Succinct Predicate and Online-Offline Multi-Input Inner Product Encryptions under Standard Static Assumptions

Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay

Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur-721302, India
{pratishdatta,ratna,sourav}@maths.iitkgp.ernet.in

Abstract. This paper presents expressive *predicate encryption* (PE) systems, namely *non-zero inner-product-predicate encryption* (NIPPE) and *attribute-based encryption* (ABE) supporting monotone span programs achieving *best* known parameters among existing similar schemes under *well-studied static* complexity assumptions. Both the constructions are built in composite order bilinear group setting and involve only 2 group elements in the ciphertexts. More interestingly, our NIPPE scheme, which additionally features only 1 group element in the decryption keys, is the *first* to attain succinct ciphertexts and decryption keys simultaneously. For proving selective security of these constructions under the *Subgroup Decision* assumptions, which are the most standard static assumptions in composite order bilinear group setting, we apply the extended version of the elegant *Déjà Q* framework, which was originally proposed as a general technique for reducing the q-type complexity assumptions to their static counter parts. Our work thus demonstrates the power of this framework in overcoming the need of q-type assumptions, which are vulnerable to serious practical attacks, for deriving security of highly expressive PE systems with compact parameters. We further introduce the concept of *online-offline multi-input functional encryption* (OO-MIFE), which is a crucial advancement towards realizing this highly promising but computationally intensive cryptographic primitive in resource bounded and power constrained devices. We also instantiate our notion of OO-MIFE by constructing such a scheme for the multi-input analog of the *inner product* functionality, which has a wide range of application in practice. Our OO-MIFE scheme for multi-input inner products is built in asymmetric bilinear groups of prime order and is proven selectively secure under the *well-studied k-Linear* (k-LIN) assumption.

Keywords: inner-product-predicate encryption, attribute-based encryption, Déjà Q, online-offline multi-input functional encryption

1 Introduction

FE: *Functional encryption* (FE) [BSW11, O'N10] is a new vision of modern cryptography that aims to overcome the potential limitation of the traditional encryption schemes, namely, the all or nothing control over decryption capabilities. FE supports restricted decryption keys which enable a decrypter to learn specific functions of encrypted messages, and nothing more. More precisely, an FE scheme for a function family $\mathfrak{F}$ involves a setup authority which holds a master secret key and publishes public system parameters. An encrypter uses the public parameters (along with a secret encryption key provided by the setup authority in case of a private key scheme) to encrypt its message $\mathscr{M}$ belonging to some supported message space $\mathbb{M}$ creating a ciphertext $\text{CT}(\mathscr{M})$. A decrypter may obtain a private decryption key $\text{SK}(\mathscr{F})$ corresponding to some $\mathscr{F} \in \mathfrak{F}$ from the setup authority provided that the authority deems that the decrypter is entitled for that key. Such a decryption key $\text{SK}(\mathscr{F})$ can be used to decrypt $\text{CT}(\mathscr{M})$ to recover $\mathscr{F}(\mathscr{M})$. The standard security notion for FE is collusion resistance, i.e., an arbitrary number of decrypters cannot jointly retrieve any more information about an encrypted message beyond the union of what they each can learn individually.

PE: An important subclass of FE is *predicate encryption* (PE) with *public index*. Consider a predicate family $\mathfrak{P} = \{\mathscr{P}_{\mathscr{Y}} : \mathbb{X} \to \{0,1\} \mid \mathscr{Y} \in \mathfrak{Y}\}$, where $\mathbb{X}$ and $\mathfrak{Y}$ are index sets. In a PE scheme for the predicate family $\mathfrak{P}$, the associated message space $\mathbb{M}$ is of the form $\mathbb{X} \times \mathbb{W}$, where $\mathbb{W}$ contains the actual payloads. The functionality $\mathscr{F}_{\mathscr{P}_{\mathscr{Y}}}$ associated with a predicate $\mathscr{P}_{\mathscr{Y}} \in \mathfrak{P}$ is defined as $\mathscr{F}_{\mathscr{P}_{\mathscr{Y}}}(\mathscr{X}, \mathscr{W}) = \mathscr{W}$, if $\mathscr{P}_{\mathscr{Y}}(\mathscr{X}) = 1$, and the empty string $\perp$, otherwise, for all $(\mathscr{X}, \mathscr{W}) \in \mathbb{M} = \mathbb{X} \times \mathbb{W}$. In the public index setting, a PE ciphertext $\text{CT}(\mathscr{M})$ encrypting some message $\mathscr{M} = (\mathscr{X}, \mathscr{W})$ includes the index $\mathscr{X}$ in the clear.

Starting with *identity-based encryption* (IBE) [BF01, Wat05], which corresponds to the equality predicate, PE has progressively evolved through a sequence of distinguished works to support more and more expressive predicate families. At the same time, achieving compact parameters and security in the standard model under static assumptions has been a persisting research direction in PE. In this work we consider public-index PE for two specific class of predicates, namely, the *non-zero inner-product predicate* and *attribute-based access policies*.

NIPPE: In case of the *non-zero inner-product-predicate encryption* (NIPPE) the index sets $\mathbb{X} = \mathbb{Y} = \mathbb{Z}_n^\ell$, for some $n, \ell \in \mathbb{N}$. A predicate $\mathscr{P}_{\vec{y}} : \mathbb{Z}_n^\ell \to \{0,1\}$ associated with some vector $\vec{y} \in \mathbb{Z}_n^\ell$ is defined for all $\vec{x} \in \mathbb{Z}_n^\ell$ as $\mathscr{P}_{\vec{y}}(\vec{x}) = 1$, if the inner product $\langle \vec{x}, \vec{y} \rangle \neq 0$, and 0, otherwise. NIPPE is known to imply identity-based revocation (IBR) [AL10], which itself is an important cryptographic primitive for efficiently and securely handling dynamic content distribution systems and group communication systems.

The first NIPPE scheme was proposed by Attrapadung and Libert [AL10] in prime order bilinear group setting with security in the co-selective model under the Decisional Linear (DLIN) and the Decisional Bilinear Diffie-Hellman (DBDH) assumptions. Their NIPPE scheme has 9 group elements in the ciphertext headers, i.e., except the group element masking the payload, while $O(\ell)$ group elements in the decryption keys and the public parameters, where ℓ is the length of the vectors. In subsequent works, Attrapadung et al. [ALDP11] and Yamada et al. [YAHK14] further reduced the NIPPE ciphertext header size to include only 2 group elements, however, the decryption keys and the public parameters continued to involve $O(\ell)$ group elements. These NIPPE constructions are also built in prime order bilinear groups and are proven secure, the former [ALDP11] in the co-selective model while the latter [YAHK14] in the selective model, under the Decisional Bilinear Diffie-Hellman Exponent (DBDHE) assumption parameterized by ℓ. The number of ciphertext header components achieved by these two NIPPE constructions is the smallest in the literature so far.

However, as demonstrated by Cheon [Che06], the q-type complexity assumptions such as DBDHE and thus the cryptosystems built on them are vulnerable to a serious attack. Specifically, Cheon developed an algorithm which recovers the secret involved in a q-type assumption in time inversely proportional to q. Later, Sakeme et al. [SHI+12] showed that the attack of Cheon can be a real threat to cryptosystems based on q-type assumptions by executing a successful experiment. Hence, it is clear that the parameters of q-type-assumption-based cryptographic constructions must scale with q in order to maintain a constant security level.

Adaptively secure NIPPE constructions with constant number of group elements in the ciphertext headers were proposed in the works by Okamoto and Takashima [OT15], and by Chen et al. [CGW15] in prime order bilinear groups under the DLIN and k-linear (k-LIN) assumptions respectively. The construction of [CGW15] meets the least number so far, i.e., 2 group elements in the ciphertext headers. However, decryption keys and public parameters consist of $O(\ell)$ group elements in each of the two constructions. In summary, to the best of our knowledge, currently there is no NIPPE scheme available in the literature that features constant number of group elements simultaneously in the ciphertext headers and in the decryption keys.

ABE: An even more expressive form of public-index PE is *attribute-based encryption* (ABE). The recent advances in cloud technology has triggered an emerging trend among individuals and organizations to outsource potentially sensitive private informations to external untrusted servers and later share various segments of the outsourced data with legitimate entities. ABE is an indispensable cryptographic tool for preserving data confidentiality in such cloud computing platforms. ABE comes in two flavors, namely, *key-policy* and *ciphertext-policy*. In a key-policy ABE system over an attribute universe $\mathbb{U}$, the index set $\mathbb{X}$ consists of all non-empty subsets of $\mathbb{U}$ and the index set $\mathbb{Y}$ is comprised of certain access structures over $\mathbb{U}$. A predicate $\mathscr{P}_{\mathscr{A}} : \mathbb{X} \to \{0,1\}$ associated with some access structure $\mathscr{A} \in \mathbb{Y}$ is defined for all attribute sets $\Gamma \in \mathbb{X}$ as $\mathscr{P}_{\mathscr{A}}(\Gamma) = 1$, if the access structure $\mathscr{A}$ accepts the attribute set Γ, and 0, otherwise. The ciphertext-policy variant interchanges the roles of attribute sets and access structures. In this work, we concentrate on key-policy ABE.

The notion of ABE was introduced by Sahai and Waters [SW05] for threshold access structures. Over time the class of access structures realizable by ABE systems has been gradually expanded by several researchers [GPSW06, OSW07, Wat11] culminating into the recent state of the art constructions which can now support access structures represented by arbitrary polynomial-size circuits [GGH+13, GVW15] and even Turing machines [GKP+13]. However, in view of the current progress in computing technology, it appears that the most expressive form of access structures supported by computationally practical

ABE systems are span programs. Besides the expressiveness of supported access structures, succinctness of ciphertext headers has been an important concern towards practicality of ABE schemes.

Emura et al. [EMN+09], Herranz et al. [HLR10], and Chen et al. [CCL+13] constructed ABE schemes with constant number of components in the ciphertext headers. However, the access structures supported by those constructions are very limited. Attrapadung et al. [ALDP11] were the first to develop a selectively secure key-policy ABE construction supporting non-monotone span programs in prime order bilinear groups featuring 3 group elements in the ciphertext headers based on the DBDHE assumption parameterized by ℓ, where ℓ is the maximum number of attributes per ciphertext. Later, Yamada et al. [YAHK14] designed another selectively secure key-policy ABE scheme for non-monotone span programs in prime order bilinear group setting that further reduced the number of ciphertext header components by 1 group element. As per our knowledge, this construction involves the least number of ciphertext header components among the computationally practical key-policy ABE systems currently available in the literature. However, the underlying complexity assumption of this construction is again DBDHE parameterized by ℓ. Attrapadung [Att14, Att15] subsequently built adaptively secure key-policy ABE schemes for monotone span programs with constant number of group elements in the ciphertext headers. The construction of [Att14] is developed in composite order bilinear groups and it has 6 group elements in the ciphertext headers, whereas, the scheme of [Att15] is constructed in prime order bilinear groups and its ciphertext headers include 18 group elements. These constructions are also based on certain q-type assumptions which are even stronger than the DBDHE assumption.

Clearly, the principal downside of all the four ABE constructions [ALDP11, YAHK14, Att14, Att15] is that they suffer from Cheon's attack [Che06] and require parameters that scale with the number of attributes per ciphertext or in the attribute universe for preserving a fixed security level. This bottleneck of using q-type complexity assumptions for building key-policy ABE systems with constant number of ciphertext header components was first mitigated by Chen and Wee [CW14], who designed a key-policy ABE scheme for monotone span programs in composite order bilinear group setting based on static assumptions featuring the least number (only 2) of group elements in the ciphertext headers among existing similar constructions. However, the static assumptions used in [CW14] are rather non-standard. More recently, Takashima [Tak14] constructed a semi-adaptively secure key-policy ABE scheme supporting non-monotone span programs in prime order bilinear groups under the well-studied static DLIN assumption featuring 17 group elements in the ciphertext headers. In all the ABE constructions discussed above, the number of group elements constituting the decryption keys and the public parameters are $O(m\ell)$ and $O(\ell)$ respectively, where m is the maximum number of rows of the matrix representing the span program.

