $g_1, \cdots, g_{k+2} \xleftarrow{R} \mathbb{G}$. It outputs the public parameter pp and the master secret-key msk as:
$$\begin{aligned}\mathsf{pp} &= \left(g, g^s, g^{x_1}, \cdots, g^{x_{k+2}}\right)\\ \mathsf{msk} &= \left(s, g_1, \cdots, g_{k+2}, \left(g_1^{x_1} \cdot g_{k+2}^{x_{k+2}}\right), \cdots, \left(g_{k+1}^{x_{k+1}} \cdot g_{k+2}^{x_{k+2}}\right)\right)\end{aligned}$$
- **KeyGen:** On input the master secret-key msk and an identity $\mathsf{id} \in \mathcal{ID}$, the key generation algorithm samples $y_1, \cdots, y_{k+1} \xleftarrow{R} \mathbb{Z}_q^*$ and outputs the secret-key $\mathsf{sk}_{\mathsf{id}} = (d_0, \cdots, d_{k+2})$ where:
$$d_{j-1} = g_j^{y_j} \text{ for } j \in [1, k+1]\ ,\ d_{k+1} = g_{k+2}^{\sum_{j=1}^{k+1} y_j}\ ,\ d_{k+2} = \left(\prod_{j=1}^{k+1} \left(g_j^{x_j} \cdot g_{k+2}^{x_{k+2}}\right)^{y_j}\right) \cdot \left(H\left(\mathsf{id}\right)\right)^s$$
Observe that $\mathsf{sk}_{\mathsf{id}} = \mathsf{PKE.Enc}\left(PK, \left(H\left(\mathsf{id}\right)\right)^s\right)$.

- **Enc:** On input the public parameter pp, an identity $\mathsf{id} \in \mathcal{ID}$ and a message $M \in \mathcal{M}$, the encryption algorithm samples $r \xleftarrow{R} \mathbb{Z}_q^*$ and outputs the ciphertext $C = (c_0, \cdots, c_{k+3})$ where:
$$c_0 = g^r\ ,\ c_j = \left(g^{x_j}\right)^r \text{ for } j \in [1, k+2]\ ,\ c_{k+3} = M \cdot e\left(H\left(\mathsf{id}\right), g^s\right)^r$$
- **Dec:** On input a ciphertext $C = (c_0, \cdots, c_{k+3})$ and a secret-key $\mathsf{sk}_{\mathsf{id}} = (d_0, \cdots, d_{k+2})$, the decryption algorithm computes:
$$M' = c_{k+3} \cdot \frac{\prod_{j=1}^{k+2} e\left(d_{j-1}, c_j\right)}{e\left(d_{k+2}, c_0\right)}$$
If $M' \in \mathcal{M}$, the decryption algorithm outputs M', else it outputs $\perp$.

Correctness. First, consider a message $M \in \mathcal{M}$, a ciphertext $C = (c_0, \cdots, c_{k+3})$ corresponding to M under an identity $\mathsf{id} \in \mathcal{ID}$ and a secret-key $\mathsf{sk}_{\mathsf{id}} = (d_0, \cdots, d_{k+2})$ corresponding to id. Then, we have:
$$\begin{aligned}M' &= M \cdot e\left(H\left(\mathsf{id}\right), g^s\right)^r \cdot \frac{\prod_{j=1}^{k+1} e\left(g_j^{y_j}, \left(g^{x_j}\right)^r\right) \cdot e\left(g^{\sum_{j=1}^{k+1} y_j}, \left(g^{x_{k+2}}\right)^r\right)}{e\left(\left(\prod_{j=1}^{k+1} \left(g_j^{x_j} \cdot g_{k+2}^{x_{k+2}}\right)^{y_j}\right) \cdot \left(H\left(\mathsf{id}\right)\right)^s, g^r\right)}\\ &= M \cdot e\left(H\left(\mathsf{id}\right), g^s\right)^r \cdot \frac{\prod_{j=1}^{k+1} e\left(g_j^{y_j}, \left(g^{x_j}\right)^r\right) \cdot e\left(g^{\sum_{j=1}^{k+1} y_j}, \left(g^{x_{k+2}}\right)^r\right)}{\prod_{j=1}^{k+1} e\left(g_j^{x_j}, g^r\right)^{y_j} \cdot e\left(g_{k+2}^{x_{k+2}}, g^r\right)^{\sum_{j=1}^{k+1} y_j} \cdot e\left(\left(H\left(\mathsf{id}\right)\right)^s, g^r\right)}\\ &= M\end{aligned}$$
Therefore as long as the ciphertext and the secret-key correspond to the same identity, the message is recovered correctly. Again, when the ciphertext and the secret-key correspond to two different identities, say id and id' respectively, the decryption algorithm computes:
$$M' = M \cdot e\left(H\left(\mathsf{id}\right) \cdot \left(H\left(\mathsf{id}'\right)\right)^{-1}, g^s\right)^r$$
Now, the restriction that $\mathcal{M}$ is a small subset of $\mathbb{G}_T$ of size less than $\left|\mathbb{G}_T\right|^{1/2}$ ensures that the probability of M' still lying in $\mathcal{M}$, for $s, r \xleftarrow{R} \mathbb{Z}_q^*$ and a collision-resistant hash function H, is negligible in the security parameter λ. This completes the proof of correctness for our generalized IBE scheme Π_k^{IBE}.

4.1 Security of Our IBE Scheme

Data Privacy. We state the following theorem for the data privacy of Π_k^{IBE}:

Theorem 4.1 *Our IBE scheme Π_k^{IBE} is adaptively data private under the* DBDH *assumption.*

Proof Overview. Due to space limitation, we only provide a brief overview of the proof idea here. The detailed proof is presented in Appendix A. The proof considers a probabilistic polynomial-time that has access to the public parameters of our scheme, and can determine from a *challenge ciphertext* whether it is associated with either of two identity-message pairs (id_0^*, M_0^*) or (id_1^*, M_1^*) with non-negligible advantage. The adversary can request for secret-keys corresponding to identities of its choice, subject to the restriction that the requested identities differ from both id_0^* and id_1^*.

In order to provide adaptive data privacy under the DBDH assumption, the proof considers a simulator that receives the DBDH instance $(g, g^{a_1}, g^{a_2}, g^{a_3}, Z)$ as input and chooses $x_1, \cdots, x_{k+2} \xleftarrow{R} \mathbb{Z}_q^*$ as input. The public parameters are set up as $(g, g^{a_1}, g^{x_1}, \cdots, g^{x_{k+2}})$, where $x_1, \cdots, x_{k+2} \xleftarrow{R} \mathbb{Z}_q^*$. The simulator simulates the hash function H as a random oracle. On receiving a hash query for some identity id_i, The simulator responds with either either g^{α_i} or $(g^{a_2})^{\alpha_i}$ for $\alpha_i \xleftarrow{R} \mathbb{Z}_q^*$, and remembers the response for future queries.

The simulator also simulates the secret-key generation oracle. On receiving a secret-key query for id_i, the simulator looks up $H(\mathsf{id}_i)$, and either aborts if $H(\mathsf{id}_i) = (g^{a_2})^{\alpha_i}$, or chooses $y_1, \cdots, y_{k+1} \xleftarrow{R} \mathbb{Z}_q^*$ and responds with:

$$\mathsf{sk}_{\mathsf{id}_i} = \left(g_1^{y_1}, \cdots, g_2^{y_{k+1}}, g_{k+2}^{\sum_{j=1}^{k+1} y_j}, \left(\prod_{j=1}^{k+2} \left(g_j^{x_j} \cdot g_{k+2}^{x_{k+2}} \right)^{y_j} \right) \cdot (g^{a_1})^{\alpha_i} \right)$$

It is easy to see when the simulator does not abort, the distribution of $\mathsf{sk}_{\mathsf{id}_i}$ is exactly as in the real world.

Finally, in the challenge phase, the simulator chooses $b \xleftarrow{R} \{0,1\}$ and looks up $H(\mathsf{id}_b^*)$. It aborts if $H(\mathsf{id}_b^*)$ is of the form g^{α^*} for some $\alpha^* \xleftarrow{R} \mathbb{Z}_q^*$. Otherwise, it must be the case that $H(\mathsf{id}_b^*)$ is of the form $g^{a_2 \cdot \alpha^*}$. In this case, it embeds the DBDH challenge in the challenge ciphertext C^* as follows:

$$C^* = \left((g^{a_3})^{(\alpha_b^*)^{-1}}, (g^{a_3})^{x_1 \cdot (\alpha_b^*)^{-1}}, \cdots, (g^{a_3})^{x_{k+2} \cdot (\alpha_b^*)^{-1}}, M_b^* \cdot Z \right)$$

This ciphertext is either well-formed or uniform and independent in the adversary's view depending on the DBDH instance received by the simulator as input. This allows us to relate the advantage of the simulator in solving the DBDH instance to the advantage of the adversary in breaking the adaptive data privacy of our scheme. Finally, the abort condition in the secret-key query phase and the challenge phase is mutually opposite, so that the overall probability of abortion can be suitably minimized. This is essentially a a modification of the original proof idea in [4] to fit our function private scheme.

Computational Function Privacy. We state the following theorem for the computational function privacy of Π_k^{IBE}:

Theorem 4.2 *Our IBE scheme Π_k^{IBE} is computationally function private under the* $(k+1)$-DLIN *assumption for identities sampled uniformly from k-sources with $k = \omega(\log \lambda)$.*

Proof. We begin by stating the following claim:

Claim 4.1 *For any probabilistic polynomial-time adversary $\mathcal{A}$ and for* $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$*, the following holds:*

$$\left| \Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi^{\mathrm{IBE}}_k,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi^{\mathrm{IBE}}_k,\mathcal{A}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

To prove this claim, we assume the contrary. Let $\mathcal{A}$ be a probabilistic polynomial-time adversary such that for $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$, we have:

$$\left| \Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi^{\mathrm{IBE}}_k,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi^{\mathrm{IBE}}_k,\mathcal{A}}(\lambda) = 1\right] \right| = \epsilon > \mathsf{negl}(\lambda)$$

We assume that the adversary $\mathcal{A}$ issues a single query to the real-or-random oracle. As discussed in Section 3, such a single-shot adversary is polynomially equivalent to its multi-shot variant considered in Definition 3.1. We construct an algorithm $\mathcal{B}$ that solves an instance of the $(k+1)$-DLIN problem with non-negligible advantage $\epsilon' = \epsilon$. $\mathcal{B}$ is given $\left(g_1, \cdots, g_{k+2}, g_1^{a_1}, \cdots, g_{k+2}^{a_{k+2}}\right)$ and interacts with $\mathcal{A}$ as follows:

- ***Setup:*** $\mathcal{B}$ samples $s, x_1, \cdots, x_{k+2} \xleftarrow{R} \mathbb{Z}_q^*$ and $g \xleftarrow{R} \mathbb{G}$. It sets $\mathsf{pp} = \left(g, g^s, g^{x_1}, \cdots, g^{x_{k+2}}\right)$ and provides pp to $\mathcal{A}$. Observe that the distribution of pp is exactly as in the real world.

- *$H -$ **Queries:*** $\mathcal{B}$ maintains a list of tuples of the form $\langle \mathsf{id}_j, H(\mathsf{id}_j)\rangle$ such that $H(\mathsf{id}_j) \in \mathbb{G}$. When $\mathcal{A}$ issues a hash query for $\mathsf{id}_i \in \mathcal{ID}$, $\mathcal{B}$ responds as follows:

  1. $\mathcal{B}$ searches the list for a matching tuple of the form $\langle \mathsf{id}_i, H(\mathsf{id}_i)\rangle$. If such a tuple is found, it returns $H(\mathsf{id}_i)$ to $\mathcal{A}$.
  2. Otherwise, $\mathcal{B}$ samples $H(\mathsf{id}_i) \xleftarrow{R} \mathbb{G}$ and adds the tuple $\langle \mathsf{id}_i, H(\mathsf{id}_i)\rangle$ to its existing list. It returns $H(\mathsf{id}_i)$ to $\mathcal{A}$.

- ***Secret-Key Queries:*** When $\mathcal{A}$ issues a secret-key query for $\mathsf{id}_i \in \mathcal{ID}$, $\mathcal{B}$ responds as follows:
  1. $\mathcal{B}$ runs the aforementioned procedure to look up $H(\mathsf{id}_i)$.
  2. $\mathcal{B}$ samples $y_1, \cdots, y_{k+1} \xleftarrow{R} \mathbb{Z}_q^*$ and responds with:

  $$\mathsf{sk}_{\mathsf{id}_i} = \left(g_1^{y_1}, \cdots, g_{k+1}^{y_{k+1}}, g_{k+2}^{\sum_{j=1}^{k+1} y_j}, \left(\prod_{j=1}^{k+1} \left(g_j^{x_j} \cdot g_{k+2}^{x_{k+2}}\right)^{y_j}\right) \cdot \left(H(\mathsf{id}_i)\right)^s \right)$$

  where $g_1, \cdots, g_{k+2}$ are part of its input instance. Once again, observe that the distribution of $\mathsf{sk}_{\mathsf{id}_i}$ is exactly as in the real world.