Online-Offline MIFE: *Multi-input functional encryption* (MIFE), introduced by Goldwasser et al. [GGG+14] and subsequently studied in [GJN15, BKS15, BGJS15, BLR+15], is a generalization of FE to the setting of multi-input functions. In an MIFE scheme for a family $\mathfrak{F}_m$ of m-ary functions, a decryption key $\mathrm{SK}(\mathscr{F})$ corresponding to some function $\mathscr{F} \in \mathfrak{F}_m$ can be used to decrypt m ciphertexts $\mathrm{CT}^{(1)}(\mathscr{M}^{(1)}), \ldots, \mathrm{CT}^{(m)}(\mathscr{M}^{(m)})$, encrypting the messages $\mathscr{M}^{(1)}, \ldots, \mathscr{M}^{(m)}$ for the input slots $1, \ldots, m$ respectively, to retrieve $\mathscr{F}(\mathscr{M}^{(1)}, \ldots, \mathscr{M}^{(m)})$. In such systems, other than generating the public system parameters and master secret key, the setup authority also creates m encryption keys $\mathrm{ENK}^{(1)}, \ldots, \mathrm{ENK}^{(m)}$ which are necessary for encrypting messages for the input slots $1, \ldots, m$ respectively. MIFE has a wide range of practical applications such as running SQL (structured query language) queries on encrypted databases, non-interactive differentially private data release, delegation of expensive computations to external servers and many more.

The *bounded-norm multi-input inner product* functionality, which we consider in this paper, is an extension of the usual single-input inner product function and has been explicitly defined by Abdalla et al. [ARW16]. A multi-input inner product function $\mathscr{F}_{\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}}$ over $\mathbb{Z}_n$ is specified by an m-tuple of vectors $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$, where for each $j \in [m]$, $\vec{y}^{(j)}$ is a vector of length ℓ over $\mathbb{Z}_n$, for some $m, n, \ell \in \mathbb{N}$. The function $\mathscr{F}_{\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}}$ takes as input m vectors $\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}$, where for each $j \in [m]$, $\vec{x}^{(j)}$ is again a vector of length ℓ over $\mathbb{Z}_n$, and outputs the sum of inner product values $\sum_{j \in [m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle$. In the bounded norm setting, it is required that the norm of each component inner product $\langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle$ be bounded by some fixed $\mathsf{B} \in \mathbb{N}$. Inner product and hence its multi-input variant is an extremely useful functionality in the context of descriptive statistics, e.g., for computing the weighted mean of a collection of values. It also enables the computations of conjunctions, disjunctions, and polynomial expressions, as well as determination of exact thresholds.

In recent years, as computation is moving on to resource bounded and power constrained devices such as mobile phones, there has been a growing demand for *online-offline* cryptography. The basic idea of the online-offline model is to provision for an expensive preparation or offline phase, where the majority of computation is performed before the actual data become available. This is followed by an efficient online phase, which is executed when the data become known.

One vital limitation of FE is that the rich functionalities often come at the expense of a serious computational load compared to traditional encryption schemes. Specifically, the decryption key generation time depends on the complexity of the functions, while the encryption time scales with the length of the message and sometime even with the complexity of the function family. The situation is evidently more severe in the context of MIFE as multi-input functionalities have much larger complexity compared to the single input ones. In fact, an exacerbating issue is that the cost for operations may vary widely between each ciphertext and decryption key, thus forcing a system to provision for a load that matches a worst case scenario. In the field of single-input FE, online-offline versions have already been considered for IBE [GMC08, LZ09], ABE [HW14], and very recently for general purpose FE supporting arbitrary polynomial-size circuits in the bounded collusion setting [AR16]. However, MIFE is not yet investigated in the online-offline model.

Our Contributions: Our goal in this work is to develop NIPPE and ABE schemes with *optimal* parameters under *well-studied static* complexity assumptions. Specifically, we present the following two interesting results:

- Firstly, we construct an NIPPE scheme with 2 group elements in the ciphertexts and, at the same time, only 1 group element in the decryption keys. To the best of our knowledge, our NIPPE construction is the *first* to achieve *succinct* ciphertexts and decryption keys simultaneously. The proposed NIPPE scheme is built in composite order bilinear group setting and is proven selectively secure under the *standard Subgroup Decision* assumptions, which are *static* in nature. However, the number of group elements forming our NIPPE public parameters remains the same as earlier constructions, i.e., $O(\ell)$, where ℓ is the length of the vectors.
- Our second construction is a selectively secure key-policy ABE scheme supporting monotone span programs in composite order bilinear groups with only 2 group elements in the ciphertexts. The security is also proven under the *well-studied static* subgroup decision assumptions. Even when compared in terms of bit length with the only existing standard static assumption-based key-policy ABE scheme [Tak14] which achieves constant number of components in ciphertext headers in prime order bilinear group setting, our ABE ciphertext size turns out to be *much shorter*. For instance, it is known that [Fre10] for achieving an equivalent security level of a 160 bit prime order bilinear group, a composite order bilinear group requires the group order of 1024 bits. Thus, at this security level, the ciphertext bit-length of [Tak14] would be 17×160, whereas, that of ours would be 2×1024, which is roughly 25% smaller. Our ABE decryption keys and public parameters involve respectively $O(m\ell)$ and $O(\ell)$ group elements which are the same as all previously known key-policy ABE construction with short ciphertexts. Here, m and ℓ respectively denotes the maximum number of rows in the matrix representing the supported span programs and the maximum number of attributes per ciphertext.

We work in the *key encapsulation mechanism* setting, where the PE ciphertexts hide a symmetric session key that can be used to symmetrically encrypt the actual payload of arbitrary length. For proving security of our PE constructions, we employ the recent extended *Déjà Q* framework presented by Wee [Wee16]. The Déjà Q framework was originally proposed by Chase et al. [CM14]. It is a general framework for reducing various q-type complexity assumptions or their generalization, namely, the family of uber assumptions [BBG05, Boy08] to their static counter parts in composite order bilinear group setting making use of the classic dual system methodology [LW10, LOS+10]. A major efficiency drawback of the framework of [CM14] was its dependence on asymmetric bilinear groups. Wee [Wee16] has advanced the Déjà Q technique to overcome the necessity of working with asymmetric bilinear groups. Further, Wee [Wee16] has applied the enriched Déjà Q technique to design IBE and broadcast encryption (BE) schemes with essentially optimal parameters without relying on q-type assumptions. Wee has left the possibility of utilizing this elegant framework as an interesting research problem towards achieving compact parameters for more advanced cryptographic primitives with security under static assumptions. In this paper, we make progress in exploring the power of the extended Déjà Q framework by employing the technique in the context of highly expressive PE systems such as NIPPE and ABE. The technical contribution of our work thus lies in demonstrating a different approach towards attaining succinctness for NIPPE and

ABE schemes without requiring q-type assumptions by allowing a crucial looseness of q in the security reduction. Regarding computation cost, note that the decryption algorithm of both of our constructions require only 2 pairing operations which is the smallest among all similar works mentioned above.

A third contribution of this paper is to introduce the notion of *online-offline multi-input functional encryption* (OO-MIFE) and to develop the *first* OO-MIFE construction for the multi-input analog of the inner product functionality. As observed in [ARW16], a multi-input inner-product encryption (MIPE) scheme in the public key domain for m-ary inner product functions can be readily constructed by running m independent copies of a single-input inner-product encryption (IPE) scheme. It is trivial to see that a similar conversion also works for the online-offline setting in this domain. At the same time, the public key single-input IPE schemes available in the literature [ABDCP15, ALS15] are easily convertible to the online-offline mode. However, such a transformation does not work in the private key regime [ARW16].

Consequently, in this work, we focus on developing a private key online-offline multi-input inner product encryption (OO-MIPE) from efficient and standard cryptographic tools. Following the earlier online-offline FE constructions [GMC08, LZ09, HW14], we attempt to develop an OO-MIPE scheme which prepares intermediate ciphertext and decryption key components in the offline phase. When the data become available in the online phase, the corresponding ciphertext and decryption key components can be quickly constructed by incorporating simple *correction factors* to those intermediate quantities. We start by observing that the only existing private key MIPE construction due to Abdalla et al. [ARW16] has well suited algebraic structure for our strategy. We develop a correction technique that carefully utilizes the algebraic structures of the ciphertext and decryption key components of [ARW16] to push almost the entire computational bulk into the offline phase without significantly increasing the ciphertext or decryption key sizes over those of [ARW16]. Like [ARW16], our OO-MIPE construction is built in asymmetric bilinear groups of prime order. We reduce the selective security of our OO-MIPE construction directly to that of the MIPE scheme of [ARW16] which is selectively secure under the well-studied k-LIN assumption. As an advantage of our reduction, our OO-MIPE scheme would inherit any future improvements in the security guarantee offered by the MIPE construction of [ARW16].

Our online operations are quite fast. Our online decryption key generation algorithm costs only $m\ell(k+1)$ modular multiplications, where m, ℓ, and k are respectively the arity of the multi-input inner product function, the length of the vectors, and the parameter of the underlying complexity assumption. Thus, for instance, if we base the security of our construction on the Symmetric External Diffie-Hellman (SXDH) assumption, then $k = 1$, so that our online decryption key generation algorithm would involve just $2m\ell$ modular multiplications. Our online encryption algorithm is even more efficient as it incurs only modular additions which is the fastest operation in bilinear group setting. Regarding communication and storage requirements, both our offline and online decryption keys contain $m(k+1)$ additional $\mathbb{Z}_n$-component over those of the MIPE scheme of [ARW16], while our offline and online ciphertexts both include only ℓ additional $\mathbb{Z}_n$-components over those of the MIPE construction of [ARW16]. The increase in the ciphertext and decryption key sizes is reminiscent with those of earlier online-offline single-input FE constructions [GMC08, LZ09, HW14]. Moreover, the sizes of our public parameters and encryption keys are exactly the same as those of the MIPE construction of [ARW16].

Concurrent Work: In a concurrent and independent work, Chen et al. [CLR16] have developed a composite order NIPPE construction identical to the one presented in this paper and have derived its selective security under the Subgroup Decision assumption, utilizing the Déjà Q technique. However, our work extends to exploring the power of Déjà Q framework towards even more sophisticated form of PE, namely, ABE. Moreover, our third contribution, namely, the notion of OO-MIFE and its realization for the inner product functionality is not reported in the work of [CLR16].

2 Preliminaries

In this section, we will provide some cryptographic backgrounds which will be necessary in the sequel. Let $\lambda \in \mathbb{N}$ denotes the security parameter and 1^λ be its unary representation. A function negl is *negligible* if for every $c \in \mathbb{N}$, there exists $k \in \mathbb{N}$ such that for all $\lambda > k$, $|\mathsf{negl}(\lambda)| < \frac{1}{\lambda^c}$. Throughout this paper we will follow notations presented in Fig. 2.1.