- ***Real-or-Random Query:*** Suppose $\mathcal{A}$ queries the real-or-random oracle with $\mathbf{ID}^*$ - a circuit representing a k-source over the identity space $\mathcal{ID}$ such that $k = \omega(\log \lambda)$. $\mathcal{B}$ samples $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$ and does the following:
  1. If $\mathsf{mode} = \mathsf{real}$, $\mathcal{B}$ samples $\mathsf{id}^* \xleftarrow{R} \mathbf{ID}^*$, while if $\mathsf{mode} = \mathsf{rand}$, it samples $\mathsf{id}^* \xleftarrow{R} \mathcal{ID}$. It then runs the aforementioned procedure to look up $H(\mathsf{id}^*)$.

2. $\mathcal{B}$ responds with the secret-key $\mathsf{sk}_{\mathsf{id}^*}$ as:

$$\mathsf{sk}_{\mathsf{id}^*} = \left(g_1^{a_1}, \cdots, g_{k+2}^{a_{k+2}}, \left(\prod_{j=1}^{k+2} \left(g_j^{a_j}\right)^{x_j} \right) \cdot \left(H\left(\mathsf{id}^*\right)\right)^s \right)$$

where $g_1^{a_1}, \cdots, g_{k+2}^{a_{k+2}}$ are part of its input instance.

- ***Guess:*** At the end of the game, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs the same bit b'.

It is easy to see that when $a_{k+2} = \sum_{j=1}^{k+1} a_j$, the secret-key $\mathsf{sk}_{\mathsf{id}^*}$ is well-formed and identically distributed to the response of the real-or-random oracle in the experiment $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$. On the other hand, when a_{k+2} is uniformly random in $\mathbb{Z}_q^*$, the secret-key $\mathsf{sk}_{\mathsf{id}^*}$ is uniformly random, and hence identically distributed to the response of the real-or-random oracle in the experiment $\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$. Now, the advantage ϵ' of $\mathcal{B}$ in solving the $(k+1)$-DLIN instance (where the probability is taken over all possible choices of $a_1, \cdots, a_{k+2} \xleftarrow{R} \mathbb{Z}_q^*$ and all possible choices of $g_1, \cdots, g_{k+2} \xleftarrow{R} \mathbb{G}$) may be quantified as:

$$\begin{aligned}\epsilon' &= \left| \Pr\left[\mathcal{B}\left(g_1, \cdots, g_{k+2}, g_1^{a_1}, \cdots, g_{k+2}^{\sum_{j=1}^{k+1} a_j} \right) = 1 \right] - \Pr\left[\mathcal{B}\left(g_1, \cdots, g_{k+2}, g_1^{a_1}, \cdots, g_{k+2}^{a_{k+2}} \right) = 1 \right] \right| \\ &= \left| \Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1 \right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1 \right] \right| \\ &= \epsilon\end{aligned}$$

This completes the proof of Claim 4.1. The proof of function privacy for Π_k^{IBE} now follows from the following observation:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{FP}}_{\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) &= \left| \Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{FP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1 \right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1 \right] \right| \\ &\leq 2 \left| \Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1 \right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1 \right] \right| \\ &= 2\epsilon \leq \mathsf{negl}(\lambda)\end{aligned}$$

Enhanced Computational Function Privacy. We state the following theorem for the enhanced computational function privacy of Π_k^{IBE}:

Theorem 4.3 *Our IBE scheme Π_k^{IBE} is computationally enhanced function private under the* $(k+1)$-DLIN *assumption for identities sampled uniformly from k-sources with $k = \omega(\log \lambda)$.*

Proof Overview. The proof is very similar to that of Theorem 4.2; hence we restrict to providing a proof overview here. The detailed proof is presented in Appendix B. An important component of the proof is to show that the challenger $\mathcal{B}$ can additionally simulate the function privacy encryption oracle, and that the real and random modes of operation of the function privacy oracle are indistinguishable even in the presence of the encryption oracle. The first part is easy to demonstrate. Let $\mathsf{sk}_{\mathsf{id}^*} = \left(d_0^*, \cdots, d_{k+2}^*\right)$ be the response of the real-or-random function privacy oracle to the adversary $\mathcal{A}$. On receiving a function-privacy encryption oracle query for a payload message M, $\mathcal{B}$ responds as follows: it samples $r \xleftarrow{R} \mathbb{Z}_q^*$ and outputs the ciphertext $C^* = \left(c_0^*, \cdots, c_{k+3}^*\right)$ where:

$$c_0^* = g^r \ , \ c_j^* = \left(g^{x_j}\right)^r \ \text{ for } j \in [1, k+2]$$

and

$$c_{k+3}^* = M \cdot \frac{e\left(d_{k+2}^*, c_0^*\right)}{\prod_{j=1}^{k+2} e\left(d_{j-1}^*, c_j^*\right)}$$

For the second part, observe that when the input $(k+1)$-DLIN challenge for $\mathcal{B}$ is valid, the ciphertext C^* is well-formed and identically distributed to the response of the function privacy encryption oracle in the experiment $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$. On the other hand, when the input $(k+1)$-DLIN challenge for $\mathcal{B}$ is invalid, the ciphertext C^* is uniformly random, and hence identically distributed to the response of the function privacy encryption oracle in the experiment $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$. In either case, decrypting C^* using $\mathsf{sk}_{\mathsf{id}^*}$ correctly retrieves M. Hence, for the real and random modes of operation of the function privacy oracle, the response of the encryption oracle is also indistinguishable under the $(k+1)$-DLIN assumption. This completes the proof overview for the enhanced computational function privacy of Π_k^{IBE}.

5 Computationally Function private Inner-Product Encryption

In this section, we present a family of selectively data private zero-IPE schemes $\{\Pi_k^{\mathrm{IPE}}\}_{k\geq 1}$ that are also computationally function private under the generalized family of k-DLIN assumptions in the standard model. Our schemes are defined over the set of attributes $\Sigma = \mathbb{Z}_N^n$ (N being a product of three primes q_1, q_2 and q_3), and the class of vectorial predicates $\mathcal{F} = \{f_{\overrightarrow{v}} \mid \overrightarrow{v} \in \mathbb{Z}_N^n\}$, such that for $I = (I_1, \cdots, I_n) \in \mathbb{Z}_N^n$, we have $f_{\overrightarrow{v}}(I) = 1$ if and only if $\langle \overrightarrow{v}, I\rangle = 0 \bmod N$. Once again, our constructions are obtained by applying our *encrypt-augment-recover* approach to the zero-IPE scheme of Katz, Sahai and Waters [3].

Construction Overview. Let $\mathbb{G}$ be a bilinear group of order $N = q_1q_2q_3$ (each of q_1, q_2 and q_3 being λ-bit primes), and let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ denote the subgroups of $\mathbb{G}$ of order q_1, q_2 and q_3, respectively. Also, let $\hat{e} : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ be an efficiently computable non-degenerate bilinear map, where $\mathbb{G}_T$ is also a group of order N. Note that if g is the generator for $\mathbb{G}$, then the element $g_1 = g^{q_2 \cdot q_3}$ is a generator for $\mathbb{G}_1$, the element $g_2 = g^{q_1 \cdot q_3}$ is a generator for $\mathbb{G}_2$, and the element $g_3 = g^{q_1 \cdot q_2}$ is a generator for $\mathbb{G}_3$. Furthermore, for any elements $h_1 \in \mathbb{G}_1$, $h_2 \in \mathbb{G}_2$ and $h_3 \in \mathbb{G}_3$, we have $\hat{e}(h_1, h_2) = \hat{e}(h_2, h_3) = \hat{e}(h_1, h_3) = 1$. Also, let $\mathsf{GroupGen}'(1^\lambda)$ be a probabilistic polynomial-time algorithm that takes as input a security parameter λ, and outputs the tuple $(\mathbb{G}, \mathbb{G}_T, q_1, q_2, q_3, g_1, g_2, g_3, \hat{e})$. Finally, the payload message space $\mathcal{M}$ is assumed to be a small subset of $\mathbb{G}_T$, namely $|\mathcal{M}| < |\mathbb{G}_T|^{1/2}$. Our function private zero-IPE scheme uses the three subgroups for three distinct roles:

- The subgroup $\mathbb{G}_2$ is used to encode the vectors $\overrightarrow{v}$ and I in the secret-key and the ciphertexts, respectively, and to compute the inner product $\langle \overrightarrow{v}, I\rangle$ in the exponent of a bilinear map computation.

- The subgroup $\mathbb{G}_1$ serves a dual purpose in our scheme. On the one hand, it has the effect of *masking* the inner product computation in $\mathbb{G}_2$, and preventing the adversary from improperly manipulating the computation in any way to reveal information about the underlying attributes. In particular, it is pivotal in ensuring the non-malleability of the secret-keys and ciphertexts generated by the scheme. On the other hand, it is in the $\mathbb{G}_1$ subgroup that we incorporate our *encrypt-augment-recover* methodology to achieve computational function privacy.

- The subgroup $\mathbb{G}_3$ serves as an additional layer of masking for the other subgroups. In particular, random elements sampled from $\mathbb{G}_3$ are multiplied with various components in both the secret-keys as well as the ciphertexts to hide possible information leakages from the subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Encrypt-Augment-Recover. We apply our *encrypt-augment-recover* approach to the zero-IPE scheme of Katz, Waters and Sahai to achieve computational function privacy. At the core of our approach is a public-key encryption algorithm $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ that is CPA-secure under the $(k+1)$-DLIN assumption. The PKE essentially operates in the subgroup $\mathbb{G}_1$ of prime order q_1, and its outputs are suitably masked before being incorporated in our scheme. We modify the algorithms of the original scheme as follows:

- The modified setup algorithm runs $(SK, PK) \xleftarrow{R} \mathsf{PKE.KeyGen}$. It incorporates PK in the master secret-key of the original scheme, and modifies the public parameter to include a one way function of SK.

- The original zero-IPE scheme of Katz, Sahai and Waters comprises of secret-keys of the form $\left(d_0, \{d_{1,i}, d_{2,i}\}_{i\in[1,n]}\right)$. The modified key-generation algorithm in our scheme generates a secret-key of the form $\mathsf{sk}_{\vec{v}} = \left(d_0, \{d^j_{1,i}, d^j_{2,i}\}_{i\in[1,n],j\in[0,k+2]}\right)$ such that $\left(\{d^j_{1,i}\}_{j\in[0,k+2]}\right) = \mathsf{PKE.Enc}\left(PK, d_{1,i}\right)$ and $\left(\{d^j_{2,i}\}_{j\in[0,k+2]}\right) = \mathsf{PKE.Enc}\left(PK, d_{2,i}\right)$ for $i \in [1,n]$ (along with suitable masking as necessary). Observe that this naturally ensures that each component of the modified secret-key is independent and identically distributed. In the proof of function privacy, we argue the indistinguishability of a well-formed secret-key component from a uniformly random one by relating it to the hardness of solving a $(k+1)$-DLIN instance in $\mathbb{G}_1$.

- The modified encryption algorithm generates an *augmented* ciphertext that retains the ciphertext of the original scheme unaltered as one of its components. The additional ciphertext components are used by the modified decryption algorithm subsequently to remove the effect of PKE and recover the payload message M. The additional components are also in the group $\mathbb{G}_1$, and are suitably masked using uniformly random elements from $\mathbb{G}_3$. The masking ensures that the data privacy guarantees of the original scheme are not weakened.

Construction Details. We now present the construction for Π_k^{IPE} in details.