Symbol	Explanation
$\aleph \xleftarrow{\$} \mathbb{S}$	$\aleph$ is uniformly sampled from a set $\mathbb{S}$.
$\varkappa \xleftarrow{\$} \mathcal{Z}(\Upsilon)$	$\varkappa$ is a random variable representing the output of a randomized algorithm $\mathcal{Z}$ on input Υ.
$\varkappa = \mathcal{Z}(\Upsilon)$	$\varkappa$ is the output of a deterministic algorithm $\mathcal{Z}$ on input Υ.
$[\mathfrak{fi}]$	$\{1, \ldots, \mathfrak{fi}\} \subset \mathbb{N}$, where $\mathfrak{fi} \in \mathbb{N}$.
$\vec{v}$	a vector $(v_1, \ldots, v_{\mathfrak{fi}}) \in \mathbb{Z}_n^{\mathfrak{fi}}$ of length $\mathfrak{fi}$, for some $n, \mathfrak{fi} \in \mathbb{N}$.
$\boldsymbol{B} = (B_{\iota,\iota'})_{\mathfrak{fi}\times\mathfrak{fi}'}$	a member of $\mathbb{Z}_n^{\mathfrak{fi}\times\mathfrak{fi}'}$, i.e., a matrix of size $\mathfrak{fi} \times \mathfrak{fi}'$ with entries $B_{\iota,\iota'} \in \mathbb{Z}_n$, for $\iota \in [\mathfrak{fi}], \iota' \in [\mathfrak{fi}']$, where $\mathfrak{fi}, \mathfrak{fi}' \in \mathbb{N}$.
$\boldsymbol{B}^\intercal$ $(\vec{v}^\intercal)$	the transpose of the matrix $\boldsymbol{B} \in \mathbb{Z}_n^{\mathfrak{fi}\times\mathfrak{fi}'}$ (the vector $\vec{v} \in \mathbb{Z}_n^{\mathfrak{fi}}$).
$\langle \vec{v}, \vec{w} \rangle$ or $\vec{v}\vec{w}^\intercal$	the inner product $\sum_{i\in[\mathfrak{fi}]} v_i w_i$ of vectors $\vec{v}, \vec{w} \in \mathbb{Z}_n^{\mathfrak{fi}}$.
$\boldsymbol{z} = g^{\vec{v}}$	a $\mathfrak{fi}$-length vector of group elements, $(z_1 = g^{v_1}, \ldots, z_{\mathfrak{fi}} = g^{v_{\mathfrak{fi}}}) \in \mathbb{G}^{\mathfrak{fi}}$, for some cyclic group $\mathbb{G}$ of order n and some $g \in \mathbb{G}$, where $\vec{v} \in \mathbb{Z}_n^{\mathfrak{fi}}$.
$g^{\delta\vec{v}} = (g^{\vec{v}})^\delta$	$(g^{\delta v_1}, \ldots, g^{\delta v_{\mathfrak{fi}}}) \in \mathbb{G}^{\mathfrak{fi}}$, where $\delta \in \mathbb{Z}_n$, $\vec{v} \in \mathbb{Z}_n^{\mathfrak{fi}}$, and $g \in \mathbb{G}$.
$g^{\vec{v}+\vec{w}} = g^{\vec{v}} g^{\vec{w}}$	$(g^{v_1+w_1}, \ldots, g^{v_{\mathfrak{fi}}+w_{\mathfrak{fi}}}) \in \mathbb{G}^{\mathfrak{fi}}$, where $\vec{v}, \vec{w} \in \mathbb{Z}_n^{\mathfrak{fi}}$ and $g \in \mathbb{G}$.
$\boldsymbol{Z} = g^{\boldsymbol{B}}$	a matrix $(g^{B_{\iota,\iota'}})_{\mathfrak{fi}\times\mathfrak{fi}'} \in \mathbb{G}^{\mathfrak{fi}\times\mathfrak{fi}'}$, where $\boldsymbol{B} = (B_{\iota,\iota'})_{\mathfrak{fi}\times\mathfrak{fi}'} \in \mathbb{Z}_n^{\mathfrak{fi}\times\mathfrak{fi}'}$ and $g \in \mathbb{G}$.
$(\boldsymbol{Z})^{\vec{v}} = (g^{\boldsymbol{B}})^{\vec{v}}$	$g^{\vec{v}\boldsymbol{B}^\intercal} \in \mathbb{G}^{\mathfrak{fi}}$, where $\vec{v} \in \mathbb{Z}_n^{\mathfrak{fi}'}$, $\boldsymbol{B} \in \mathbb{Z}_n^{\mathfrak{fi}\times\mathfrak{fi}'}$, and $g \in \mathbb{G}$.

Fig. 2.1. Notations

2.1 Bilinear Pairing Groups

Definition 2.1 (Bilinear Pairing Groups). A bilinear pairing group is a tuple $(n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ of $n \in \mathbb{N}$; cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order n; and a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

i) (*Bilinearity*) For all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and $\zeta_1, \zeta_2 \in \mathbb{Z}_n$, $e(g_1^{\zeta_1}, g_2^{\zeta_2}) = e(g_1, g_2)^{\zeta_1\zeta_2}$.

ii) (*Non-Degeneracy*) There exists $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2)$ has order n in $\mathbb{G}_T$.

A bilinear pairing group generation algorithm $\mathcal{G}$ takes as input the unary encoded security parameter λ and outputs the description of a bilinear pairing group $(n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with n of size $\Theta(\lambda)$. $\mathbb{G}_1$ and $\mathbb{G}_2$ are referred to as the *source groups*, while $\mathbb{G}_T$ is referred to as the *target group*. The group operations in the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, and the pairing e are assumed to be computable in polynomial time with respect to the security parameter λ.

A bilinear pairing group $(n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is said to be *symmetric* if $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ (say), otherwise, it is called *asymmetric*. On the other hand, in case n is a prime integer, the bilinear pairing group is said to be of *prime order*, else, it is of *composite order*. We assume that a bilinear pairing group generator $\mathcal{G}$ also takes as input the variables $\mathtt{type} \in \{\mathtt{symmetric}, \mathtt{asymmetric}\}$ and $\mathtt{ord} \in \{\mathtt{prime}, \mathtt{composite}\}$ in addition to the security parameter λ, and outputs the description of a bilinear pairing group of the specified kind.

Observe that given a bilinear pairing group $(n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, for any $q \in \mathbb{N}$, we can define a new bilinear pairing $E_q : \mathbb{G}_1^q \times \mathbb{G}_2^q \to \mathbb{G}_T$ as $E_q(\boldsymbol{a}, \boldsymbol{b}) = \prod_{\iota\in[q]} e(a_\iota, b_\iota)$, for all $\boldsymbol{a} \in \mathbb{G}_1^q, \boldsymbol{b} \in \mathbb{G}_2^q$. The bilinearity and non-degeneracy properties of the pairing E_q follows readily from those of the pairing e. Note that $E_q(g_1^{\vec{v}}, g_2^{\vec{w}}) = e(g_1, g_2)^{\langle\vec{v},\vec{w}\rangle}$, for any $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and any $\vec{v}, \vec{w} \in \mathbb{Z}_n^q$.

In this paper, we consider symmetric bilinear pairing groups of composite order and asymmetric ones of prime order. For symmetric composite order bilinear pairing groups $(n, \mathbb{G}, \mathbb{G}_T, e)$, we consider $n = p_1p_2p_3$, where p_1, p_2, p_3 are three distinct prime integers of size $\Theta(\lambda)$. We let $\mathbb{G}_{p_\iota}$ and $\mathbb{G}_{p_\iota p_{\iota'}}$ denote the subgroups of $\mathbb{G}$ of order p_ι and $p_\iota p_{\iota'}$ respectively, for $\iota, \iota' \in [3]$. Note that these subgroups are *orthogonal* to each other under the bilinear pairing e, i.e., if $\widetilde{g}, \widetilde{g}'$ are elements of two different subgroups out of $\{\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}\}$, then $e(\widetilde{g}, \widetilde{g}')$ is the identity element in $\mathbb{G}_T$. If $g, \hat{g}$, and $\check{g}$ are some elements of $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}$, and $\mathbb{G}_{p_3}$ respectively, then every element $\widetilde{g} \in \mathbb{G}$ can be expressed as $\widetilde{g} = g^\xi \hat{g}^{\hat{\xi}} \check{g}^{\check{\xi}}$, for some $\xi, \hat{\xi}, \check{\xi} \in \mathbb{Z}_n$. We will refer to g^ξ as the "$\mathbb{G}_{p_1}$ component of $\widetilde{g}$", and similarly others. Observe that for all $\iota \in [3]$, given some $\widetilde{g} \in \mathbb{G}_{p_\iota}$, one can efficiently sample uniformly from $\mathbb{G}_{p_\iota}$ by selecting $\widetilde{\xi} \xleftarrow{\$} \mathbb{Z}_n$ and computing $\widetilde{g}^{\widetilde{\xi}}$.

2.2 Complexity Assumptions

Assumption 2.1 (Subgroup Decision-I: SD-I [LW10]). The SD-I assumption is to distinguish between the distributions $\varpi_{\hat{\beta}} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, \Re_{\hat{\beta}})$, for $\hat{\beta} \in \{0,1\}$, where $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{symmetric}, \texttt{composite})$, $g \xleftarrow{\$} \mathbb{G}_{p_1}$, $\breve{g} \xleftarrow{\$} \mathbb{G}_{p_3}$, and

$$\Re_{\hat{\beta}} = \begin{cases} g^\sigma \xleftarrow{\$} \mathbb{G}_{p_1}, & \text{if } \hat{\beta} = 0 \\ g^\sigma \hat{g}^{\hat{\sigma}} \xleftarrow{\$} \mathbb{G}_{p_1p_2}, & \text{if } \hat{\beta} = 1 \end{cases}$$

with $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$ and $\sigma, \hat{\sigma} \xleftarrow{\$} \mathbb{Z}_n$. The SD-I assumption states that for any probabilistic polynomial-time algorithm $\mathcal{B}$, for any security parameter λ, the distinguishing advantage of $\mathcal{B}$ in the SD-I problem,

$$\mathsf{Adv}^{\mathsf{SD\text{-}I}}_{\mathcal{B}}(\lambda) = \left|\mathsf{Pr}[\mathcal{B}(1^\lambda, \varpi_0) = 1] - \mathsf{Pr}[\mathcal{B}(1^\lambda, \varpi_1) = 1]\right| \leq \mathsf{negl}(\lambda),$$

for some negligible function negl.

Assumption 2.2 (Subgroup Decision-II: SD-II [LW10]). The SD-II problem is to distinguish between the distributions $\varpi_{\hat{\beta}} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, v, s, \Re_{\hat{\beta}})$, for $\hat{\beta} \in \{0,1\}$, where $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{symmetric}, \texttt{composite})$, $g \xleftarrow{\$} \mathbb{G}_{p_1}$, $\breve{g} \xleftarrow{\$} \mathbb{G}_{p_3}$, $v = g^\sigma \hat{g}^{\hat{\sigma}} \xleftarrow{\$} \mathbb{G}_{p_1p_2}$, $s = \hat{g}^{\hat{\varphi}} \breve{g}^{\breve{\varphi}} \xleftarrow{\$} \mathbb{G}_{p_2p_3}$, and

$$\Re_{\hat{\beta}} = \begin{cases} g^\omega \breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}_{p_1p_3}, & \text{if } \hat{\beta} = 0 \\ g^\omega \hat{g}^{\hat{\omega}} \breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}, & \text{if } \hat{\beta} = 1 \end{cases}$$

with $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$ and $\sigma, \hat{\sigma}, \hat{\varphi}, \breve{\varphi}, \omega, \hat{\omega}, \breve{\omega} \xleftarrow{\$} \mathbb{Z}_n$. The SD-II states that for any probabilistic polynomial-time algorithm $\mathcal{B}$, for any security parameter λ, the distinguishing advantage of $\mathcal{B}$ in the SD-II problem,

$$\mathsf{Adv}^{\mathsf{SD\text{-}II}}_{\mathcal{B}}(\lambda) = \left|\mathsf{Pr}[\mathcal{B}(1^\lambda, \varpi_0) = 1] - \mathsf{Pr}[\mathcal{B}(1^\lambda, \varpi_1) = 1]\right| \leq \mathsf{negl}(\lambda),$$

for some negligible function negl.