- **Setup:** The setup algorithm samples $(\mathbb{G}, \mathbb{G}_T, q_1, q_2, q_3, g_1, g_2, g_3, \hat{e}) \xleftarrow{R} \mathsf{GroupGen}'(1^\lambda)$. It also samples $\{x_{1,j}, x_{2,j} \xleftarrow{R} \mathbb{Z}^*_{q_1}\}_{j\in[1,k+2]}$, $\{g_{1,j}, g_{2,j} \xleftarrow{R} \mathbb{G}_1\}_{j\in[1,k+2]}$, $\{h_{1,i}, h_{2,i} \xleftarrow{R} \mathbb{G}_1\}_{i\in[1,n]}$ and $\{R^j_{1,i}, R^j_{2,i} \xleftarrow{R} \mathbb{G}_3\}_{i\in[1,n],j\in[0,k+2]}$. It additionally samples $h \xleftarrow{R} \mathbb{G}_1$, $\gamma \xleftarrow{R} \mathbb{Z}^*_{q_1}$ and $R_3 \xleftarrow{R} \mathbb{G}_3$, and sets:

$$
\begin{aligned}
Q &= g_2 \cdot R_3 \\
S^0_{1,i} &= h_{1,i} \cdot R^0_{1,i} \ , \ S^0_{2,i} = h_{2,i} \cdot R^0_{2,i} \text{ for } i \in [1,n] \\
S^j_{1,i} &= h_{1,i}^{x_{1,j}} \cdot R^j_{1,i} \ , \ S^j_{2,i} = h_{2,i}^{x_{2,j}} \cdot R^j_{2,i} \text{ for } i \in [1,n], j \in [1,k+2]
\end{aligned}
$$

It outputs the public parameter pp and the master secret-key msk as:

$$
\begin{aligned}
\mathsf{pp} &= \left(g_1, g_3, Q, \{S^j_{1,i}, S^j_{2,i}\}_{i\in[1,n],j\in[0,k+2]}, \hat{e}\left(g_1, h\right)^{\gamma}\right) \\
\mathsf{msk} &= \left(q_1, q_2, q_3, g_2, \{g_{1,j}, g_{2,j}\}_{j\in[1,k+2]}, \{g_{1,j}^{x_{1,j}} \cdot g_{1,k+2}^{x_{1,k+2}}, g_{2,j}^{x_{2,j}} \cdot g_{2,k+2}^{x_{2,k+2}}\}_{j\in[1,k+1]}, \{h_{1,i}, h_{2,i}\}_{i\in[1,n]}, h^{\gamma}\right)
\end{aligned}
$$

- **KeyGen:** On input the master secret-key msk and a vector $\overrightarrow{v} = (v_1, \cdots, v_n)$, the key generation algorithm samples $\{z_{1,i}, z_{2,i} \xleftarrow{R} \mathbb{Z}^*_{q_1}\}_{i\in[1,n]}$, $\{y^j_{1,i}, y^j_{2,i} \xleftarrow{R} \mathbb{Z}^*_{q_1}\}_{i\in[1,n],j\in[1,k+1]}$, $Q_4 \xleftarrow{R} \mathbb{G}_2$, $R_5 \xleftarrow{R} \mathbb{G}_3$ and $f_1, f_2 \xleftarrow{R} \mathbb{Z}^*_{q_2}$. It then sets $d_0 = Q_4 \cdot R_5 \Big/ \left(h^{\gamma} \cdot \prod_{i=1}^{n} h_{1,i}^{z_{1,i}} \cdot h_{2,i}^{z_{2,i}}\right)$. It also sets:

$$
\begin{aligned}
d^0_{1,i} &= g_1^{z_{1,i}} \cdot g_2^{f_1 \cdot v_i} \Bigg/ \left(\prod_{j=1}^{k+1} \left(g_{1,j}^{x_{1,j}} \cdot g_{1,k+2}^{x_{1,k+2}}\right)^{y^j_{1,i}}\right) \text{ for } i \in [1,n] \\
d^0_{2,i} &= g_1^{z_{2,i}} \cdot g_2^{f_2 \cdot v_i} \Bigg/ \left(\prod_{j=1}^{k+1} \left(g_{2,j}^{x_{2,j}} \cdot g_{2,k+2}^{x_{2,k+2}}\right)^{y^j_{2,i}}\right) \text{ for } i \in [1,n]
\end{aligned}
$$

Finally, it sets the following additional components:

$$
\begin{aligned}
d^j_{1,i} &= g_{1,j}^{y^j_{1,i}} \quad , \quad d^j_{2,i} = g_{2,j}^{y^j_{2,i}} \text{ for } i \in [1,n], j \in [1,k+1] \\
d^{k+2}_{1,i} &= g_{1,k+2}^{\sum_{j=1}^{k+1} y^j_{1,i}} \quad , \quad d^{k+2}_{2,i} = g_{2,k+2}^{\sum_{j=1}^{k+1} y^j_{2,i}} \text{ for } i \in [1,n]
\end{aligned}
$$

and outputs the secret-key $\mathsf{sk}_{\overrightarrow{v}}$ as:

$$
\mathsf{sk}_{\overrightarrow{v}} = \left(d_0, \{d^j_{1,i}, d^j_{2,i}\}_{i\in[1,n],j\in[0,k+2]}\right)
$$

- **Enc:** On input the public parameter pp, an attribute $I = (I_1, \cdots, I_n) \in \Sigma$ and a payload message $M \in \mathcal{M}$, the encryption algorithm samples $r, \alpha, \beta \xleftarrow{R} \mathbb{Z}^*_N$ and $\{R^j_{6,i}, R^j_{7,i} \xleftarrow{R} \mathbb{G}_3\}_{i\in[1,n],j\in[0,k+2]}$. It then sets $c_0 = g_1^r$. It also sets :

$$
\begin{aligned}
c^0_{1,i} &= \left(S^0_{1,i}\right)^r \cdot Q^{\alpha\cdot I_i} \cdot R^0_{6,i} \quad , \quad c^0_{2,i} = \left(S^0_{2,i}\right)^r \cdot Q^{\beta\cdot I_i} \cdot R^0_{7,i} \text{ for } i \in [1,n] \\
c^j_{1,i} &= \left(S^j_{1,i}\right)^r \cdot R^j_{6,i} \quad , \quad c^j_{2,i} = \left(S^j_{2,i}\right)^r \cdot R^j_{7,i} \text{ for } i \in [1,n], j \in [1,k+2]
\end{aligned}
$$

Finally, it sets $c_3 = M \cdot \left(\hat{e}\left(g_1, h\right)^{\gamma}\right)^r$ and outputs the ciphertext C as:

$$
C = \left(c_0, \{c^j_{1,i}, c^j_{2,i}\}_{i\in[1,n],j\in[0,k+2]}, c_3\right)
$$

- **Dec:** On input a ciphertext $C = \left(c_0, \{c_{1,i}^j, c_{2,i}^j\}_{i\in[1,n], j\in[0,k+2]}\right)$ and a secret-key $\mathsf{sk}_{\overrightarrow{v}} = \left(d_0, \{d_{1,i}^j, d_{2,i}^j\}_{i\in[1,n], j\in[0,k+2]}\right)$, the decryption algorithm computes:

$$M' = c_3 \cdot \hat{e}\left(d_0, c_0\right) \cdot \left(\prod_{i=1}^{n}\prod_{j=0}^{k+2} \hat{e}\left(d_{1,i}^j, c_{1,i}^j\right) \cdot \hat{e}\left(d_{2,i}^j, c_{2,i}^j\right)\right)$$

  If $M' \in \mathcal{M}$, the decryption algorithm outputs M', else it outputs $\perp$.

Correctness. To see that correctness holds for our zero-IPE scheme, let C and $\mathsf{sk}_{\overrightarrow{v}}$ be as described in Section 5. Then we have:

$$\begin{aligned}
M' &= c_3 \cdot \hat{e}\left(d_0, c_0\right) \cdot \left(\prod_{i=1}^{n}\prod_{j=0}^{k+2} \hat{e}\left(d_{1,i}^j, c_{1,i}^j\right) \cdot \hat{e}\left(d_{2,i}^j, c_{2,i}^j\right)\right) \\
&= M \cdot \left(\hat{e}\left(g_1, h\right)^{\gamma}\right)^r \cdot \left(\frac{\prod_{i=1}^{n} \hat{e}\left(g_1^{z_{1,i}}, h_{1,i}^r\right) \cdot \hat{e}\left(g_1^{z_{2,i}}, h_{2,i}^r\right)}{\hat{e}\left(h^{\gamma}, g_1^r\right) \cdot \hat{e}\left(\prod_{i=1}^{n} h_{1,i}^{z_{1,i}} \cdot h_{2,i}^{z_{2,i}}, g_1^r\right)}\right) \cdot \left(\prod_{i=1}^{n} \hat{e}\left(g_2^{f_1\cdot v_i}, g_2^{\alpha\cdot I_i}\right) \cdot \hat{e}\left(g_2^{f_2\cdot v_i}, g_2^{\beta\cdot I_i}\right)\right) \\
&\cdot \prod_{i=1}^{n}\left(\frac{\prod_{j=1}^{k+1}\left(\hat{e}\left(g_{1,j}^{y_{1,i}^j}, \left(h_{1,i}^{x_{1,j}}\right)^r\right) \cdot \hat{e}\left(g_{2,j}^{y_{2,i}^j}, \left(h_{2,i}^{x_{2,j}}\right)^r\right)\right) \cdot \hat{e}\left(g_{1,k+2}^{\sum_{j=1}^{k+1} y_{1,i}^j}, \left(h_{1,i}^{x_{1,k+2}}\right)^r\right) \cdot \hat{e}\left(g_{2,k+2}^{\sum_{j=1}^{k+1} y_{2,i}^j}, \left(h_{2,i}^{x_{2,k+2}}\right)^r\right)}{\hat{e}\left(\prod_{j=1}^{k+1}\left(g_{1,j}^{x_{1,j}} \cdot g_{1,k+2}^{x_{1,k+2}}\right)^{y_{1,i}^j}, h_{1,i}^r\right) \cdot \hat{e}\left(\prod_{j=1}^{k+1}\left(g_{2,j}^{x_{2,j}} \cdot g_{2,k+2}^{x_{2,k+2}}\right)^{y_{2,i}^j}, h_{2,i}^r\right)}\right) \\
&= M \cdot \prod_{i=1}^{n} \hat{e}\left(g_2, g_2\right)^{(\alpha f_1 + \beta f_2)\cdot x_i \cdot v_i} \\
&= M \cdot \hat{e}\left(g_2, g_2\right)^{(\alpha f_1 + \beta f_2 \bmod q_2)\cdot\langle\overrightarrow{v}, I\rangle}
\end{aligned}$$

where α, β are uniformly random in $\mathbb{Z}_N^*$ and f_1, f_2 are uniformly random in $\mathbb{Z}_{q_2}^*$. If $\langle\overrightarrow{v}, I\rangle = 0 \bmod N$, then we have $M' = M$. If $\langle\overrightarrow{v}, I\rangle \neq 0 \bmod N$, there are two cases: if $\langle\overrightarrow{v}, I\rangle \neq 0 \bmod q_2$, then with all but negligible probability (over random choice of α, β, f_1, f_2), M' does not lie in $\mathbb{G}_T$ (since $\mathcal{M}$ is a small subset of $\mathbb{G}_T$). Otherwise, we have $\langle\overrightarrow{v}, I\rangle = 0 \bmod q_2$, in which case M' will always be equal to M; however, this would reveal a non-trivial factor of N, and so this too occurs with negligible probability. In fact, the data privacy property of this zero-IPE construction relies on a set of assumptions in bilinear groups that imply the hardness of finding a non-trivial factor of N.

The Need for Two Sub-Systems. Note that our zero-IPE scheme uses two parallel sub-systems (in the key generation and encryption algorithms) that are apparently redundant since they perform the same functions. Indeed, our scheme inherits this feature from the original zero-IPE scheme of Katz, Sahai and Waters [3]. Eliminating one of the sub-systems from our scheme would retain functional correctness as well as computational function privacy, while also improving performance and efficiency. However, the proof methodology in [3] for data privacy relies on the existence of the parallel sub-systems in an essential way. Since our aim is to retain the same data privacy guarantees as in the original scheme, we stick to the use of two parallel sub-systems in our augmented zero-IPE scheme.

5.1 Security of Our IPE Scheme

Data Privacy. We state the following theorem for the data privacy of Π_k^{IPE}:

Theorem 5.1 *Our zero-IPE scheme Π_k^{IPE} retains the selective data privacy guarantees of the original zero-IPE scheme of Katz, Sahai and Waters* [3].

Proof Overview. We provide a brief overview of the proof technique for our scheme, which essentially follows the proof technique presented in [3]. We consider a probabilistic polynomial-time adversary that tries to determine whether the *challenge ciphertext* is associated with either of the two attributes I_0 or I_1. The proof proceeds via a sequence of hybrid games in which an entire attribute used in the challenge ciphertext is changed in one step, instead of changing them component by component for reasons mentioned in the proof of the original scheme in [3]. This is facilitated by the presence of the two parallel sub-systems, which allows the hybrid games to use *ill-formed* ciphertexts that are encrypted with respect to two *different* attributes I and I' and in the two sub-systems. Let such a ciphertext be denoted informally as (I, I'). The proof establishes indistinguishability between the well-formed ciphertexts (I_0, I_0) and (I_1, I_1) via a sequence of intermediate hybrid games using the ill-formed ciphertexts $\left(I_0, \overrightarrow{0}\right)$, (I_0, I_1) and $\left(\overrightarrow{0}, I_1\right)$. The zero vector is used since it orthogonal to any other vector. The simulator in our proof works in one sub-system independent of what happens in the other one. In each hybrid game, the simulator embeds a subgroup-decision like assumption in the challenge ciphertext, and the structure of the challenge determines whether a sub-system embeds a given vector or a zero vector. This is essentially an adoption of the proof technique originally presented in [3] to our function private scheme.

An additional requirement in our proof is that the simulator should be able to embed the $(k+1)$-DLIN instances when responding to the key generation queries from the adversary. As demonstrated in the proof of Theorem 4.1, this is straightforward to achieve: since the simulator in the data privacy game is allowed to set up the $(k+1)$-DLIN instances entirely on its own, it can easily augment the secret-key generation process in the proof of the original scheme by appropriately embedding these instances where necessary. Moreover, since the $(k+1)$-DLIN instances are sampled uniformly at random, the resulting distribution of secret-keys is exactly as in the real world from the point of view of the adversary. Similarly, in the challenge phase, the simulator generates the additional components in the augmented ciphertext uniformly at random, without altering the nature of the ciphertext distribution from the adversary's point of view.

Computational Function Privacy. We state the following theorem for the computational function privacy of Π_k:

Theorem 5.2 *Our zero-IPE scheme Π_k^{IPE} is computationally function private under the* $(k+1)$-DLIN *assumption for predicate vectors sampled uniformly from (n, k)-block sources with $k = \omega(\log \lambda)$.*

Proof. The proof aims to show that any probabilistic poly-time adversary $\mathcal{A}$ cannot distinguish between the real and random modes of operation of the function privacy oracle, provided that the oracle is queried with circuits that sample sufficiently unpredictable distributions over the space of predicates. In particular, such distributions should be (n, k)-block sources over $\mathbb{Z}_N^n$, such that each component of a vector $\overrightarrow{v}$ sampled from an adversarially chosen distribution has a min-entropy of $k = \omega(\log \lambda)$, and is uncorrelated with all other components. We define a series of hybrid experiments $\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda)$ for $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$ and $m \in [0, n]$ as follows:

- $\mathsf{Expt}^{\mathsf{mode},0}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda)$ is exactly identical to $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda)$.
- $\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda)$ for $m \in [1, n]$ is identical to $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda)$ except that the secret-key $\mathsf{sk}_{\overrightarrow{v^*}} = \left(d^*_0, \{d^{*j}_{1,i}, d^{*j}_{2,i}\}_{i\in[1,n],j\in[0,k+2]}\right)$ generated by the real-or-random oracle is such that the set of components $\{d^{*j}_{1,i}, d^{*j}_{2,i}\}_{i\in[1,m],j\in[0,k+2]}$ are uniformly random and independent of the underlying vector $\overrightarrow{v^*}$.

Quite evidently, the following holds:

$$\left| \Pr\left[\mathsf{Expt}^{\mathsf{real},n}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand},n}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda) = 1\right] \right| = 0 \qquad (1)$$

We now state and prove the following claim:

Claim 5.1 *For any probabilistic polynomial-time adversary $\mathcal{A}$, for* $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$ *and for* $m \in [0, n-1]$*, the following holds:*

$$\left| \Pr\left[\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},m+1}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

To prove this claim, we assume the contrary. Let $\mathcal{A}$ be a probabilistic polynomial-time adversary such that:

$$\left| \Pr\left[\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},m+1}_{\mathsf{FP},\Pi^{\mathrm{IPE}}_k,\mathcal{A}}(\lambda) = 1\right] \right| = \epsilon > \mathsf{negl}(\lambda)$$

for some $m \in [0, n-1]$. We construct an algorithm $\mathcal{B}$ such that:

$$\Bigg| \Pr\left[\mathcal{B}\left(\left(g_{1,1}, \cdots, g_{1,k+2}, g_{1,1}^{a_1}, \cdots, g_{1,k+2}^{\sum_{j=1}^{k+1} a_j}\right), \left(g_{2,1}, \cdots, g_{2,k+2}, g_{2,1}^{a'_1}, \cdots, g_{2,k+2}^{\sum_{j=1}^{k+1} a'_j}\right)\right) = 1\right] -$$
$$\Pr\left[\mathcal{B}\left(\left(g_{1,1}, \cdots, g_{1,k+2}, g_{1,1}^{a_1}, \cdots, g_{1,k+2}^{a_{k+2}}\right), \left(g_{2,1}, \cdots, g_{2,k+2}, g_{2,1}^{a'_1}, \cdots, g_{2,k+2}^{a'_{k+2}}\right)\right) = 1\right] \Bigg| = \epsilon$$

where the probability is over random choice of $\{a_j, a'_j \xleftarrow{R} \mathbb{Z}^*_{q_1}\}_{j\in[1,k+2]}$, and over random choice of $\{g_{1,j}, g_{2,j} \xleftarrow{R} \mathbb{G}_1\}_{j\in[1,k+2]}$ ($\mathbb{G}_1$ being a group of prime order q_1). Observe that $\mathcal{B}$ can in turn be trivially used to construct another algorithm that has advantage at least ϵ in solving a given instance of the $(k+1)$-DLIN problem. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

- ***Setup:*** $\mathcal{B}$ uniformly samples two other λ-bit primes q_2, q_3 apart from q_1 which is the order of the group $\mathbb{G}_1$ in its input instance. It also sets $N = q_1 q_2 q_3$, and then sets up the groups $\mathbb{G}$, $\mathbb{G}_2$ and $\mathbb{G}_3$ of order N, q_2 and q_3 respectively, along with $\hat{e} : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. It then samples $\{x_{1,j}, x_{2,j} \xleftarrow{R} \mathbb{Z}^*_{q_1}\}_{j\in[1,k+2]}$, $\{h_{1,i}, h_{2,i} \xleftarrow{R} \mathbb{G}_1\}_{i\in[1,n]}$ and $\{R^j_{1,i}, R^j_{2,i} \xleftarrow{R} \mathbb{G}_3\}_{i\in[1,n],j\in[0,k+2]}$. It additionally samples $h \xleftarrow{R} \mathbb{G}_1$, $\gamma \xleftarrow{R} \mathbb{Z}^*_{q_1}$ and $R_3 \xleftarrow{R} \mathbb{G}_3$, and sets:

$$\begin{aligned} &Q = g_2 \cdot R_3 \\ &S^0_{1,i} = h_{1,i} \cdot R^0_{1,i} \ , \ S^0_{2,i} = h_{2,i} \cdot R^0_{2,i} \text{ for } i \in [1, n] \\ &S^j_{1,i} = h_{1,i}^{x_{1,j}} \cdot R^j_{1,i} \ , \ S^j_{2,i} = h_{2,i}^{x_{2,j}} \cdot R^j_{2,i} \text{ for } i \in [1, n], j \in [1, k+2] \end{aligned}$$

Finally, it sets the public parameter pp as:

$$\mathsf{pp} = \left(g_1, g_3, Q, \{S_{1,i}^j, S_{2,i}^j\}_{i\in[1,n],j\in[0,k+2]}, \hat{e}\left(g_1, h\right)^{\gamma}\right)$$

and provides the same to $\mathcal{A}$. Observe that pp is distributed exactly as in the real world.

- ***Secret-Key Queries:*** When $\mathcal{A}$ issues a secret-key query for $\overrightarrow{v} \in \mathbb{Z}_N^n$, $\mathcal{B}$ samples $\{z_{1,i}, z_{2,i} \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,n]}$, $\{y_{1,i}^j, y_{2,i}^j \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,n],j\in[1,k+1]}$, $Q_4 \xleftarrow{R} \mathbb{G}_2$, $R_5 \xleftarrow{R} \mathbb{G}_3$ and $f_1, f_2 \xleftarrow{R} \mathbb{Z}_{q_2}^*$. It then sets:

$$d_0 = Q_4 \cdot R_5 \bigg/ \left(\prod_{i=1}^{n} h_{1,i}^{z_{1,i}} \cdot h_{2,i}^{z_{2,i}}\right)$$

$$d_{1,i}^0 = g_1^{z_{1,i}} \cdot g_2^{f_1\cdot v_i} \bigg/ \left(\prod_{j=1}^{k+1} \left(g_{1,j}^{x_{1,j}} \cdot g_{1,k+2}^{x_{1,k+2}}\right)^{y_{1,i}^j}\right) \quad \text{for } i \in [1,n]$$

$$d_{2,i}^0 = g_1^{z_{2,i}} \cdot g_2^{f_2\cdot v_i} \bigg/ \left(\prod_{j=1}^{k+1} \left(g_{2,j}^{x_{2,j}} \cdot g_{2,k+2}^{x_{2,k+2}}\right)^{y_{2,i}^j}\right) \quad \text{for } i \in [1,n]$$

  Finally, it sets the following additional components:

$$d_{1,i}^j = g_{1,j}^{y_{1,i}^j} \quad , \quad d_{2,i}^j = g_{2,j}^{y_{2,i}^j} \text{ for } i \in [1,n], j \in [1,k+1]$$

$$d_{1,i}^{k+2} = g_{1,k+2}^{\sum_{j=1}^{k+1} y_{1,i}^j} \quad , \quad d_{2,i}^{k+2} = g_{2,k+2}^{\sum_{j=1}^{k+1} y_{2,i}^j} \text{ for } i \in [1,n]$$

  and outputs the secret-key $\mathsf{sk}_{\overrightarrow{v}}$ as:

$$\mathsf{sk}_{\overrightarrow{v}} = \left(d_0, \{d_{1,i}^j, d_{2,i}^j\}_{i\in[1,n],j\in[0,k+2]}\right)$$

- ***Real-or-Random Query:*** Suppose $\mathcal{A}$ queries the real-or-random oracle with an (n,k)-block source $\mathbf{V}^* = (V_1^*, \cdots, V_n^*)$ over $\mathbb{Z}_N^n$ such that $k = \omega(\log\lambda)$. $\mathcal{B}$ samples $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$ and does the following:

  1. For each $i \in [1,n]$, $\mathcal{B}$ samples $v_i^* \xleftarrow{R} V_i^*$ if $\mathsf{mode} = \mathsf{real}$, or $v_i^* \xleftarrow{R} \mathbb{Z}_N$ if $\mathsf{mode} = \mathsf{rand}$. The vector $\overrightarrow{v^*} = (v_1^*, \cdots, v_n^*)$ is the challenge vector that $\mathcal{B}$ uses to respond to the query from $\mathcal{A}$.

  2. As in the response to the secret-key queries, $\mathcal{B}$ samples $\{z_{1,i}, z_{2,i} \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,n]}$, $\{y_{1,i}^j, y_{2,i}^j \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,n]\setminus\{m+1\},j\in[1,k+1]}$, $Q_4 \xleftarrow{R} \mathbb{G}_2$, $R_5 \xleftarrow{R} \mathbb{G}_3$ and $f_1, f_2 \xleftarrow{R} \mathbb{Z}_{q_2}^*$. It also samples $\{y_{1,i}^{k+2}, y_{2,i}^{k+2} \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,m]}$. Observe that $\mathcal{B}$ does not sample $y_{1,m+1}^j$ or $y_{2,m+1}^j$ for $j \in [1,k+2]$. This is because $\mathcal{B}$ embeds the its input pair of $(k+1)$-DLIN instances in its response by formally setting $y_{1,m+1}^j = a_j$ and $y_{2,m+1}^j = a'_j$ for each $j \in [1,k+2]$ as described next.

3. $\mathcal{B}$ now sets the various components of the secret-key as follows:

$$d_0^* = Q_4 \cdot R_5 \Big/ \left(\prod_{i=1}^{n} h_{1,i}^{z_{1,i}} \cdot h_{2,i}^{z_{2,i}} \right)$$

$$d^{*0}_{1,i} = g_1^{z_{1,i}} \cdot g_2^{f_1 \cdot v_i} \Big/ \left(\prod_{j=1}^{k+1} \left(g_{1,j}^{x_{1,j}} \cdot g_{1,k+2}^{x_{1,k+2}} \right)^{y_{1,i}^j} \right) \text{ for } i \in [1,n] \setminus \{m+1\}$$

$$d^{*0}_{2,i} = g_1^{z_{2,i}} \cdot g_2^{f_2 \cdot v_i} \Big/ \left(\prod_{j=1}^{k+1} \left(g_{2,j}^{x_{2,j}} \cdot g_{2,k+2}^{x_{2,k+2}} \right)^{y_{2,i}^j} \right) \text{ for } i \in [1,n] \setminus \{m+1\}$$

It also sets the following additional components:

$$d^{*j}_{1,i} = g_{1,j}^{y_{1,i}^j} \quad , \quad d^{*j}_{2,i} = g_{2,j}^{y_{2,i}^j} \text{ for } i \in [1,n] \setminus \{m+1\}, j \in [1,k+1]$$

$$d^{*k+2}_{1,i} = g_{1,k+2}^{y_{1,i}^{k+2}} \quad , \quad d^{*k+2}_{2,i} = g_{2,k+2}^{y_{2,i}^{k+2}} \text{ for } i \in [1,m]$$

$$d^{*k+2}_{1,i} = g_{1,k+2}^{\sum_{j=1}^{k+1} y_{1,i}^j} \quad , \quad d^{*k+2}_{2,i} = g_{2,k+2}^{\sum_{j=1}^{k+1} y_{2,i}^j} \text{ for } i \in [m+2,n]$$

Observe that the first m components of the secret-key are crafted to be uniformly random, while the last $(n-m-1)$ components are well-formed.

4. Finally, $\mathcal{B}$ embeds its input pair of $(k+1)$-DLIN instances in the $(m+1)^{\text{th}}$ component of the secret-key by setting:

$$d^{*0}_{1,m+1} = g_1^{z_{1,m+1}} \cdot g_1^{f_1 \cdot v_{m+1}} \Big/ \left(\prod_{j=1}^{k+2} \left(g_{1,j}^{a'_j} \right)^{x_{1,j}} \right)$$

$$d^{*0}_{2,m+1} = g_1^{z_{2,m+1}} \cdot g_2^{f_2 \cdot v_{m+1}} \Big/ \left(\prod_{j=1}^{k+2} \left(g_{2,j}^{a'_j} \right)^{x_{2,j}} \right)$$

$$d^{*j}_{1,m+1} = g_{1,j}^{a_j} \quad , \quad d^{*j}_{2,m+1} = g_{2,j}^{a'_j} \text{ for } j \in [1,k+2]$$

$\mathcal{B}$ responds to $\mathcal{A}$ with the secret-key $\mathsf{sk}_{\overrightarrow{v^*}}$ as:

$$\mathsf{sk}_{\overrightarrow{v^*}} = \left(d_0^*, \{ d^{*j}_{1,i}, d^{*j}_{2,i} \}_{i \in [1,n], j \in [0,k+2]} \right)$$

- ***Guess:*** At the end of the game, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs the same bit b'.