Assumption 2.3 (k-Linear: k-LIN [EHK+13]). The k-LIN$^{(\iota)}$ problem, for fixed $\iota \in \{1,2\}$, is to distinguish between the distributions $\varpi^{(\iota)}_{\hat{\beta}} = ((n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, \boldsymbol{A}_\iota, \boldsymbol{\Re}^{(\iota)}_{\hat{\beta}})$, for $\hat{\beta} \in \{0,1\}$, where $(n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{asymmetric}, \texttt{prime})$, $g_1 \xleftarrow{\$} \mathbb{G}_1$, $g_2 \xleftarrow{\$} \mathbb{G}_2$, $\boldsymbol{A}_\iota = g_\iota^{\boldsymbol{A}}$, and

$$\boldsymbol{\Re}^{(\iota)}_{\hat{\beta}} = \begin{cases} g_\iota^{\vec{s}\boldsymbol{A}^\intercal}, & \text{if } \hat{\beta} = 0 \\ g_\iota^{\vec{v}}, & \text{if } \hat{\beta} = 1 \end{cases}$$

with $\boldsymbol{A} \xleftarrow{\$} \mathbb{D}_k$, $\vec{s} \xleftarrow{\$} \mathbb{Z}_n^k$, and $\vec{v} \xleftarrow{\$} \mathbb{Z}_n^{k+1}$, $\mathbb{D}_k$ being an efficiently samplable distribution over full-ranked matrices in $\mathbb{Z}_n^{(k+1)\times k}$. The k-LIN assumption states that for any probabilistic polynomial-time algorithm $\mathcal{B}$, for any security parameter λ, the distinguishing advantage of $\mathcal{B}$ in the k-LIN$^{(\iota)}$ problem for fixed $\iota \in \{1,2\}$,

$$\mathsf{Adv}^{k\text{-}\mathsf{LIN},\iota}_{\mathcal{B}}(\lambda) = \left|\mathsf{Pr}[\mathcal{B}(1^\lambda, \varpi^{(\iota)}_0) = 1] - \mathsf{Pr}[\mathcal{B}(1^\lambda, \varpi^{(\iota)}_1) = 1]\right| \leq \mathsf{negl}(\lambda),$$

for some negligible function negl.

2.3 Pairwise Independent Hash Families and the Left-Over Hash Lemma

Definition 2.2 (Min-Entropy). Let R be a random variable over some finite set $\mathbb{R}$. The min-entropy H_∞ of the random variable R is defined as $\mathsf{H}_\infty(R) = -\log(\max_{\varrho \in \mathbb{R}} \mathsf{Pr}[R = \varrho])$.

Definition 2.3 (Pairwise Independent Hash Families). Let $\mathbb{V}, \mathbb{W} \subset \{0,1\}^*$ be finite sets. A family $\mathbb{H}_2$ of hash functions $\mathcal{H} : \mathbb{V} \to \mathbb{W}$ is said to be pairwise independent if for any $v_1, v_2 \in \mathbb{V}$ such that $v_1 \neq v_2$ and for any $w_1, w_2 \in \mathbb{W}$, we have

$$\Pr_{\mathcal{H} \xleftarrow{\$} \mathbb{H}_2} [(\mathcal{H}(v_1) = w_1) \ \wedge \ (\mathcal{H}(v_2) = w_2)] = \frac{1}{(\sharp\mathbb{W})^2},$$

where $\sharp\mathbb{W}$ denotes the cardinality of the set $\mathbb{W}$.

The *left-over hash lemma*, first stated by Impagliazzo et al. [ILL89], is a classic instantiation of *strong randomness extractors* [Vad02], based on pairwise independent hash families. Randomness extractors are functions which take as input a non-uniform random source along with some uniform randomness, called the seed, and output nearly uniform bits. Randomness extractors are in general extremely useful objects in cryptography. Arguably, it is difficult to find "pure", i.e., uniform randomness in nature, but we may still hope to find "significant" randomness in the form of a random variable that is hard to predict, i.e., possessing reasonable amount of statistical entropy. If we can manage to get "just a little" pure randomness, extractors can provide us with some output which can be used as pure randomness in practice. We will use the left-over hash lemma for extracting uniformly random bits from some random variable in the proofs of our NIPPE and ABE constructions.

Lemma 2.1 (Left-Over Hash Lemma [ILL89]). *Let R be a random variable over some set $\mathbb{R}$ and let $\Im \in \mathbb{N}$. Let $\mathcal{H} : \mathbb{R} \rightarrow \{0,1\}^{\Im}$ be sampled from a pairwise independent hash family $\mathbb{H}_2$ uniformly and independently of R. If $\Im = \mathsf{H}_\infty(R) - 2\log\frac{1}{\epsilon} - \mathrm{O}(1)$, then $\Delta((\mathcal{H}(R), \mathcal{H}), (U_\Im, \mathcal{H})) \leq \frac{\epsilon}{2}$, where $U_\Im$ is the uniform random variable over $\{0,1\}^{\Im}$ and $\Delta(V, W) = \frac{1}{2}\sum_{\partial \in \mathbb{D}} |\mathsf{Pr}[V = \partial] - \mathsf{Pr}[W = \partial]|$ is a statistical distance between any two random variables V and W over some set $\mathbb{D}$.*

2.4 Overview of the Déjà Q Framework and its Extension

We now briefly explain the core techniques of the Déjà Q framework. This framework is an extension of the classic dual system methodology [LW10, LOS+10] that eliminates the use of q-type complexity assumptions for deriving security of bilinear-map-based cryptographic constructions. This was beyond the reach of earlier techniques in numerous settings. However, the original Déjà Q framework of Chase et al. [CM14] is not competent to handle advanced encryption systems such as PE, where certain secret exponents appear in both ciphertexts and decryption keys, i.e., on both arguments of the pairing. Wee [Wee16] has recently enriched the framework to overcome this bottleneck, employing several simple but interesting ideas.

The Déjà Q framework elegantly utilizes the dual system technique to reduce the security of a composite order analog of a cryptographic construction, originally developed in the prime order bilinear group setting with security under a q-type assumption, to the Subgroup Decision assumptions. To illustrate the basic ideas of the extended Déjà Q technique, let us assume that we want to apply this technique to prove the security of the composite order version of certain prime order PE construction secure under the q-DBDHE assumption. Suppressing many details pertaining to randomization and subgroups, let us assume that our composite order construction is built in the bilinear group $(n, \mathbb{G}, \mathbb{G}_T, e)$, where n is the product of two primes p_1 and p_2. Recall that any prime order PE construction that requires the q-DBDHE assumption for the security reduction, involves group elements of the form $\widetilde{u}_\iota = u^{\alpha^\iota}$, for $\iota \in [2q]$, where u is a generator of the group and α is a secret exponent. The group elements $\{\widetilde{u}_\iota\}_{\iota \in [2q] \setminus \{q+1\}}$ are made public through the public parameters or the decryption keys, while $\widetilde{u}_{q+1}$ is kept secret and the statistical entropy resulting from this group element is used to hide the message in the challenge ciphertext. When instantiating the PE scheme in the composite order bilinear group, the $\mathbb{G}_{p_1}$ subgroup is used for functionality, i.e., the composite order scheme involves the group elements $\widetilde{u}_\iota = u^{\alpha^\iota}$, for $\iota \in [2q]$, where $u \xleftarrow{\$} \mathbb{G}_{p_1}$ and $\alpha \xleftarrow{\$} \mathbb{Z}_n$.

The extended Déjà Q framework proceeds to reduce the security of the PE construction as follows: First, using the $\mathbb{G}_{p_1}/\mathbb{G}_{p_1p_2}$-Subgroup Decision (SD) assumption, which asserts that random elements of $\mathbb{G}_{p_1}$ and those of $\mathbb{G}_{p_1p_2}$ are computationally indistinguishable, $\widetilde{u}_\iota$'s are indistinguishably switched to $\widetilde{u}_\iota = u^{\alpha^\iota}\hat{g}^{\hat{\vartheta}_1\alpha^\iota}$, for $\iota \in [2q]$, where $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$ and $\hat{\vartheta}_1 \xleftarrow{\$} \mathbb{Z}_n$. Now, by the Chinese Remainer Theorem (CRT), it follows that $\alpha \mod p_1$ and $\alpha \mod p_2$ are independent random values. Hence, $\alpha \mod p_2$ can be replaced by $\hat{\alpha}_1 \mod p_2$ for a fresh $\hat{\alpha}_1 \xleftarrow{\$} \mathbb{Z}_n$ as long as no information about $\alpha \mod p_2$ is leaked through the public parameters, decryption keys, or challenge ciphertext.The framework carefully employs this fact to further transform $\widetilde{u}_\iota$'s to $\widetilde{u}_\iota = u^{\alpha^\iota}\hat{g}^{\hat{\vartheta}_1\hat{\alpha}_1^\iota}$, for $\iota \in [2q]$. Performing this two-step transition $2q$ times, the framework modifies $\widetilde{u}_\iota$'s to

$$\widetilde{u}_\iota = u^{\alpha^\iota} \xrightarrow{\mathsf{SD}} u^{\alpha^\iota}\hat{g}^{\hat{\vartheta}_1\alpha^\iota} \xrightarrow{\mathsf{CRT}} u^{\alpha^\iota}\hat{g}^{\hat{\vartheta}_1\hat{\alpha}_1^\iota} \xrightarrow{\mathsf{SD}} u^{\alpha^\iota}\hat{g}^{\hat{\vartheta}_2\alpha^\iota}\hat{g}^{\hat{\vartheta}_1\hat{\alpha}_1^\iota} \xrightarrow{\mathsf{CRT}} u^{\alpha^\iota}\hat{g}^{\sum_{j\in[2]}\hat{\vartheta}_j\hat{\alpha}_j^\iota} \xrightarrow{\mathsf{SD}} \ldots \xrightarrow{\mathsf{CRT}} u^{\alpha^\iota}\hat{g}^{\sum_{j\in[2q]}\hat{\vartheta}_j\hat{\alpha}_j^\iota},$$

where $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_{2q}, \hat{\alpha}_1, \ldots, \hat{\alpha}_{2q} \xleftarrow{\$} \mathbb{Z}_n$.

At this point, the framework invokes the following lemma, which is proven in [Wee16] and in [CM14] in a more general form.

Lemma 2.2 (Core Lemma of the Déjà Q Framework [Wee16, CM14]). *Consider any prime integer p and any positive integer $\pounds$. For any $\upsilon_1, \ldots, \upsilon_{\pounds}, \delta_1, \ldots, \delta_{\pounds} \in \mathbb{Z}_p$, define the function $\mathcal{F}^{(\pounds)}_{\upsilon_1,\ldots,\upsilon_{\pounds},\delta_1,\ldots,\delta_{\pounds}} : [\pounds] \to \mathbb{Z}_p$ as $\mathcal{F}^{(\pounds)}_{\upsilon_1,\ldots,\upsilon_{\pounds},\delta_1,\ldots,\delta_{\pounds}}(\chi) = \sum_{j\in[\pounds]} \upsilon_j \delta_j^{\chi}$. Then, for any (possibly computationally unbounded) algorithm $\mathcal{B}$,*

$$\left| \Pr_{\{\upsilon_j,\delta_j\}_{j\in[\pounds]} \xleftarrow{\$} \mathbb{Z}_p} \left[\mathcal{B}(1^{\pounds}, \{\mathcal{F}^{(\pounds)}_{\upsilon_1,\ldots,\upsilon_{\pounds},\delta_1,\ldots,\delta_{\pounds}}(\chi)\}_{\chi\in[\pounds]}) = 1\right] - \Pr\left[\mathcal{B}(1^{\pounds}, \{\mathcal{R}^{(\pounds)}(\chi)\}_{\chi\in[\pounds]}) = 1\right] \right| \leq \mathrm{O}\big(\frac{\pounds^2}{p}\big),$$

where $\mathcal{R}^{(\pounds)} : [\pounds] \to \mathbb{Z}_p$ is a truly random function.

The proof of Lemma 2.2 primarily relies on the fact that the Vandermonde matrix $\begin{pmatrix} \delta_1 & \delta_2 & \ldots & \delta_{\pounds} \\ \delta_1^2 & \delta_2^2 & \ldots & \delta_{\pounds}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_1^{\pounds} & \delta_2^{\pounds} & \ldots & \delta_{\pounds}^{\pounds} \end{pmatrix}$ is invertable as long as $\delta_1, \ldots, \delta_{\pounds}$ are distinct, which happens with overwhelming probability over $\delta_1, \ldots, \delta_{\pounds} \xleftarrow{\$} \mathbb{Z}_p$.

Going back to our example, since $\hat{\alpha}_1 \mod p_2, \ldots, \hat{\alpha}_{2q} \mod p_2$ are distinct with overwhelming probability over the choice of $\hat{\alpha}_1, \ldots, \hat{\alpha}_{2q} \xleftarrow{\$} \mathbb{Z}_n$, invoking Lemma 2.2, the reduction writes $\widetilde{u}_\iota = u^{\alpha^\iota} \hat{g}^{\mathcal{R}^{(2q)}(\iota)}$, for $\iota \in [2q]$, where $\mathcal{R}^{(2q)} : [2q] \to \mathbb{Z}_{p_2}$ is a truly random function. Thus, each $\widetilde{u}_\iota$ now has uniformly and independently random $\mathbb{G}_{p_2}$-component. Therefore, if only $\{\widetilde{u}_\iota\}_{\iota\in[2q]\setminus\{q+1\}}$ are disclosed through the public parameters or the decryption keys, the $\mathbb{G}_{p_2}$-component of $\widetilde{u}_{q+1}$ remains absolutely hidden. In order to utilize the statistical entropy arising out of $\widetilde{u}_{q+1}$ for hiding the message in the challenge ciphertext, the framework also introduces additional $\mathbb{G}_{p_2}$-component in the challenge ciphertext during the course of the reduction that would interact with the random $\mathbb{G}_{p_2}$-component of $\widetilde{u}_{q+1}$ through the pairing to generate the entropy. Finally, the framework further amplifies this entropy to completely hide the message in the challenge ciphertext, using a strong randomness extractor [Vad02].

3 Our Non-Zero Inner-Product-Predicate Encryption Scheme

In this section, we present the first ever non-zero inner-product predicate encryption (NIPPE) construction featuring constant number of components in both the ciphertexts and the decryption keys. Our NIPPE scheme is built in composite order bilinear groups and its selective security is proven under the well-studied static Subgroup Decision assumptions, using the extended Déjà Q framework.

3.1 Notion

Definition 3.1 (Non-Zero Inner-Product-Predicate Encryption: NIPPE [AL10, YAHK14]). A (public-index) non-zero inner-product-predicate encryption (NIPPE) for vectors in $\mathbb{Z}_n^\ell$, for some $n, \ell \in \mathbb{N}$, consists of the following polynomial-time algorithms:

- $\mathsf{NIPPE.Setup}(1^\lambda, \ell) \to (\mathrm{MPK}, \mathrm{MSK})$: The setup authority takes as input the unary encoded security parameter 1^λ together with the length ℓ of vectors. It publishes the public parameters MPK while keeps the master secret key MSK to itself.
- $\mathsf{NIPPE.KeyGen}(\mathrm{MPK}, \mathrm{MSK}, \vec{y}) \to \mathrm{SK}(\vec{y})$: On input the public parameters MPK, the master secret key MSK, and a vector $\vec{y} \in \mathbb{Z}_n^\ell$, the setup authority provides a decryption key $\mathrm{SK}(\vec{y})$ (which includes $\vec{y}$ in the clear) to a legitimate decrypter.
- $\mathsf{NIPPE.Encrypt}(\mathrm{MPK}, \vec{x}) \to (\mathrm{CT}(\vec{x}), \mathrm{EK})$: Taking as input the public parameters MPK along with a vector $\vec{x} \in \mathbb{Z}_n^\ell$, an encrypter creates a ciphertext $\mathrm{CT}(\vec{x})$ (which includes the vector $\vec{x}$ in the clear) and a session key EK.
- $\mathsf{NIPPE.Decrypt}(\mathrm{MPK}, \mathrm{SK}(\vec{y}), \mathrm{CT}(\vec{x})) \to \mathrm{EK}'$ or $\perp$: A decrypter takes as input the public parameters MPK, a decryption key $\mathrm{SK}(\vec{y})$ corresponding to some vector $\vec{y} \in \mathbb{Z}_n^\ell$, and a ciphertext $\mathrm{CT}(\vec{x})$ corresponding to some vector $\vec{x} \in \mathbb{Z}_n^\ell$. It outputs either a session key EK' or the empty string $\perp$ indicating failure.

The algorithms NIPPE.Setup, NIPPE.KeyGen, and NIPPE.Encrypt are randomized, while the algorithm NIPPE.Decrypt is deterministic.

■ **Correctness**: An NIPPE scheme is correct if for any security parameter $\lambda \in \mathbb{N}$, any $n, \ell \in \mathbb{N}$, and any $\vec{x}, \vec{y} \in \mathbb{Z}_n^\ell$ with $\langle \vec{x}, \vec{y} \rangle \neq 0$, we have

$$\mathsf{Pr}\Big[(\text{MPK}, \text{MSK}) \xleftarrow{\$} \mathsf{NIPPE.Setup}(1^\lambda, \ell);\ \text{SK}(\vec{y}) \xleftarrow{\$} \mathsf{NIPPE.KeyGen}(\text{MPK}, \text{MSK}, \vec{y});$$
$$(\text{CT}(\vec{x}), \text{EK}) \xleftarrow{\$} \mathsf{NIPPE.Encrypt}(\text{MPK}, \vec{x})\ :\ \mathsf{NIPPE.Decrypt}(\text{MPK}, \text{SK}(\vec{y}), \text{CT}(\vec{x})) = \text{EK}\Big] = 1.$$

■ **Security**: The selective indistinguishability-based (SEL-IND) security notion for an NIPPE scheme is formalized in terms of the following experiment involving a probabilistic polynomial-time adversary $\mathcal{A}$ and a probabilistic polynomial-time challenger $\mathcal{B}$:

- $\mathcal{A}$ declares a challenge vector $\vec{x}^* \in \mathbb{Z}_n^\ell$ to $\mathcal{B}$.
- $\mathcal{B}$ generates $(\text{MPK}, \text{MSK}) \xleftarrow{\$} \mathsf{NIPPE.Setup}(1^\lambda, \ell)$ and provides MPK to $\mathcal{A}$.
- $\mathcal{B}$ creates $(\text{CT}^*, \text{EK}_0^*) \xleftarrow{\$} \mathsf{NIPPE.Encrypt}(\text{MPK}, \vec{x}^*)$ and selects $\text{EK}_1^* \xleftarrow{\$} \{0,1\}^\lambda$. Next, $\mathcal{B}$ chooses a bit $\beta \in \{0,1\}$ and gives $(\text{CT}^*, \text{EK}_\beta^*)$ to $\mathcal{A}$.
- $\mathcal{A}$ may adaptively make any polynomial number of decryption key queries. In response to a decryption key query of $\mathcal{A}$ corresponding to some vector $\vec{y} \in \mathbb{Z}_n^\ell$ subject to the restriction that $\langle \vec{x}^*, \vec{y} \rangle = 0$, $\mathcal{B}$ forms $\text{SK}(\vec{y}) \xleftarrow{\$} \mathsf{NIPPE.KeyGen}(\text{MPK}, \text{MSK}, \vec{y})$ and hands $\text{SK}(\vec{y})$ to $\mathcal{A}$.
- $\mathcal{A}$ eventually outputs a guess bit $\beta' \in \{0,1\}$.

An NIPPE is said to be SEL-IND secure if for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{NIPPE,SEL\text{-}IND}}(\lambda) = \Big|\mathsf{Pr}[\beta = \beta'] - 1/2\Big| \leq \mathsf{negl}(\lambda),$$

for some negligible function negl.

3.2 Construction

$\mathsf{NIPPE.Setup}(1^\lambda, \ell) \to (\text{MPK}, \text{MSK})$: The setup authority takes as input the unary encoded security parameter 1^λ together with the length ℓ of vectors, and proceed as follows:

1. It first generates $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{symmetric}, \texttt{composite})$.
2. Next, it selects $\alpha, \gamma \xleftarrow{\$} \mathbb{Z}_n$, $g, u \xleftarrow{\$} \mathbb{G}_{p_1}$, and $\breve{r}_1, \ldots, \breve{r}_{2\ell} \xleftarrow{\$} \mathbb{G}_{p_3}$.
3. Then, it computes $h_0 = g^\gamma$, $h_\iota = g^{\alpha^\iota}$, for $\iota \in [\ell]$, $\widetilde{u}_\iota = u^{\alpha^\iota}\breve{r}_\iota$, for $\iota \in [2\ell]$, and $e(g, \widetilde{u}_{\ell+1})$.
4. After that, it samples $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ from a pairwise independent hash family $\mathbb{H}_2$.
5. It publishes the public parameters $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, h_0, \{h_\iota\}_{\iota \in [\ell]}, \{\widetilde{u}_\iota\}_{\iota \in [2\ell] \setminus \{\ell+1\}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$, while keeps the master secret key $\text{MSK} = (p_1, p_2, p_3, \alpha, \gamma, u)$.

$\mathsf{NIPPE.KeyGen}(\text{MPK}, \text{MSK}, \vec{y}) \to \text{SK}(\vec{y})$: On input the public parameters $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, h_0, \{h_\iota\}_{\iota \in [\ell]}, \{\widetilde{u}_\iota\}_{\iota \in [2\ell] \setminus \{\ell+1\}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$, the master secret key $\text{MSK} = (p_1, p_2, p_3, \alpha, \gamma, u)$, and a vector $\vec{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}_n^\ell$, the setup authority executes the following steps:

1. It picks $\breve{r}_{\vec{y}} \xleftarrow{\$} \mathbb{G}_{p_3}$.
2. It computes $k_{\vec{y}} = u^{\gamma \sum\limits_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}} \breve{r}_{\vec{y}}$.
3. It provides the decryption key $\text{SK}(\vec{y}) = (\vec{y}, k_{\vec{y}})$ to a legitimate decrypter.

$\mathsf{NIPPE.Encrypt}(\text{MPK}, \vec{x}) \to (\text{CT}(\vec{x}), \text{EK})$: Taking as input the public parameters $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, h_0, \{h_\iota\}_{\iota \in [\ell]}, \{\widetilde{u}_\iota\}_{\iota \in [2\ell] \setminus \{\ell+1\}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$ along with a vector $\vec{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_n^\ell$, an encrypter performs the following steps:

1. It selects $\theta \xleftarrow{\$} \mathbb{Z}_n$.
2. It computes $c_1 = g^\theta$, $c_2 = (h_0 \prod\limits_{\iota \in [\ell]} h_\iota^{x_\iota})^\theta = g^{(\gamma + \sum\limits_{\iota \in [\ell]} x_\iota \alpha^\iota)\theta}$, and $T = e(g, \widetilde{u}_{\ell+1})^\theta$.
3. It outputs the ciphertext $\text{CT}(\vec{x}) = (\vec{x}, c_1, c_2)$ and the session key $\text{EK} = \mathcal{H}(T)$.