It is easy to see that when $a_{k+2} = \sum_{j=1}^{k+1} a_j$ and $a'_{k+2} = \sum_{j=1}^{k+1} a'_j$, the secret-key $\mathsf{sk}_{\overrightarrow{v^*}}$ is identically distributed to the response of the real-or-random oracle in the experiment $\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$. On the other hand, when either or both of a_{k+2} and a'_{k+2} are uniformly random in $\mathbb{Z}_q^*$, the secret-key $\mathsf{sk}_{\overrightarrow{v^*}}$ is identically distributed to the response of the real-or-random oracle in the experiment $\mathsf{Expt}^{\mathsf{mode},m+1}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$. It follows readily that $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$ in solving its input instance pair. This completes the proof of Claim 5.1.

We now make the following observation:

$$\begin{aligned}
&\left|\Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},n}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&\quad \leq \sum_{m=0}^{n-1}\left|\Pr\left[\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},m+1}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&\quad \leq \mathsf{negl}(\lambda) \text{ (from Claim 5.1) for } n = \mathsf{poly}(\lambda)
\end{aligned}$$

Consequently, for $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$, we have:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{FP}}_{\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) &= \left|\Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&\leq \left|\Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{real},n}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&\quad + \left|\Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand},n}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&\quad\quad + \left|\Pr\left[\mathsf{Expt}^{\mathsf{real},n}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand},n}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&\leq 2\left|\Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},n}_{\mathsf{FP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| \text{ (from Equation 1)} \\
&= 2\epsilon' \leq \mathsf{negl}(\lambda)
\end{aligned}$$

This completes the proof of function privacy for Π_k^{IPE}. Note that the proof does not essentially rely on the presence of two parallel sub-systems in our scheme. In fact, eliminating one of the subsystems gives a simpler proof. However, the two sub-system version allows us to adopt the proof of data privacy in [3] for our scheme.

Enhanced Computational Function Privacy. We state the following theorem for the enhanced computational function privacy of Π_k:

Theorem 5.3 *Our zero-IPE scheme Π_k^{IPE} is computationally enhanced function private under the $(k+1)$-DLIN assumption for predicate vectors sampled uniformly from (n,k)-block sources with $k = \omega(\log\lambda)$.*

*Proof Overview***.** Once again, the proof is very similar to that of Theorem 5.2; hence we restrict to providing a proof overview here. The detailed proof is presented in Appendix C. The proof basically shows that the simulator $\mathcal{B}$ can additionally simulate the function privacy encryption oracle, and that the real and random modes of operation of the function privacy oracle are indistinguishable even in the presence of the encryption oracle. We again consider a series of hybrid experiments that gradually randomize the secret-key $\mathsf{sk}_{\vec{v}^*}$ generated by the real-or-random function privacy oracle component by component. The challenge is to simulate the function privacy encryption oracle such that its behavior does not provide the adversary with any additional distinguisher between two consecutive experiments.

Let $\mathsf{sk}_{\overrightarrow{v^*}} = \left(d_0^*, \{{d^*}_{1,i}^j, {d^*}_{2,i}^j\}_{i\in[1,n],j\in[0,k+2]}\right)$ be the response of the real-or-random function privacy oracle to the adversary $\mathcal{A}$. On receiving a function-privacy encryption oracle query for a payload message M, $\mathcal{B}$ responds as follows: it samples $I^* = (I_1^*, \cdots, I_n^*)$ such that $\left\langle \overrightarrow{v^*}, I^* \right\rangle = 0$. It then samples $r, \alpha, \beta \xleftarrow{R} \mathbb{Z}_N^*$ and $\{R_{6,i}^j, R_{7,i}^j \xleftarrow{R} \mathbb{G}_3\}_{i\in[1,n],j\in[0,k+2]}$ and sets $c_0^* = g_1^r$. It also sets:

$$
\begin{aligned}
{c^*}_{1,i}^0 = \left(S_{1,i}^0\right)^r \cdot Q^{\alpha\cdot I_i^*} \cdot R_{6,i}^0 \ \ , \ \ {c^*}_{2,i}^0 = \left(S_{2,i}^0\right)^r \cdot Q^{\beta\cdot I_i^*} \cdot R_{7,i}^0 \text{ for } i \in [1,n] \\
{c^*}_{1,i}^j = \left(S_{1,i}^j\right)^r \cdot R_{6,i}^j \ \ , \ \ {c^*}_{2,i}^j = \left(S_{2,i}^j\right)^r \cdot R_{7,i}^j \text{ for } i \in [1,n], j \in [1,k+2]
\end{aligned}
$$

Finally, it sets:

$$
c_3^* = M \Big/ \hat{e}\left(d_0^*, c_0^*\right) \cdot \left(\prod_{i=1}^{n}\prod_{j=0}^{k+2} \hat{e}\left({d^*}_{1,i}^j, {c^*}_{1,i}^j\right) \cdot \hat{e}\left({d^*}_{2,i}^j, {c^*}_{2,i}^j\right)\right)
$$

and outputs the ciphertext $C^* = \left(c_0^*, \{{c^*}_{1,i}^j, {c^*}_{2,i}^j\}_{i\in[1,n],j\in[0,k+2]}, c_3^*\right)$.

Clearly, the ciphertext C^* is dependent solely on $\mathsf{sk}_{\overrightarrow{v^*}}$ and the input message M, and is hence independent of the vector $\overrightarrow{v^*}$. In addition, it produces M upon decryption using $\mathsf{sk}_{\overrightarrow{v^*}}$ irrespective of $\mathcal{B}$'s input challenge. Consequently, the distribution of C^* in the two experiments is indistinguishable from $\mathcal{A}$'s point of view. This in turn implies that the adversary's advantage in distinguishing between two hybrid experiments is the same as $\mathcal{B}$'s advantage in solving its input $(k+1)$-DLIN challenge pair.

6 Extensions and Open Problems

Our work solves the open problem of constructing computationally function private predicate encryption systems in the private key setting. Our approach, denoted as *encrypt-augment-recover*, offers a methodology for converting an existing public-key predicate encryption scheme into a computationally function private one, without compromising the data privacy guarantees of the original scheme. Our approach yields the first fully data private and computationally function private constructions for IBE and public-key IPE, and, in general, for searchable encryption schemes supporting a wide range of predicates. In this section, we discuss some interesting extensions and open problems that arise from our work.

Extension to Private-Key Predicate Encryption. Our *encrypt-augment-recover* methodology is equally applicable for achieving computationally function private predicate encryption schemes in the private-key setting, even when the underlying predicates are not necessarily sampled from distributions with at least super-logarithmic min-entropy. In particular, the core function privacy arguments for our constructions presented in this paper do not essentially rely on the unpredictability of the predicate distributions; this assumption is additionally made to rule out trivial attacks in the public-key setting. In the private-key setting, where the adversary does not have access to an encryption oracle, our approach anticipates an expansion to the existing body of work in designing function private predicate encryption schemes.

Extension to Multi-Input Predicate Encryption. In this work, we have focused on designing function private predicate encryption systems in the single-input setting. Multi-input predicate encryption (MIPE) introduced by Goldwasser et al. [30] is a generalization of functional encryption to the setting of multi-input predicates. An MIPE scheme has several encryption slots and each decryption key sk_f for a multi-input predicate f jointly decrypts the ciphertexts $\mathsf{Enc}(I_1), ..., \mathsf{Enc}(I_n)$ for all slots to obtain $f(I_1, ..., I_n)$ without revealing anything more about the encrypted attributes. In particular, this provides a framework to evaluate *bounded-norm* multi-input IPE: each predicate is specified by a collection of vectors $\overrightarrow{v}_1, \cdots, \overrightarrow{v}_n$, and takes as input a collection of vectors $\overrightarrow{x}_1, \cdots, \overrightarrow{x}_n$ to output $f_{\overrightarrow{v}_1,\cdots,\overrightarrow{v}_n}(\overrightarrow{x}_1, \cdots, \overrightarrow{x}_n) = \sum_{i=1}^{n} \{\overrightarrow{v}_i, \overrightarrow{x}_i\}$.

We point out that our technique can be easily generalized to obtain function private IPE schemes in the multi-input setting as follows: we first use our technique to obtain a function private IPE construction in the single-input setting, and then run n independent copies of this construction. The i^{th} copy is used to encrypt $\overrightarrow{x}_i$ in the i^{th} slot, while the new secret-key is the ensemble of the n secret-keys corresponding to $\overrightarrow{v}_1, \cdots, \overrightarrow{v}_n$. The decryption algorithm computes each inner product individually, and returns their sum. Although this means that the adversary also learns each individual inner product, this is an inherent leakage in the public-key setting and does not weaken the security guarantees. The data privacy guarantees of the underlying scheme ensure no further leakage, while the function privacy guarantees of the underlying scheme continue to hold as long as each $\overrightarrow{v}_i$ is sampled from block sources with sufficient min-entropy, and is independent of the other $n-1$ vectors.

Generalization of Our Approach. In this work, we have demonstrated concrete constructions for function private IBE and IPE in the public-key setting based on existing constructions in the literature. An interesting open problem is to explore whether our approach can be generalized so as to be applicable to *any* public-key predicate encryption scheme, particularly those based on lattices [2], without compromising on their data privacy guarantees. It would also be interesting to identify the properties (if any) of existing predicate encryption schemes that make them amenable to modification using our approach.

Hidden Vector Encryption and Polynomial Evaluation. Boneh and Waters [1] proposed hidden vector encryption (HVE), a pre-cursor to IPE, that supports search using conjunctive, range and comparison-based query predicates. In HVE, attributes correspond to vectors over an alphabet Σ, while secret-keys correspond to predicate vectors over the augmented alphabet $\Sigma_\star = \Sigma \cup \{\star\}$ containing the wild card character $\star$. Decryption succeeds if the attribute matches the predicate vector in every coordinate that is not $\star$. We note that although IPE can be used to realize HVE [3], our computational function privacy definitions do not naturally extend to HVE. In particular, the presence of the wild card character $\star$ in the predicate vectors of HVE trivially violates our min-entropy requirements, making it difficult to *hide* their presence in the secret-key.

A weaker notion of function privacy for HVE, that our framework can realistically incorporate, is to ask that the secret-key reveals nothing more about the the predicate vector than the locations of the wild card character $\star$. We believe that HVE schemes satisfying this weaker notion of computational function privacy can be achieved using our *encrypt-augment-recover* approach. It is still an open problem, however, to formalize a stronger function privacy definition for HVE, and to realize function private constructions satisfying this definition. This would also provide insight into the limits of function privacy for searchable encryption schemes supporting comparison and range queries. Finally, it is also open to formalize security definitions and realize constructions for function private encryption schemes that support arbitrary polynomial evaluation predicates [3].

References

1. Dan Boneh and Brent Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 535–554, 2007.
2. Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional Encryption for Inner Product Predicates from Learning with Errors. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 21–40, 2011.
3. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. *J. Cryptology*, 26(2):191–224, 2013.
4. Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
5. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology–EUROCRYPT 2005*, pages 440–456. Springer, 2005.
6. Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. *J. Cryptology*, 21(3):350–391, 2008.
7. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 733–751, 2015.
8. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 333–362, 2016.
9. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 506–522, 2004.
10. Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-Private Identity-Based Encryption: Hiding the Function in Functional Encryption. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 461–478, 2013.
11. Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-Private Subspace-Membership Encryption and Its Applications. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, pages 255–275, 2013.
12. Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 79–88, 2006.
13. Emily Shen, Elaine Shi, and Brent Waters. Predicate Privacy in Encryption Systems. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 457–473, 2009.
14. Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, pages 470–491, 2015.

15. Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, pages 164–195, 2016.
16. Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 93–110, 2013.
17. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 13–25, 1998.
18. Zvika Brakerski and Gil Segev. Function-Private Functional Encryption in the Private-Key Setting. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 306–324, 2015.
19. Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.
20. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*, pages 44–55, 2000.
21. Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 442–455, 2005.
22. Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 577–594, 2010.
23. Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 965–976, 2012.
24. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 41–55, 2004.
25. Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *IACR Cryptology ePrint Archive*, 2007:74, 2007.
26. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206, 2008.
27. Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology-CRYPTO 2006*, pages 290–307. Springer, 2006.
28. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 98–115, 2010.
29. Kaoru Kurosawa and Le Trieu Phong. Leakage resilient IBE and IPE under the DLIN assumption. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pages 487–501, 2013.
30. Vipul Goyal, Aayush Jain, and Adam O'Neill. Multi-input functional encryption with unbounded-message security. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 531–556, 2016.

A Detailed Proof of Theorem 4.1

We now present the detailed proof. Let $\mathcal{A}$ be any probabilistic polynomial-time adversary. The proof aims to show the following:

$$\mathbf{Adv}^{\mathsf{DP}}_{\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)=1\right]\right| \leq \mathsf{negl}(\lambda)$$

We define a second experiment $\mathsf{Expt}^{(b)}_{\mathsf{rand},\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$ that is identical to $\mathsf{Expt}^{(b)}_{\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$ except in Step 3, where the challenge ciphertext C^* is generated uniformly and independent of the challenge pair (id_b^*, M_b^*). Then, the following is obvious to see:

$$\left|\Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{rand},\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{rand},\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)=1\right]\right| = 0$$

We now state and prove the following claim:

Claim A.1 *For any probabilistic polynomial-time adversary $\mathcal{A}$ and for $b \in \{0,1\}$, the following holds:*

$$\left|\Pr\left[\mathsf{Expt}^{(b)}_{\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Expt}^{(b)}_{\mathsf{rand},\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)=1\right]\right| \leq \mathsf{negl}(\lambda)$$

To prove this claim, we assume the contrary. Let $\mathcal{A}$ be a probabilistic polynomial-time adversary such that:

$$\left|\Pr\left[\mathsf{Expt}^{(b)}_{\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Expt}^{(b)}_{\mathsf{rand},\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)=1\right]\right| = \epsilon > \mathsf{negl}(\lambda)$$

Also, let $Q_K = \mathsf{poly}(\lambda)$ be the maximum number of secret-key queries made by $\mathcal{A}$ in either of the experiments $\mathsf{Expt}^{(b)}_{\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{(b)}_{\mathsf{rand},\mathsf{DP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$. We construct an algorithm $\mathcal{B}$ that solves an instance of the DBDH problem with non-negligible advantage $\epsilon' \geq \epsilon/\left(\mathsf{exp}(1)\cdot(Q_K+1)\right)$. $\mathcal{B}$ is given $(g, g^{a_1}, g^{a_2}, g^{a_3}, Z)$ and interacts with $\mathcal{A}$ as follows:

- ***Setup:*** $\mathcal{B}$ samples $x_1, \cdots, x_k \xleftarrow{R} \mathbb{Z}_q^*$ and $g_1, \cdots, g_{k+2} \xleftarrow{R} \mathbb{G}$. It sets $\mathsf{pp} = (g, g^{a_1}, g^{x_1}, \cdots, g^{x_{k+2}})$ and provides pp to $\mathcal{A}$. Observe that the distribution of pp is exactly as in the real world.

- $H-$***Queries:*** $\mathcal{B}$ maintains a list of tuples of the form $\langle \mathsf{id}_j, H(\mathsf{id}_j), \alpha_j, \beta_j\rangle$ such that $H(\mathsf{id}_j) \in \mathbb{G}$, $\alpha_j \in \mathbb{Z}_q^*$ and $\beta_j \in \{0,1\}$. When $\mathcal{A}$ issues a hash query for $\mathsf{id}_i \in \mathcal{ID}$, $\mathcal{B}$ responds as follows:

  1. $\mathcal{B}$ searches the list for a matching tuple of the form $\langle \mathsf{id}_i, H(\mathsf{id}_i), \alpha_i, \beta_i\rangle$. If such a tuple is found, it returns $H(\mathsf{id}_i)$ to $\mathcal{A}$.
  2. Otherwise, $\mathcal{B}$ samples $\alpha_i \xleftarrow{R} \mathbb{Z}_q^*$ and $\beta_i \leftarrow \{0,1\}$ such that $\Pr[\beta_i = 0] = 1 - 1/(Q_K+1)$.
     - If $\beta_i = 0$, it sets $H(\mathsf{id}_i) = g^{\alpha_i}$
     - If $\beta_i = 1$, it sets $H(\mathsf{id}_i) = (g^{a_2})^{\alpha_i}$, where g^{a_2} is a part of its input instance.
  3. $\mathcal{B}$ now adds the tuple $\langle \mathsf{id}_i, H(\mathsf{id}_i)\rangle$ to its existing list, and returns $H(\mathsf{id}_i)$ to $\mathcal{A}$.

- ***Secret-Key Queries:*** When $\mathcal{A}$ issues a secret-key query for $\mathsf{id}_i \in \mathcal{ID}$, $\mathcal{B}$ responds as follows:

1. $\mathcal{B}$ runs the aforementioned procedure to look up $H(\mathsf{id}_i)$ and obtains the tuple $\langle \mathsf{id}_i, H(\mathsf{id}_i), \alpha_i, \beta_i \rangle$.
2. If $\beta_i = 1$, $\mathcal{B}$ aborts by outputting a uniformly sampled bit.
3. If $\beta_i = 0$, we must have $H(\mathsf{id}_i) = g^{\alpha_i}$; hence, $\mathcal{B}$ chooses $y_1, \cdots, y_{k+1} \xleftarrow{R} \mathbb{Z}_q^*$ and responds with:

$$\mathsf{sk}_{\mathsf{id}_i} = \left(g_1^{y_1}, \cdots, g_2^{y_{k+1}}, g_{k+2}^{\sum_{j=1}^{k+1} y_j}, \left(\prod_{j=1}^{k+2} \left(g_j^{x_j} \cdot g_{k+2}^{x_{k+2}} \right)^{y_j} \right) \cdot (g^{a_1})^{\alpha_i} \right)$$

Once again, observe that the distribution of $\mathsf{sk}_{\mathsf{id}_i}$ is exactly as in the real world.

- ***Challenge:*** $\mathcal{A}$ outputs the challenge pair $((\mathsf{id}_0^*, M_0^*), (\mathsf{id}_1^*, M_1^*))$. $\mathcal{B}$ samples $b \xleftarrow{R} \{0,1\}$ and does the following:

1. $\mathcal{B}$ runs the aforementioned procedure to look up $H(\mathsf{id}_b^*)$ and obtains the tuple $\langle \mathsf{id}_b^*, H(\mathsf{id}_b^*), \alpha_b^*, \beta_b^* \rangle$.
2. If $\beta_b^* = 0$, $\mathcal{B}$ aborts by outputting a uniformly sampled bit.
3. If $\beta_b^* = 1$, we must have $H(\mathsf{id}_b^*) = (g^{a_2})^{\alpha_b^*}$. $\mathcal{B}$ accordingly responds with the challenge ciphertext C^* as:

$$C^* = \left((g^{a_3})^{(\alpha_b^*)^{-1}}, (g^{a_3})^{x_1 \cdot (\alpha_b^*)^{-1}}, \cdots, (g^{a_3})^{x_{k+2} \cdot (\alpha_b^*)^{-1}}, M_b^* \cdot Z \right)$$

where g^{a_3} and Z are part of its input instance.

- ***Guess:*** At the end of the game, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs the same bit b'.

It is easy to see that when $Z = e(g,g)^{a_1 \cdot a_2 \cdot a_3}$, the ciphertext C^* is well-formed and identically distributed to the challenge in the experiment $\mathsf{Expt}^{(b)}_{\mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda)$. To see this, set $s = a_1$ and $r = a_3 \cdot (\alpha_b^*)^{-1}$ and observe that:

$$(g^{a_3})^{(\alpha_b^*)^{-1}} = g^r \;\;,\;\; (g^{a_3})^{x_j \cdot (\alpha_b^*)^{-1}} = (g^{x_j})^r \;\text{ for } j \in [1, k+2]$$
$$Z = e(g,g)^{a_1 \cdot a_2 \cdot a_3} = e\left((g^{a_2})^{\alpha_b^*}, g^{a_1} \right)^{a_3 \cdot (\alpha_b^*)^{-1}} = e\left(H(\mathsf{id}_b^*), g^s \right)^r$$

On the other hand, if Z is uniform random in $\mathbb{G}_T$, then C^* is independent of b and hence identically distributed to the challenge in the experiment $\mathsf{Expt}^{(b)}_{\mathsf{rand}, \mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda)$.

It remains to bound the probability that $\mathcal{B}$ aborts either during a secret-key query or during the challenge phase, denoted as $\Pr[\mathsf{Abort}]$. It is easy to see that the probability that $\mathcal{B}$ aborts during a given secret-key query is $1/(Q_K + 1)$. Assuming that $\mathcal{B}$ makes at most Q_K secret-key queries, the probability that t never aborts for any of these queries is thus lower bounded as $(1 - 1/(Q_K+1))^{Q_K}$. The probability that $\mathcal{B}$ does not abort during the challenge phase is lower bounded as $1/(Q_K + 1)$. Thus the overall probability that $\mathcal{B}$ does not abort during the entire game (equivalently $\Pr\left[\overline{\mathsf{Abort}}\right]$) is lower bounded as $(1 - 1/(Q_K+1))^{Q_K} / (Q_K + 1)$.

Finally, the advantage ϵ' of $\mathcal{B}$ in solving the DBDH instance (where the probability is taken over all possible choices of $a_1, a_2, a_3 \xleftarrow{R} \mathbb{Z}_q^*$) and all possible choice of $Z \xleftarrow{R} \mathbb{G}_T$) may be quantified as:

$$\begin{aligned}
\epsilon' &= \left| \Pr\left[\mathcal{B}\left(g, g^{a_1}, g^{a_2}, g^{a_3}, e\left(g,g\right)^{a_1 \cdot a_2 \cdot a_3}\right) = 1\right] - \Pr\left[\mathcal{B}\left(g, g^{a_1}, g^{a_2}, g^{a_3}, Z\right) = 1\right] \right| \\
&= |\Pr\left[\mathcal{B}\left(g, g^{a_1}, g^{a_2}, g^{a_3}, e\left(g,g\right)^{a_1 \cdot a_2 \cdot a_3}\right) = 1 | \overline{\mathsf{Abort}}\right] \cdot \Pr\left[\overline{\mathsf{Abort}}\right] - \\
&\qquad \Pr\left[\mathcal{B}\left(g, g^{a_1}, g^{a_2}, g^{a_3}, Z\right) = 1 | \overline{\mathsf{Abort}}\right] \cdot \Pr\left[\overline{\mathsf{Abort}}\right] | \\
&= \left| \Pr\left[\mathsf{Expt}^{(b)}_{\mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \cdot \Pr\left[\overline{\mathsf{Abort}}\right] - \Pr\left[\mathsf{Expt}^{(b)}_{\mathsf{rand}, \mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \cdot \Pr\left[\overline{\mathsf{Abort}}\right] \right| \\
&\geq \epsilon \cdot \left(1 - 1/\left(Q_K + 1\right)\right)^{Q_K} / \left(Q_K + 1\right) \\
&\geq \epsilon / \left(\mathsf{exp}(1) \cdot \left(Q_K + 1\right)\right)
\end{aligned}$$

The above derivation uses the fact that the abort condition for $\mathcal{B}$ is independent of the view of the adversary $\mathcal{A}$. This completes the proof of Claim A.1. The proof of data privacy for Π_k^{IBE} now follows from the following observation:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{DP}}_{\Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) &= \left| \Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \right| \\
&\leq \left| \Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{rand}, \mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \right| + \\
&\qquad \left| \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{rand}, \mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \right| + \\
&\qquad\quad \left| \Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{rand}, \mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{rand}, \mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \right| \\
&\leq 2 \left| \Pr\left[\mathsf{Expt}^{(b)}_{\mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(b)}_{\mathsf{rand}, \mathsf{DP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \right| \text{ (from Equation 1)} \\
&= 2\epsilon \leq \mathsf{negl}(\lambda)
\end{aligned}$$

The above proof reinforces the fact that applying our *encrypt-augment-cancel* approach *does not affect* the adaptive data privacy of the original IBE scheme of Boneh and Franklin.

B Detailed Proof of Theorem 4.3

We begin by stating the following claim:

Claim B.1 *For any probabilistic polynomial-time adversary $\mathcal{A}$ and for* $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$, *the following holds:*

$$\left| \Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

To prove this claim, we assume the contrary. Let $\mathcal{A}$ be a probabilistic polynomial-time adversary such that for $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$, we have:

$$\left| \Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP}, \Pi_k^{\mathrm{IBE}}, \mathcal{A}}(\lambda) = 1\right] \right| = \epsilon > \mathsf{negl}(\lambda)$$

We assume that the adversary $\mathcal{A}$ issues a single query to the real-or-random oracle. As discussed in Section 3, such a single-shot adversary is polynomially equivalent to its multi-shot variant considered in Definition 3.1. We construct an algorithm $\mathcal{B}$ that solves an instance of the $(k+1)$-DLIN problem with non-negligible advantage $\epsilon' = \epsilon$. $\mathcal{B}$ is given $\left(g_1, \cdots, g_{k+2}, g_1^{a_1}, \cdots, g_{k+2}^{a_{k+2}}\right)$ and interacts with $\mathcal{A}$ as follows:

- ***Setup:*** $\mathcal{B}$ samples $s, x_1, \cdots, x_{k+2} \xleftarrow{R} \mathbb{Z}_q^*$ and a generator $g \xleftarrow{R} \mathbb{G}$. It sets $\mathsf{pp} = (g, g^s, g^{x_1}, \cdots, g^{x_{k+2}})$ and provides pp to $\mathcal{A}$. Observe that the distribution of pp is exactly as in the real world.