$\mathsf{NIPPE.Decrypt}(\text{MPK}, \text{SK}(\vec{y}), \text{CT}(\vec{x})) \to \text{EK}'$ or $\perp$: A decrypter takes as input the public parameters $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, h_0, \{h_\iota\}_{\iota \in [\ell]}, \{\widetilde{u}_\iota\}_{\iota \in [2\ell] \setminus \{\ell+1\}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$, a decryption key $\text{SK}(\vec{y}) = (\vec{y}, k_{\vec{y}})$, and a ciphertext $\text{CT}(\vec{x}) = (\vec{x}, c_1, c_2)$. If $\langle \vec{x}, \vec{y} \rangle = 0$, then it outputs $\perp$. Otherwise, it proceeds as follows:

1. It computes $T' = \left[\dfrac{e\Big(c_2, \prod\limits_{\iota' \in [\ell]} \widetilde{u}_{\ell+1-\iota'}^{y_{\iota'}}\Big)}{e\Big(c_1, k_{\vec{y}} \prod\limits_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} \widetilde{u}_{\ell+1+\iota-\iota'}^{x_\iota y_{\iota'}}\Big)}\right]^{\frac{1}{\langle \vec{x}, \vec{y} \rangle}}$.
2. It retrieves the session key as $\text{EK}' = \mathcal{H}(T')$.

■ **Correctness**

The correctness of the proposed NIPPE scheme can be verified as follows: Observe that for any ciphertext $\text{CT}(\vec{x}) = (\vec{x}, c_1 = g^\theta, c_2 = g^{(\gamma + \sum\limits_{\iota \in [\ell]} x_\iota \alpha^\iota)\theta})$ corresponding to some vector $\vec{x} \in \mathbb{Z}_n^\ell$ and any decryption key $\text{SK}(\vec{y}) = (\vec{y}, k_{\vec{y}} = u^{\gamma \sum\limits_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}} \breve{r}_{\vec{y}})$ corresponding to some vector $\vec{y} \in \mathbb{Z}_n^\ell$, we have

$$
\begin{aligned}
T' &= \left[\frac{e\Big(c_2 \prod\limits_{\iota' \in [\ell]} \widetilde{u}_{\ell+1-\iota'}^{y_{\iota'}}\Big)}{e\Big(c_1, k_{\vec{y}} \prod\limits_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} \widetilde{u}_{\ell+1+\iota-\iota'}^{x_\iota y_{\iota'}}\Big)}\right]^{\frac{1}{\langle \vec{x}, \vec{y} \rangle}} \\
&= \left[\frac{e\Big(g^{(\gamma + \sum\limits_{\iota \in [\ell]} x_\iota \alpha^\iota)\theta}, u^{\sum\limits_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}}\Big) e\Big(g^{(\gamma + \sum\limits_{\iota \in [\ell]} x_\iota \alpha^\iota)\theta}, \prod\limits_{\iota' \in [\ell]} \breve{r}_{\ell+1-\iota'}^{y_{\iota'}}\Big)}{e\Big(g^\theta, u^{\gamma \sum\limits_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}} u^{\sum\limits_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} x_\iota y_{\iota'} \alpha^{\ell+1+\iota-\iota'}}\Big) e\Big(g^\theta, \breve{r}_{\vec{y}} \prod\limits_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} \breve{r}_{\ell+1+\iota-\iota'}^{x_\iota y_{\iota'}}\Big)}\right]^{\frac{1}{\langle \vec{x}, \vec{y} \rangle}} \\
&= \left[\frac{e\Big(g^{(\gamma + \sum\limits_{\iota \in [\ell]} x_\iota \alpha^\iota)\theta}, u^{\sum\limits_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}}\Big)}{e\Big(g^\theta, u^{\gamma \sum\limits_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}} u^{\sum\limits_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} x_\iota y_{\iota'} \alpha^{\ell+1+\iota-\iota'}}\Big)}\right]^{\frac{1}{\langle \vec{x}, \vec{y} \rangle}} \\
&\quad \text{(by orthogonality of the subgroups of } \mathbb{G} \text{ with respect to } e) \\
&= \left[\frac{e(g,u)^{\theta\gamma \sum\limits_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}} e(g,u)^{\theta \alpha^{\ell+1} \langle \vec{x}, \vec{y} \rangle} e(g,u)^{\theta \sum\limits_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} x_\iota y_{\iota'} \alpha^{\ell+1+\iota-\iota'}}}{e(g,u)^{\theta\gamma \sum\limits_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}} e(g,u)^{\theta \sum\limits_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} x_\iota y_{\iota'} \alpha^{\ell+1+\iota-\iota'}}}\right]^{\frac{1}{\langle \vec{x}, \vec{y} \rangle}} \\
&= e(g,u)^{\theta \alpha^{\ell+1}} = e(g, \widetilde{u}_{\ell+1})^\theta = T.
\end{aligned}
$$

Therefore, $\text{EK}' = \mathcal{H}(T') = \mathcal{H}(T) = \text{EK}$.

3.3 Security Analysis

Theorem 3.1 (Security of Our NIPPE Scheme). *The* NIPPE *construction proposed in Section 3.2 is* SEL-IND *secure, as per the security model described in Section 3.1, under the* SD-I *and* SD-II *assumptions.*

Proof. In order to prove Theorem 3.1, we consider a sequence of hybrid experiments. The first hybrid corresponds to the real SEL-IND security experiment for NIPPE, described in Section 3.1, while the final hybrid corresponds to one in which the adversary has no advantage. The sequence of hybrid experiments follows:

Sequence of Hybrid Experiments

Hyb_0: This experiment corresponds to the real SEL-IND security experiment of Section 3.1. More precisely, this experiment proceeds as follows:

- The adversary $\mathcal{A}$ submits a challenge vector $\vec{x}^*$ to the challenger $\mathcal{B}$.

- The challenger $\mathcal{B}$ creates $\Big(\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, h_0 = g^{\gamma}, \{h_\iota = g^{\alpha^\iota}\}_{\iota \in [\ell]}, \{\widetilde{u}_\iota = u^{\alpha^\iota} \breve{r}_\iota\}_{\iota \in [2\ell] \setminus \{\ell+1\}}, e(g, \widetilde{u}_{\ell+1} = u^{\alpha^{\ell+1}} \breve{r}_{\ell+1}), \mathcal{H}), \text{MSK} = (p_1, p_2, p_3, \alpha, \gamma, u)\Big) \xleftarrow{\$} \mathsf{NIPPE.Setup}(1^\lambda, \ell)$ and provides MPK to $\mathcal{A}$.
- To frame the challenge, $\mathcal{B}$ forms $\Big(\text{CT}^* = (\vec{x}^*, c_1^* = g^\theta, c_2^* = (h_0 \prod_{\iota \in [\ell]} h_\iota^{x_\iota^*})^\theta = g^{(\gamma + \sum_{\iota \in [\ell]} x_\iota^* \alpha^\iota)\theta}), \text{EK}_0^* = \mathcal{H}(e(g, \widetilde{u}_{\ell+1})^\theta)\Big) \xleftarrow{\$} \mathsf{NIPPE.Encrypt}(\text{MPK}, \vec{x}^*)$ and picks $\text{EK}_1^* \xleftarrow{\$} \{0,1\}^\lambda$. Next, $\mathcal{B}$ selects a bit $\beta \xleftarrow{\$} \{0,1\}$ and gives $(\text{CT}^*, \text{EK}_\beta^*)$ to $\mathcal{A}$.
- In response to a decryption key query of $\mathcal{A}$ corresponding to some vector $\vec{y}$ subject to the restriction that $\langle \vec{x}^*, \vec{y} \rangle = 0$, $\mathcal{B}$ creates $\text{SK}(\vec{y}) = (\vec{y}, k_{\vec{y}} = u^{\gamma \sum_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}} \breve{r}_{\vec{y}}) \xleftarrow{\$} \mathsf{NIPPE.KeyGen}(\text{MPK}, \text{MSK}, \vec{y})$ and hands $\text{SK}(\vec{y})$ to $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs a guess bit $\beta' \in \{0,1\}$.

Hyb_1: This experiment is analogous to Hyb_0 with the exception that while setting up the NIPPE system, $\mathcal{B}$ selects $\breve{\gamma} \xleftarrow{\$} \mathbb{Z}_n$ and sets $\gamma = \breve{\gamma} - \sum_{\iota \in [\ell]} x_\iota^* \alpha^\iota$, where $\vec{x}^*$ is the challenge vector declared by $\mathcal{A}$. More formally, this experiment proceeds as follows:

- As earlier $\mathcal{A}$ submits a challenge vector $\vec{x}^*$ to $\mathcal{B}$.
- For setting up the NIPPE system, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first generates $(n = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{symmetric}, \texttt{composite})$.
  2. Next, $\mathcal{B}$ picks $\alpha, \breve{\gamma} \xleftarrow{\$} \mathbb{Z}_n$, $g, u \xleftarrow{\$} \mathbb{G}_{p_1}$, and $\breve{r}_1, \ldots, \breve{r}_{2\ell} \xleftarrow{\$} \mathbb{G}_{p_3}$.
  3. Then, $\mathcal{B}$ sets $\gamma = \breve{\gamma} - \sum_{\iota \in [\ell]} x_\iota^* \alpha^\iota$ and computes $h_0 = g^{\breve{\gamma} - \sum_{\iota \in [\ell]} x_\iota^* \alpha^\iota} = g^\gamma$, $h_\iota = g^{\alpha^\iota}$, for $\iota \in [\ell]$, $\widetilde{u}_\iota = u^{\alpha^\iota} \breve{r}_\iota$, for $\iota \in [2\ell]$, together with $e(g, \widetilde{u}_{\ell+1})$.
  4. $\mathcal{B}$ also uniformly samples $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ from a pairwise independent hash family $\mathbb{H}_2$.
  5. $\mathcal{B}$ gives $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, h_0, \{h_\iota\}_{\iota \in [\ell]}, \{\widetilde{u}_\iota\}_{\iota \in [2\ell] \setminus \{\ell+1\}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$ to $\mathcal{A}$.
- To frame the challenge, $\mathcal{B}$ performs the following steps:
  1. $\mathcal{B}$ chooses $\theta \xleftarrow{\$} \mathbb{Z}_n$.
  2. Then, $\mathcal{B}$ computes $c_1^* = g^\theta$, $c_2^* = (g^\theta)^{\breve{\gamma}} = g^{(\gamma + \sum_{\iota \in [\ell]} x^* \alpha^\iota)\theta}$, together with $\text{EK}_0^* = \mathcal{H}(e(g^\theta, \widetilde{u}_{\ell+1}))$, and uniformly draws $\text{EK}_1^* \xleftarrow{\$} \{0,1\}^\lambda$.
  3. Next, $\mathcal{B}$ selects a bit $\beta \xleftarrow{\$} \{0,1\}$ and gives $(\text{CT}^* = (\vec{x}^*, c_1^*, c_2^*), \text{EK}_\beta^*)$ to $\mathcal{A}$.
- In response to a decryption key query of $\mathcal{A}$ corresponding to some vector $\vec{y}$ subject to the restriction that $\langle \vec{x}^*, \vec{y} \rangle = 0$, $\mathcal{B}$ executes the following steps:
  1. $\mathcal{B}$ first selects $\breve{r}'_{\vec{y}} \xleftarrow{\$} \mathbb{G}_{p_3}$.
  2. Next, $\mathcal{B}$ computes

$$\begin{aligned} K_{\vec{y}} &= \Big(\prod_{\iota' \in [\ell]} \widetilde{u}_{\ell+1-\iota'}^{y_{\iota'}}\Big)^{\breve{\gamma}} \Big(\prod_{\substack{\iota, \iota' \in [\ell] \\ \iota \neq \iota'}} \widetilde{u}_{\ell+1+\iota-\iota'}^{-x_\iota^* y_{\iota'}}\Big) \breve{r}'_{\vec{y}} \\ &= u^{\breve{\gamma} \sum_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'} - \sum_{\substack{\iota, \iota' \in [\ell] \\ \iota \neq \iota'}} x_\iota^* y_{\iota'} \alpha^{\ell+1+\iota-\iota'}} \breve{r}_{\vec{y}} \\ &= u^{\gamma \sum_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'} - \sum_{\iota, \iota' \in [\ell]} x_\iota^* y_{\iota'} \alpha^{\ell+1+\iota-\iota'}} \breve{r}_{\vec{y}} \\ &= u^{\gamma \sum_{\iota' \in [\ell]} y_{\iota'} \alpha^{\ell+1-\iota'}} \breve{r}_{\vec{y}}, \end{aligned}$$

since $\langle \vec{x}^*, \vec{y} \rangle = 0$. Here, $\breve{r}_{\vec{y}} = \Big(\prod_{\iota' \in [\ell]} \breve{r}_{\ell+1-\iota'}^{y_{\iota'}}\Big)^{\breve{\gamma}} \Big(\prod_{\substack{\iota, \iota' \in [\ell] \\ \iota \neq \iota'}} \breve{r}_{\ell+1+\iota-\iota'}^{-x_\iota^* y_{\iota'}}\Big) \breve{r}'_{\vec{y}}$ is uniformly distributed in $\mathbb{G}_{p_3}$ as $\breve{r}'_{\vec{y}}$ is so.

3. $\mathcal{B}$ hands the decryption key $\text{SK}(\vec{y}) = (\vec{y}, k_{\vec{y}})$ to $\mathcal{A}$.

- At the end, $\mathcal{A}$ outputs a guess bit $\beta' \in \{0,1\}$.

Hyb_2: This experiment is similar to Hyb_1 with the only exception that while creating the challenge, $\mathcal{B}$ selects $w \xleftarrow{\$} \mathbb{G}_{p_1p_2}$ and sets $\underline{c_1^* = w}$, $\underline{c_2^* = w^{\check{\gamma}}}$, and $\underline{\text{EK}_0^* = \mathcal{H}(e(w, \widetilde{u}_{\ell+1}))}$.

Hyb_3: This experiment is the same as Hyb_2 with the exception that while setting up the NIPPE system, $\mathcal{B}$ sets $\underline{\widetilde{u}_\iota = u^{\alpha^\iota} \hat{g}^{\sum_{j\in[2\ell]} \hat{\vartheta}_j \hat{\alpha}_j^\iota} \check{r}_\iota}$, for $\iota \in [2\ell]$, where $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_{2\ell}, \hat{\alpha}_1, \ldots, \hat{\alpha}_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$, $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$. This in turn affects the distributions of MPK, EK_0^*, and all the $\text{SK}(\vec{y})$'s provided to $\mathcal{A}$.

Hyb_4: In this experiment at the time of setting up the NIPPE system, $\mathcal{B}$ sets $\underline{\widetilde{u}_\iota = u^{\alpha^\iota} \hat{g}^{\mathcal{R}^{(2\ell)}(\iota)} \check{r}_\iota}$, for $\iota \in [2\ell]$, where $\mathcal{R}^{(2\ell)} : [2\ell] \to \mathbb{Z}_{p_2}$ is a truly random function. This further changes the distributions of MPK, EK_0^*, and all the $\text{SK}(\vec{y})$'s given to $\mathcal{A}$. The rest of the experiment proceeds identically to Hyb_3.

Hyb_5: This experiment is analogous to Hyb_4 with the only exception that while framing the challenge, $\mathcal{B}$ selects $\underline{\text{EK}_0^* \xleftarrow{\$} \{0,1\}^\lambda}$.

Analysis

Let $\mathsf{Adv}_{\mathcal{A}}^{(i)}(\lambda)$ be the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the challenge bit, in Hyb_i, for $i \in \{0, \ldots, 5\}$. By definition, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{NIPPE,SEL\text{-}IND}}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda)$. Also, note that in Hyb_5, both EK_0^* and EK_1^* are uniformly distributed over $\{0,1\}^\lambda$. Therefore, the view of $\mathcal{A}$ in Hyb_5 is statistically independent of the challenge bit $\beta \xleftarrow{\$} \{0,1\}$ selected by $\mathcal{B}$. Hence, $\mathsf{Adv}_{\mathcal{A}}^{(5)}(\lambda) = 0$. Moreover, it readily follows that the distributions of MPK, CT^*, EK_0^*, and all the $\text{SK}(\vec{y})$'s provided to $\mathcal{A}$ in Hyb_0 and those in Hyb_1 are identical. Thus, the view of $\mathcal{A}$ in Hyb_0 and that in Hyb_1 are also the same. Therefore, $\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda)$. Hence, we have

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{NIPPE,SEL\text{-}IND}}(\lambda) \leq \sum_{i\in[4]} \left|\mathsf{Adv}_{\mathcal{A}}^{(i)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(i+1)}(\lambda)\right|. \tag{3.1}$$

Lemmas 3.1–3.4 will show that the RHS of Eq. (3.1) is negligible. Hence, Theorem 3.1 follows. □

Lemma 3.1. *If the* SD-I *assumption holds, then for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2)}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function negl.*

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2)}(\lambda)|$ is non-negligible. We construct a probabilistic polynomial-time algorithm $\mathcal{B}$ that attempts to solve the SD-I problem using $\mathcal{A}$ as a sub-routine. The description of $\mathcal{B}$ follows:

- $\mathcal{B}$ receives an instance of the SD-I problem $\varpi_{\hat{\beta}} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \check{g}, \Re_{\hat{\beta}})$, where $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{symmetric}, \texttt{composite})$, $g \xleftarrow{\$} \mathbb{G}_{p_1}$, $\check{g} \xleftarrow{\$} \mathbb{G}_{p_3}$, and $\Re_{\hat{\beta}} = g^\sigma \xleftarrow{\$} \mathbb{G}_{p_1}$ or $g^\sigma \hat{g}^{\hat{\sigma}} \xleftarrow{\$} \mathbb{G}_{p_1p_2}$ according as $\hat{\beta} = 0$ or 1 with $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$ and $\sigma, \hat{\sigma} \xleftarrow{\$} \mathbb{Z}_n$. $\mathcal{B}$ then initializes $\mathcal{A}$ on input 1^λ and obtains a challenge vector $\vec{x}^*$ of length ℓ from $\mathcal{A}$.
- In order to setup the NIPPE system, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects $\alpha, \check{\gamma}, \mu, \check{\nu}_1, \ldots, \check{\nu}_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$ and sets $u = g^\mu$, $\check{r}_\iota = \check{g}^{\check{\nu}_\iota}$, for $\iota \in [2\ell]$.
  2. Next, $\mathcal{B}$ computes $h_0 = g^{\check{\gamma} - \sum_{\iota\in[\ell]} x_\iota^* \alpha^\iota}$, $h_\iota = g^{\alpha^\iota}$, for $\iota \in [\ell]$, $\widetilde{u}_\iota = u^{\alpha^\iota} \check{r}_\iota$, for $\iota \in [2\ell]$, and $e(g, \widetilde{u}_{\ell+1})$.
  3. Then, $\mathcal{B}$ uniformly samples $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ from a pairwise independent hash family $\mathbb{H}_2$.
  4. $\mathcal{B}$ provides $\mathcal{A}$ with $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, h_0, \{h_\iota\}_{\iota\in[\ell]}, \{\widetilde{u}_\iota\}_{\iota\in[2\ell]\setminus\{\ell+1\}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$ to $\mathcal{A}$.
- To prepare the challenge $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first sets $c_1^* = \Re_{\hat{\beta}}$, $c_2^* = \Re_{\hat{\beta}}^{\check{\gamma}}$, together with $\text{EK}_0^* = \mathcal{H}(e(\Re_{\hat{\beta}}, \widetilde{u}_{\ell+1}))$, and selects $\text{EK}_1^* \xleftarrow{\$} \{0,1\}^\lambda$.
  2. Next, $\mathcal{B}$ chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and gives $(\text{CT}^* = (\vec{x}^*, c_1^*, c_2^*), \text{EK}_\beta^*)$ to $\mathcal{A}$.

- In response to a decryption key query of $\mathcal{A}$ corresponding to some vector $\vec{y}$ of length ℓ subject to the restriction that $\langle \vec{x}^*, \vec{y} \rangle = 0$, $\mathcal{B}$ executes the following steps:
  1. $\mathcal{B}$ first selects $\breve{\nu}'_{\vec{y}} \xleftarrow{\$} \mathbb{Z}_n$ and computes $\breve{r}'_{\vec{y}} = \breve{g}^{\breve{\nu}'_{\vec{y}}}$.
  2. Then, $\mathcal{B}$ computes $k_{\vec{y}} = (\prod_{\iota' \in [\ell]} \widetilde{u}_{\ell+1-\iota'}^{y_{\iota'}})^{\breve{\gamma}} (\prod_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} \widetilde{u}_{\ell+1+\iota-\iota'}^{-x^*_\iota y_{\iota'}}) \breve{r}'_{\vec{y}}$ exactly as in Hyb_1.
  3. $\mathcal{B}$ hands $\mathrm{SK}(\vec{y}) = (\vec{y}, k_{\vec{y}})$ to $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs a guess bit $\beta' \in \{0,1\}$. If $\beta = \beta'$, then $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Observe that if $\hat{\beta} = 0$, i.e., $\Re_{\hat{\beta}} = g^\sigma \xleftarrow{\$} \mathbb{G}_{p_1}$, then $\mathcal{B}$ perfectly simulates Hyb_1 by *implicitly* viewing $\theta = \sigma$. On the other hand, if $\hat{\beta} = 1$, i.e., $\Re_{\hat{\beta}} = g^\sigma \hat{g}^{\hat{\sigma}} \xleftarrow{\$} \mathbb{G}_{p_1 p_2}$, then $\mathcal{B}$ perfectly simulates Hyb_2 by viewing $w = \Re_{\hat{\beta}}$. This completes the proof of Lemma 3.1. □

Lemma 3.2. *If the* SD-II *assumption holds, then for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(2)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function* negl.

Proof. In order to prove Lemma 3.2, we consider a sequence of 4ℓ intermediate hybrid experiments, namely, $\{\mathsf{Hyb}_{2,\tau,0}, \mathsf{Hyb}_{2,\tau,1}\}_{\tau \in [2\ell]}$ between Hyb_2 and Hyb_3 as follows:

Sequence of Intermediate Hybrids between $\mathbf{Hyb_2}$ and $\mathbf{Hyb_3}$

$\mathbf{Hyb_{2,\tau,0}(\tau \in [2\ell])}$: This experiment is similar to Hyb_2 except that while setting up the NIPPE system, $\mathcal{B}$ sets $\widetilde{u}_\iota = u^{\alpha^\iota} \hat{g}^{\hat{\vartheta}_\tau \alpha^\iota + \sum_{j \in [\tau-1]} \hat{\vartheta}_j \hat{\alpha}^\iota_j} \breve{r}_\iota$, for $\iota \in [2\ell]$, where $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_\tau, \hat{\alpha}_1, \ldots, \hat{\alpha}_{\tau-1} \xleftarrow{\$} \mathbb{Z}_n$, $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$, and the other quantities are the same as in Hyb_2.

$\mathbf{Hyb_{2,\tau,1}(\tau \in [2\ell])}$: This experiment is identical to $\mathsf{Hyb}_{2,\tau,0}$ except that for all $\iota \in [2\ell]$, $\mathcal{B}$ replaces $\hat{g}^{\hat{\vartheta}_\tau \alpha^\iota}$ by $\hat{g}^{\hat{\vartheta}_\tau \hat{\alpha}^\iota_\tau}$ in the expression of $\widetilde{u}_\iota$, where $\hat{\alpha}_\tau \xleftarrow{\$} \mathbb{Z}_n$, i.e., in other words, this experiment is analogous to Hyb_2 with the exception that $\mathcal{B}$ sets $\widetilde{u}_\iota = u^{\alpha^\iota} \hat{g}^{\sum_{j \in [\tau]} \hat{\vartheta}_j \hat{\alpha}^\iota_j} \breve{r}_\iota$, where $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_\tau, \hat{\alpha}_1, \ldots, \hat{\alpha}_\tau \xleftarrow{\$} \mathbb{Z}_n$, $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$, and the other quantities are the same as in Hyb_2.