- $H-$ ***Queries:*** $\mathcal{B}$ maintains a list of tuples of the form $\langle \mathsf{id}_j, H(\mathsf{id}_j)\rangle$ such that $H(\mathsf{id}_j) \in \mathbb{G}$. When $\mathcal{A}$ issues a hash query for $\mathsf{id}_i \in \mathcal{ID}$, $\mathcal{B}$ responds as follows:

  1. $\mathcal{B}$ searches the list for a matching tuple of the form $\langle \mathsf{id}_i, H(\mathsf{id}_i)\rangle$. If such a tuple is found, it returns $H(\mathsf{id}_i)$ to $\mathcal{A}$.
  2. Otherwise, $\mathcal{B}$ samples $H(\mathsf{id}_i) \xleftarrow{R} \mathbb{G}$ and adds the tuple $\langle \mathsf{id}_i, H(\mathsf{id}_i)\rangle$ to its existing list. It returns $H(\mathsf{id}_i)$ to $\mathcal{A}$.

- ***Secret-Key Queries:*** When $\mathcal{A}$ issues a secret-key query for $\mathsf{id}_i \in \mathcal{ID}$, $\mathcal{B}$ responds as follows:
  1. $\mathcal{B}$ runs the aforementioned procedure to look up $H(\mathsf{id}_i)$.
  2. $\mathcal{B}$ samples $y_1, \cdots, y_{k+1} \xleftarrow{R} \mathbb{Z}_q^*$ and responds with:

  $$\mathsf{sk}_{\mathsf{id}_i} = \left(g_1^{y_1}, \cdots, g_{k+1}^{y_{k+1}}, g_{k+2}^{\sum_{j=1}^{k+1} y_j}, \left(\prod_{j=1}^{k+1} \left(g_j^{x_j} \cdot g_{k+2}^{x_{k+2}} \right)^{y_j} \right) \cdot (H(\mathsf{id}_i))^s \right)$$

  where $g_1, \cdots, g_{k+2}$ are part of its input instance. Once again, observe that the distribution of $\mathsf{sk}_{\mathsf{id}_i}$ is exactly as in the real world.

- ***Real-or-Random Query:*** Suppose $\mathcal{A}$ queries the real-or-random oracle with $\mathbf{ID}^*$ - a circuit representing a k-source over the identity space $\mathcal{ID}$ such that $k = \omega(\log \lambda)$. $\mathcal{B}$ samples $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$ and does the following:
  1. If $\mathsf{mode} = \mathsf{real}$, $\mathcal{B}$ samples $\mathsf{id}^* \xleftarrow{R} \mathbf{ID}^*$, while if $\mathsf{mode} = \mathsf{rand}$, it samples $\mathsf{id}^* \xleftarrow{R} \mathcal{ID}$. It then runs the aforementioned procedure to look up $H(\mathsf{id}^*)$.
  2. $\mathcal{B}$ responds with the secret-key $\mathsf{sk}_{\mathsf{id}^*}$ as:

  $$\mathsf{sk}_{\mathsf{id}^*} = \left(g_1^{a_1}, \cdots, g_{k+2}^{a_{k+2}}, \left(\prod_{j=1}^{k+2} \left(g_j^{a_j} \right)^{x_j} \right) \cdot (H(\mathsf{id}^*))^s \right)$$

  where $g_1^{a_1}, \cdots, g_{k+2}^{a_{k+2}}$ are part of its input instance.

- ***Function Privacy Encryption Oracle Query:*** Suppose $\mathcal{A}$ queries the function privacy encryption oracle with a message M. Let $\mathsf{sk}_{\mathsf{id}^*} = \left(d_0^*, \cdots, d_{k+2}^*\right)$ be the response of the real-or-random function privacy oracle to the adversary $\mathcal{A}$. On receiving a function-privacy encryption oracle query for a payload message M, $\mathcal{B}$ responds as follows: it samples $r \xleftarrow{R} \mathbb{Z}_q^*$ and outputs the ciphertext $C^* = \left(c_0^*, \cdots, c_{k+3}^*\right)$ where:

  $$c_0^* = g^r \ , \ c_j^* = \left(g^{x_j}\right)^r \ \text{ for } j \in [1, k+2]$$

and

$$c_{k+3}^* = M \cdot \frac{e\left(d_{k+2}^*, c_0^*\right)}{\prod_{j=1}^{k+2} e\left(d_{j-1}^*, c_j^*\right)}$$

- ***Guess:*** At the end of the game, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs the same bit b'.

It is now easy to see the following:

- When $a_{k+2} = \sum_{j=1}^{k+1} a_j$, the secret-key $\mathsf{sk}_{\mathsf{id}^*}$ is well-formed and identically distributed to the response of the real-or-random oracle in the experiment $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$. Additionally, the ciphertext C^* is also well-formed and identically distributed to the response of the function privacy encryption oracle in the experiment $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$.
- On the other hand, when a_{k+2} is uniformly random in $\mathbb{Z}_q^*$, the secret-key $\mathsf{sk}_{\mathsf{id}^*}$ is uniformly random, and hence identically distributed to the response of the real-or-random oracle in the experiment $\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$. Correspondingly, the ciphertext C^* is also uniformly random, and hence identically distributed to the response of the function privacy encryption oracle in the experiment $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda)$.
- In either case, decrypting C^* using $\mathsf{sk}_{\mathsf{id}^*}$ correctly retrieves M. This essentially implies that they correspond to the same underlying identity, which is either id^* sampled by $\mathcal{B}$ when $a_{k+2} = \sum_{j=1}^{k+1} a_j$, or some other uniformly random identity in $\mathcal{ID}$ when a_{k+2} is uniformly random in $\mathbb{Z}_q^*$.

Now, the advantage ϵ' of $\mathcal{B}$ in solving the $(k+1)$-DLIN instance (where the probability is taken over all possible choices of $a_1, \cdots, a_{k+2} \xleftarrow{R} \mathbb{Z}_q^*$ and all possible choices of $g_1, \cdots, g_{k+2} \xleftarrow{R} \mathbb{G}$) may be quantified as:

$$\begin{aligned}
\epsilon' &= \left|\Pr\left[\mathcal{B}\left(g_1, \cdots, g_{k+2}, g_1^{a_1}, \cdots, g_{k+2}^{\sum_{j=1}^{k+1} a_j}\right) = 1\right] - \Pr\left[\mathcal{B}\left(g_1, \cdots, g_{k+2}, g_1^{a_1}, \cdots, g_{k+2}^{a_{k+2}}\right) = 1\right]\right| \\
&= \left|\Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&= \epsilon
\end{aligned}$$

This completes the proof of Claim B.1. The proof of enhanced function privacy for Π_k^{IBE} now follows from the following observation:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{EFP}}_{\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) &= \left|\Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&\leq 2\left|\Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP},\Pi_k^{\mathrm{IBE}},\mathcal{A}}(\lambda) = 1\right]\right| \\
&= 2\epsilon \leq \mathsf{negl}(\lambda)
\end{aligned}$$

C Detailed Proof of Theorem 5.3

We present the detailed proof here. The proof aims to show that any probabilistic poly-time adversary $\mathcal{A}$ cannot distinguish between the real and random modes of operation of the function privacy oracle, provided that the oracle is queried with circuits that sample sufficiently unpredictable distributions over the space of predicates. In particular, such distributions should be

(n,k)-block sources over $\mathbb{Z}_N^n$, such that each component of a vector $\overrightarrow{v}$ sampled from an adversarially chosen distribution has a min-entropy of $k = \omega(\log \lambda)$, and is uncorrelated with all other components. We define a series of hybrid experiments $\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$ for $\mathsf{mode} \in \{\mathsf{real},\mathsf{rand}\}$ and $m \in [0,n]$ as follows:

- $\mathsf{Expt}^{\mathsf{mode},0}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$ is exactly identical to $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$.
- $\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$ for $m \in [1,n]$ is identical to $\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$ except that the secret-key $\mathsf{sk}_{\overrightarrow{v^*}} = \left(d_0^*, \{{d^*}^j_{1,i}, {d^*}^j_{2,i}\}_{i\in[1,n],j\in[0,k+2]}\right)$ generated by the real-or-random oracle is such that the set of components $\{{d^*}^j_{1,i}, {d^*}^j_{2,i}\}_{i\in[1,m],j\in[0,k+2]}$ are uniformly random and independent of the underlying vector $\overrightarrow{v^*}$. In addition, the ciphertext C^* for a message M generated by the function privacy encryption oracle is uniformly random and independent of $\overrightarrow{v^*}$; it, however, produces M upon decryption using the $\mathsf{sk}_{\overrightarrow{v^*}}$ generated by the real-or-random oracle.

Quite evidently, the following holds:

$$\left|\Pr\left[\mathsf{Expt}^{\mathsf{real},n}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand},n}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| = 0 \tag{2}$$

We now state and prove the following claim:

Claim C.1 *For any probabilistic polynomial-time adversary $\mathcal{A}$, for* $\mathsf{mode} \in \{\mathsf{real},\mathsf{rand}\}$ *and for $m \in [0,n-1]$, the following holds:*

$$\left|\Pr\left[\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},m+1}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| \le \mathsf{negl}(\lambda)$$

To prove this claim, we assume the contrary. Let $\mathcal{A}$ be a probabilistic polynomial-time adversary such that:

$$\left|\Pr\left[\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},m+1}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\right| = \epsilon > \mathsf{negl}(\lambda)$$

for some $m \in [0,n-1]$. We construct an algorithm $\mathcal{B}$ such that:

$$\Bigg|\Pr\left[\mathcal{B}\left(\left(g_{1,1},\cdots,g_{1,k+2},g_{1,1}^{a_1},\cdots,g_{1,k+2}^{\sum_{j=1}^{k+1}a_j}\right),\left(g_{2,1},\cdots,g_{2,k+2},g_{2,1}^{a'_1},\cdots,g_{2,k+2}^{\sum_{j=1}^{k+1}a'_j}\right)\right)=1\right] -$$
$$\Pr\left[\mathcal{B}\left(\left(g_{1,1},\cdots,g_{1,k+2},g_{1,1}^{a_1},\cdots,g_{1,k+2}^{a_{k+2}}\right),\left(g_{2,1},\cdots,g_{2,k+2},g_{2,1}^{a'_1},\cdots,g_{2,k+2}^{a'_{k+2}}\right)\right)=1\right]\Bigg| = \epsilon$$

where the probability is over random choice of $\{a_j, a'_j \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{j\in[1,k+2]}$, and over random choice of $\{g_{1,j}, g_{2,j} \xleftarrow{R} \mathbb{G}_1\}_{j\in[1,k+2]}$ ($\mathbb{G}_1$ being a group of prime order q_1). Observe that $\mathcal{B}$ can in turn be trivially used to construct another algorithm that has advantage at least ϵ in solving a given instance of the $(k+1)$-DLIN problem. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

- ***Setup:*** $\mathcal{B}$ uniformly samples two other λ-bit primes q_2, q_3 apart from q_1 which is the order of the group $\mathbb{G}_1$ in its input instance. It also sets $N = q_1q_2q_3$, and then sets up the groups $\mathbb{G}$, $\mathbb{G}_2$ and $\mathbb{G}_3$ of order N, q_2 and q_3 respectively, along with the bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \longrightarrow$

$\mathbb{G}_T$. It then samples $\{x_{1,j}, x_{2,j} \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{j\in[1,k+2]}$, $\{h_{1,i}, h_{2,i} \xleftarrow{R} \mathbb{G}_1\}_{i\in[1,n]}$ and $\{R_{1,i}^j, R_{2,i}^j \xleftarrow{R} \mathbb{G}_3\}_{i\in[1,n],j\in[0,k+2]}$. It additionally samples $h \xleftarrow{R} \mathbb{G}_1$, $\gamma \xleftarrow{R} \mathbb{Z}_{q_1}^*$ and $R_3 \xleftarrow{R} \mathbb{G}_3$, and sets:

$$\begin{aligned} Q &= g_2 \cdot R_3 \\ S_{1,i}^0 &= h_{1,i} \cdot R_{1,i}^0 \ , \ S_{2,i}^0 = h_{2,i} \cdot R_{2,i}^0 \text{ for } i \in [1,n] \\ S_{1,i}^j &= h_{1,i}^{x_{1,j}} \cdot R_{1,i}^j \ , \ S_{2,i}^j = h_{2,i}^{x_{2,j}} \cdot R_{2,i}^j \text{ for } i \in [1,n], j \in [1,k+2] \end{aligned}$$

Finally, it sets the public parameter pp as:

$$\mathsf{pp} = \left(g_1, g_3, Q, \{S_{1,i}^j, S_{2,i}^j\}_{i\in[1,n],j\in[0,k+2]}, \hat{e}\left(g_1, h\right)^\gamma\right)$$

and provides the same to $\mathcal{A}$. Observe that pp is distributed exactly as in the real world.

- ***Secret-Key Queries:*** When $\mathcal{A}$ issues a secret-key query for $\overrightarrow{v} \in \mathbb{Z}_N^n$, $\mathcal{B}$ samples $\{z_{1,i}, z_{2,i} \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,n]}$, $\{y_{1,i}^j, y_{2,i}^j \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,n],j\in[1,k+1]}$, $Q_4 \xleftarrow{R} \mathbb{G}_2$, $R_5 \xleftarrow{R} \mathbb{G}_3$ and $f_1, f_2 \xleftarrow{R} \mathbb{Z}_{q_2}^*$. It then sets:

$$\begin{aligned} d_0 &= Q_4 \cdot R_5 \Big/ \left(\prod_{i=1}^{n} h_{1,i}^{z_{1,i}} \cdot h_{2,i}^{z_{2,i}}\right) \\ d_{1,i}^0 &= g_1^{z_{1,i}} \cdot g_2^{f_1 \cdot v_i} \Big/ \left(\prod_{j=1}^{k+1} \left(g_{1,j}^{x_{1,j}} \cdot g_{1,k+2}^{x_{1,k+2}}\right)^{y_{1,i}^j}\right) \text{ for } i \in [1,n] \\ d_{2,i}^0 &= g_1^{z_{2,i}} \cdot g_2^{f_2 \cdot v_i} \Big/ \left(\prod_{j=1}^{k+1} \left(g_{2,j}^{x_{2,j}} \cdot g_{2,k+2}^{x_{2,k+2}}\right)^{y_{2,i}^j}\right) \text{ for } i \in [1,n] \end{aligned}$$

  Finally, it sets the following additional components:

$$\begin{aligned} d_{1,i}^j &= g_{1,j}^{y_{1,i}^j} \ , \ d_{2,i}^j = g_{2,j}^{y_{2,i}^j} \text{ for } i \in [1,n], j \in [1,k+1] \\ d_{1,i}^{k+2} &= g_{1,k+2}^{\sum_{j=1}^{k+1} y_{1,i}^j} \ , \ d_{2,i}^{k+2} = g_{2,k+2}^{\sum_{j=1}^{k+1} y_{2,i}^j} \text{ for } i \in [1,n] \end{aligned}$$

  and outputs the secret-key $\mathsf{sk}_{\overrightarrow{v}}$ as:

$$\mathsf{sk}_{\overrightarrow{v}} = \left(d_0, \{d_{1,i}^j, d_{2,i}^j\}_{i\in[1,n],j\in[0,k+2]}\right)$$

- ***Real-or-Random Query:*** Suppose $\mathcal{A}$ queries the real-or-random oracle with an (n,k)-block source $\mathbf{V}^* = (V_1^*, \cdots, V_n^*)$ over $\mathbb{Z}_N^n$ such that $k = \omega(\log \lambda)$. $\mathcal{B}$ samples $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$ and does the following:

  1. For each $i \in [1,n]$, $\mathcal{B}$ samples $v_i^* \xleftarrow{R} V_i^*$ if $\mathsf{mode} = \mathsf{real}$, or $v_i^* \xleftarrow{R} \mathbb{Z}_N$ if $\mathsf{mode} = \mathsf{rand}$. The vector $\overrightarrow{v^*} = (v_1^*, \cdots, v_n^*)$ is the challenge vector that $\mathcal{B}$ uses to respond to the query from $\mathcal{A}$.

  2. As in the response to the secret-key queries, $\mathcal{B}$ samples $\{z_{1,i}, z_{2,i} \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,n]}$, $\{y_{1,i}^j, y_{2,i}^j \xleftarrow{R} \mathbb{Z}_{q_1}^*\}_{i\in[1,n]\setminus\{m+1\},j\in[1,k+1]}$, $Q_4 \xleftarrow{R} \mathbb{G}_2$, $R_5 \xleftarrow{R} \mathbb{G}_3$ and $f_1, f_2 \xleftarrow{R} \mathbb{Z}_{q_2}^*$. It also samples

$\{y^{k+2}_{1,i}, y^{k+2}_{2,i} \xleftarrow{R} \mathbb{Z}^*_{q_1}\}_{i\in[1,m]}$. Observe that $\mathcal{B}$ does not sample $y^j_{1,m+1}$ or $y^j_{2,m+1}$ for $j \in [1, k+2]$. This is because $\mathcal{B}$ embeds the its input pair of $(k+1)$-DLIN instances in its response by formally setting $y^j_{1,m+1} = a_j$ and $y^j_{2,m+1} = a'_j$ for each $j \in [1, k+2]$ as described next.

3. $\mathcal{B}$ now sets the various components of the secret-key as follows:

$$d^*_0 = Q_4 \cdot R_5 \Big/ \left(\prod_{i=1}^{n} h^{z_{1,i}}_{1,i} \cdot h^{z_{2,i}}_{2,i}\right)$$

$$d^{*0}_{1,i} = g_1^{z_{1,i}} \cdot g_2^{f_1\cdot v_i} \Big/ \left(\prod_{j=1}^{k+1} \left(g^{x_{1,j}}_{1,j} \cdot g^{x_{1,k+2}}_{1,k+2}\right)^{y^j_{1,i}}\right) \quad \text{for } i \in [1,n]\setminus\{m+1\}$$

$$d^{*0}_{2,i} = g_1^{z_{2,i}} \cdot g_2^{f_2\cdot v_i} \Big/ \left(\prod_{j=1}^{k+1} \left(g^{x_{2,j}}_{2,j} \cdot g^{x_{2,k+2}}_{2,k+2}\right)^{y^j_{2,i}}\right) \quad \text{for } i \in [1,n]\setminus\{m+1\}$$

It also sets the following additional components:

$$d^{*j}_{1,i} = g^{y^j_{1,i}}_{1,j} \quad , \quad d^{*j}_{2,i} = g^{y^j_{2,i}}_{2,j} \text{ for } i \in [1,n]\setminus\{m+1\}, j \in [1,k+1]$$

$$d^{*k+2}_{1,i} = g^{y^{k+2}_{1,i}}_{1,k+2} \quad , \quad d^{*k+2}_{2,i} = g^{y^{k+2}_{2,i}}_{2,k+2} \text{ for } i \in [1,m]$$

$$d^{*k+2}_{1,i} = g^{\sum_{j=1}^{k+1} y^j_{1,i}}_{1,k+2} \quad , \quad d^{*k+2}_{2,i} = g^{\sum_{j=1}^{k+1} y^j_{2,i}}_{2,k+2} \text{ for } i \in [m+2,n]$$

Observe that the first m components of the secret-key are crafted to be uniformly random, while the last $(n-m-1)$ components are well-formed.

4. Finally, $\mathcal{B}$ embeds its input pair of $(k+1)$-DLIN instances in the $(m+1)^{\text{th}}$ component of the secret-key by setting:

$$d^{*0}_{1,m+1} = g_1^{z_{1,m+1}} \cdot g_1^{f_1\cdot v_{m+1}} \Big/ \left(\prod_{j=1}^{k+2} \left(g^{a'_j}_{1,j}\right)^{x_{1,j}}\right)$$

$$d^{*0}_{2,m+1} = g_1^{z_{2,m+1}} \cdot g_2^{f_2\cdot v_{m+1}} \Big/ \left(\prod_{j=1}^{k+2} \left(g^{a'_j}_{2,j}\right)^{x_{2,j}}\right)$$

$$d^{*j}_{1,m+1} = g^{a_j}_{1,j} \quad , \quad d^{*j}_{2,m+1} = g^{a'_j}_{2,j} \text{ for } j \in [1,k+2]$$

$\mathcal{B}$ responds to $\mathcal{A}$ with the secret-key $\mathsf{sk}_{\vec{v}^*}$ as:

$$\mathsf{sk}_{\vec{v}^*} = \left(d^*_0, \{d^{*j}_{1,i}, d^{*j}_{2,i}\}_{i\in[1,n], j\in[0,k+2]}\right)$$

- ***Function Privacy Encryption Oracle Query:*** Let $\mathsf{sk}_{\vec{v}^*} = \left(d^*_0, \{d^{*j}_{1,i}, d^{*j}_{2,i}\}_{i\in[1,n], j\in[0,k+2]}\right)$ be the response of the real-or-random function privacy oracle to the adversary $\mathcal{A}$. Suppose

$\mathcal{A}$ queries the function privacy encryption oracle with a message M. $\mathcal{B}$ responds as follows: it samples $I^* = (I_1^*, \cdots, I_n^*)$ such that $\left\langle \overrightarrow{v^*}, I^* \right\rangle = 0$. It then samples $r, \alpha, \beta \xleftarrow{R} \mathbb{Z}_N^*$ and $\{R_{6,i}^j, R_{7,i}^j \xleftarrow{R} \mathbb{G}_3\}_{i\in[1,n],j\in[0,k+2]}$ and sets $c_0^* = g_1^r$. It also sets:

$$c^{*0}_{1,i} = \left(S_{1,i}^0\right)^r \cdot Q^{\alpha \cdot I_i^*} \cdot R_{6,i}^0 \quad , \quad c^{*0}_{2,i} = \left(S_{2,i}^0\right)^r \cdot Q^{\beta \cdot I_i^*} \cdot R_{7,i}^0 \text{ for } i \in [1,n]$$
$$c^{*j}_{1,i} = \left(S_{1,i}^j\right)^r \cdot R_{6,i}^j \quad , \quad c^{*j}_{2,i} = \left(S_{2,i}^j\right)^r \cdot R_{7,i}^j \text{ for } i \in [1,n], j \in [1,k+2]$$

Finally, it sets:

$$c_3^* = M \Big/ \hat{e}\left(d_0^*, c_0^*\right) \cdot \left(\prod_{i=1}^{n} \prod_{j=0}^{k+2} \hat{e}\left(d^{*j}_{1,i}, c^{*j}_{1,i}\right) \cdot \hat{e}\left(d^{*j}_{2,i}, c^{*j}_{2,i}\right) \right)$$

and outputs the ciphertext $C^* = \left(c_0^*, \{c^{*j}_{1,i}, c^{*j}_{2,i}\}_{i\in[1,n],j\in[0,k+2]}, c_3^*\right)$. Clearly, for $m = 0$, the ciphertext C^* is well-formed, while for $m \in [1,n]$ independent of $\overrightarrow{v^*}$ and is crafted by $\mathcal{B}$ solely based on $\mathsf{sk}_{\overrightarrow{v^*}}$ to produce the correct output M upon decryption.

- ***Guess:*** At the end of the game, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs the same bit b'.

It is easy to see that when $a_{k+2} = \sum_{j=1}^{k+1} a_j$ and $a'_{k+2} = \sum_{j=1}^{k+1} a'_j$, the secret-key $\mathsf{sk}_{\overrightarrow{v^*}}$ is identically distributed to the response of the real-or-random oracle in the experiment $\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$. On the other hand, when either or both of a_{k+2} and a'_{k+2} are uniformly random in $\mathbb{Z}_q^*$, the secret-key $\mathsf{sk}_{\overrightarrow{v^*}}$ is identically distributed to the response of the real-or-random oracle in the experiment $\mathsf{Expt}^{\mathsf{mode},m+1}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda)$. In either case, the ciphertext C^* is dependent solely on $\mathsf{sk}_{\overrightarrow{v^*}}$ and the input message M, and is hence independent of the vector $\overrightarrow{v^*}$. In addition, it produces M upon decryption using $\mathsf{sk}_{\overrightarrow{v^*}}$ irrespective of whether $a_{k+2} = \sum_{j=1}^{k+1} a_j$ and $a'_{k+2} = \sum_{j=1}^{k+1} a'_j$. Consequently, the distribution of C^* in the two experiments is indistinguishable from $\mathcal{A}$'s point of view. It follows readily that $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$ in solving its input instance pair. This completes the proof of Claim C.1.

We now make the following observation:

$$\begin{aligned}
&|\Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},n}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\Big| \\
&\quad \leq \sum_{m=0}^{n-1} \Big|\Pr\left[\mathsf{Expt}^{\mathsf{mode},m}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},m+1}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right]\Big| \\
&\quad \leq \mathsf{negl}(\lambda) \text{ (from Claim C.1)for } n = \mathsf{poly}(\lambda)
\end{aligned}$$

Consequently, for $\mathsf{mode} \xleftarrow{R} \{\mathsf{real}, \mathsf{rand}\}$, we have:

$$
\begin{aligned}
\mathbf{Adv}^{\mathsf{EFP}}_{\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) &= \left| \Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] \right| \\
&\leq \left| \Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{real},n}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] \right| \\
&\quad + \left| \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand},n}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] \right| \\
&\quad\quad + \left| \Pr\left[\mathsf{Expt}^{\mathsf{real},n}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand},n}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] \right| \\
&\leq 2\left| \Pr\left[\mathsf{Expt}^{\mathsf{mode}}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{mode},n}_{\mathsf{EFP},\Pi_k^{\mathrm{IPE}},\mathcal{A}}(\lambda) = 1\right] \right| \text{ (from Equation 2)} \\
&= 2\epsilon' \leq \mathsf{negl}(\lambda)
\end{aligned}
$$

This completes the proof of function privacy for Π_k^{IPE}. Note that the proof does not essentially rely on the presence of two parallel sub-systems in our scheme. In fact, eliminating one of the subsystems gives a simpler proof. However, the two sub-system version allows us to adopt the proof of data privacy in [3] for our scheme.