Analysis

Let $\mathsf{Adv}^{(2,\tau,\widetilde{b})}_{\mathcal{A}}(\lambda)$ represents the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the challenge bit selected by the challenger $\mathcal{B}$, in $\mathsf{Hyb}_{2,\tau,\widetilde{b}}$, for $\tau \in [2\ell], \widetilde{b} \in \{0,1\}$. Clearly, $\mathsf{Hyb}_{2,0,1}$ coincides with Hyb_2 while $\mathsf{Hyb}_{2,2\ell,1}$ corresponds to Hyb_3. Hence, $\mathsf{Adv}^{(2)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(2,0,1)}_{\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(2,2\ell,1)}_{\mathcal{A}}(\lambda)$. Therefore, we have

$$\left|\mathsf{Adv}^{(2)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda)\right| \leq \sum_{\tau \in [2\ell]} \left|\mathsf{Adv}^{(2,(\tau-1),1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda)\right| + \sum_{\tau \in [2\ell]} \left|\mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2,\tau,1)}_{\mathcal{A}}(\lambda)\right|. \tag{3.2}$$

Now, observe that for all $\tau \in [2\ell]$, $\mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{(2,\tau,1)}_{\mathcal{A}}(\lambda)$. This follows from the fact that $\alpha \mod p_1$ and $\alpha \mod p_2$ are independent random values by the Chinese Remainder Theorem, as well as $\alpha \mod p_2$ is completely hidden to the adversary $\mathcal{A}$ given MPK, $(\mathrm{CT}^*, \mathrm{EK}^*_\beta)$, and $\mathrm{SK}(\vec{y})$'s queried by $\mathcal{A}$. Therefore, $\alpha \mod p_2$ can be replaced by $\alpha_\tau \mod p_2$ for a fresh $\alpha_\tau \xleftarrow{\$} \mathbb{Z}_n$. Moreover, Claim 3.1 shows that the first term in the RHS of Eq. (3.2) is negligible. Hence, Lemma 3.2 follows. □

Claim 3.1. *If the* SD-II *assumption holds, then for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(2,(\tau-1),1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function* negl.

Proof. Suppose that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ for which $|\mathsf{Adv}^{(2,(\tau-1),1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda)|$ is non-negligible. Below we construct a probabilistic polynomial-time algorithm $\mathcal{B}$ that attempts to solve the SD-II problem using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ receives an instance of the SD-II problem $\varpi_{\hat{\beta}} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, v, s, \Re_{\hat{\beta}})$, where $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{symmetric}, \texttt{composite})$, $g \xleftarrow{\$} \mathbb{G}_{p_1}$, $\breve{g} \xleftarrow{\$} \mathbb{G}_{p_3}$, $v = g^\sigma \hat{g}^{\hat{\sigma}} \xleftarrow{\$} \mathbb{G}_{p_1p_2}$, $s = \hat{g}^{\hat{\varphi}} \breve{g}^{\breve{\varphi}} \xleftarrow{\$} \mathbb{G}_{p_2p_3}$, and $\Re_{\hat{\beta}} = g^\omega \breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}_{p_1p_3}$ or $g^\omega \hat{g}^{\hat{\omega}} \breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}$ according as $\hat{\beta} = 0$ or 1 with $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$, and $\sigma, \hat{\sigma}, \hat{\varphi}, \breve{\varphi}, \omega, \hat{\omega}, \breve{\omega} \xleftarrow{\$} \mathbb{Z}_n$. $\mathcal{B}$ then initializes $\mathcal{A}$ on input 1^λ and obtains a challenge vector $\vec{x}^*$ of length ℓ from $\mathcal{A}$.
- In order to setup the NIPPE system, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects $\alpha, \hat{\alpha}_1, \ldots, \hat{\alpha}_{\tau-1}, \hat{\vartheta}'_1, \ldots, \hat{\vartheta}'_{\tau-1}, \breve{\gamma}, \breve{\nu}'_1, \ldots, \breve{\nu}'_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$, and sets $\breve{r}'_\iota = \breve{g}^{\breve{\nu}'_\iota}$, for $\iota \in [2\ell]$.
  2. After that, $\mathcal{B}$ computes $h_0 = g^{\breve{\gamma} - \sum_{\iota \in [\ell]} x^*_\iota \alpha^\iota}$, $h_\iota = g^{\alpha^\iota}$, for $\iota \in [\ell]$, $\widetilde{u}_\iota = \Re_{\hat{\beta}}^{\alpha^\iota} s^{\sum_{j \in [\tau-1]} \hat{\vartheta}'_j \hat{\alpha}^\iota_j} \breve{r}'_\iota$, for $\iota \in [2\ell]$, and $e(g, \widetilde{u}_{\ell+1})$.
  3. Then, $\mathcal{B}$ uniformly samples $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ from a pairwise independent hash family $\mathbb{H}_2$.
  4. $\mathcal{B}$ provides $\mathcal{A}$ with $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, h_0, \{h_\iota\}_{\iota \in [\ell]}, \{\widetilde{u}_\iota\}_{\iota \in [2\ell] \setminus \{\ell+1\}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$.
- To frame the challenge, $\mathcal{B}$ executes the following steps:
  1. $\mathcal{B}$ first sets $c^*_1 = v$, $c^*_2 = v^{\breve{\gamma}}$, and $\text{EK}^*_0 = \mathcal{H}(e(v, \widetilde{u}_{\ell+1})$, while selects $\text{EK}^*_1 \xleftarrow{\$} \{0,1\}^\lambda$.
  2. Then, $\mathcal{B}$ chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and gives $(\text{CT}^* = (\vec{x}^*, c^*_1, c^*_2), \text{EK}^*_\beta)$ to $\mathcal{A}$.
- In response to a decryption key query of $\mathcal{A}$ corresponding to some vector $\vec{y}$ of length ℓ subject to the restriction that $\langle \vec{x}^*, \vec{y} \rangle = 0$, $\mathcal{B}$ performs the following steps:
  1. $\mathcal{B}$ first selects $\breve{\nu}'_{\vec{y}} \xleftarrow{\$} \mathbb{Z}_n$ and sets $\breve{r}'_{\vec{y}} = \breve{g}^{\breve{\nu}'_{\vec{y}}}$.
  2. Then, $\mathcal{B}$ computes $k_{\vec{y}} = (\prod_{\iota' \in [\ell]} \widetilde{u}_{\ell+1-\iota'}^{y_{\iota'}})^{\breve{\gamma}} (\prod_{\substack{\iota, \iota' \in [\ell] \\ \iota \neq \iota'}} \widetilde{u}_{\ell+1+\iota-\iota'}^{-x^*_\iota y_{\iota'}}) \breve{r}'_{\vec{y}}$ exactly as in $\mathsf{Hyb}_{2,(\tau-1),1}$.
  3. $\mathcal{B}$ hands $\text{SK}(\vec{y}) = (\vec{y}, k_{\vec{y}})$ to $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs a guess bit $\beta' \in \{0,1\}$. If $\beta = \beta'$, then $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Observe that in case $\hat{\beta} = 0$, i.e., $\Re_{\hat{\beta}} = g^\omega \breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}_{p_1p_3}$, then for all $\iota \in [2\ell]$, $\widetilde{u}_\iota = (g^\omega)^{\alpha^\iota} \hat{g}^{\sum_{j \in [\tau-1]} \hat{\varphi} \hat{\vartheta}'_j \hat{\alpha}^\iota_j} \breve{r}_\iota$, where $\breve{r}_\iota = (\breve{g}^{\breve{\omega}})^{\alpha^\iota} (\breve{g}^{\breve{\varphi}})^{\sum_{j \in [\tau-1]} \hat{\vartheta}'_j \hat{\alpha}^\iota_j} \breve{r}'_\iota$, and thus $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{2,(\tau-1),1}$. On the other hand, if $\hat{\beta} = 1$, i.e., $\Re_{\hat{\beta}} = g^\omega \hat{g}^{\hat{\omega}} \breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}$, then for all $\iota \in [2\ell]$, $\widetilde{u}_\iota = (g^\omega)^{\alpha^\iota} \hat{g}^{\hat{\omega}\alpha^\iota + \sum_{j \in [\tau-1]} \hat{\varphi} \hat{\vartheta}'_j \hat{\alpha}^\iota_j} \breve{r}_\iota$, where $\breve{r}_\iota = (\breve{g}^{\breve{\omega}})^{\alpha^\iota} (\breve{g}^{\breve{\varphi}})^{\sum_{j \in [\tau-1]} \hat{\vartheta}'_j \hat{\alpha}^\iota_j} \breve{r}'_\iota$ and hence $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{2,\tau,0}$. This completes the proof of Claim 3.1. □

Lemma 3.3. *For any (possibly computationally unbounded) adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(4)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function negl.*

Proof. The difference between Hyb_3 and Hyb_4 is that for all $\iota \in [2\ell]$, $\widetilde{u}_\iota$ is set as $\widetilde{u}_\iota = u^{\alpha^\iota} \hat{g}^{\sum_{j \in [2\ell]} \hat{\vartheta}_j \hat{\alpha}^\iota_j} \breve{r}_\iota$ in Hyb_3, whereas, $\widetilde{u}_\iota = u^{\alpha^\iota} \hat{g}^{\mathcal{R}^{(2\ell)}(\iota)} \breve{r}_\iota$ in Hyb_4, where $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_{2\ell}, \hat{\alpha}_1, \ldots, \hat{\alpha}_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$, and $\mathcal{R}^{(2\ell)} : [2\ell] \to \mathbb{Z}_{p_2}$ is a truly random function. Hence, Lemma 3.3 readily follows from the core lemma of the Déjà Q framework (Lemma 2.2) stated in Section 2.4 and the fact that $\hat{\alpha}_1 \mod p_2, \ldots, \hat{\alpha}_{2\ell} \mod p_2$ are distinct with overwhelming probability over $\hat{\alpha}_1, \ldots, \hat{\alpha}_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$. □

Lemma 3.4. *For any (possibly computationally unbounded) adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(4)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(5)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function negl.*

Proof. The only difference between Hyb_4 and Hyb_5 is that while preparing the challenge, $\mathcal{B}$ selects $\text{EK}^*_0 \xleftarrow{\$} \{0,1\}^\lambda$ in Hyb_5 rather than computing $\text{EK}^*_0 = \mathcal{H}(e(w, \widetilde{u}_{\ell+1}))$ as in Hyb_4, where $w \xleftarrow{\$} \mathbb{G}_{p_1p_2}$, $\widetilde{u}_{\ell+1} = u^{\alpha^\iota} \hat{g}^{\mathcal{R}^{(2\ell)}(\ell+1)} \breve{r}_{\ell+1}$, and $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ is uniformly sampled from a pairwise independent hash family $\mathbb{H}_2$. Now, note that in Hyb_4, while creating the decryption keys $\text{SK}(\vec{y}) = (\vec{y}, k_{\vec{y}})$ queried by $\mathcal{A}$, $k_{\vec{y}}$'s are

formed in the exact same fashion to Hyb_1 as $k_{\vec{y}} = (\prod_{\iota' \in [\ell]} \widetilde{u}_{\ell+1-\iota'}^{y_{\iota'}})^{\check{\gamma}} (\prod_{\substack{\iota,\iota' \in [\ell] \\ \iota \neq \iota'}} \widetilde{u}_{\ell+1+\iota-\iota'}^{-x_{\iota}^{*} y_{\iota'}}) \check{r}'_{\vec{y}}$, which depend on $\widetilde{u}_1, \ldots, \widetilde{u}_\ell, \widetilde{u}_{\ell+2}, \ldots, \widetilde{u}_{2\ell}$. Also, the public parameters MPK provided to $\mathcal{A}$ includes $\widetilde{u}_1, \ldots, \widetilde{u}_\ell, \widetilde{u}_{\ell+2}, \ldots, \widetilde{u}_{2\ell}$ and $e(g, \widetilde{u}_{\ell+1}) = e(g, u^{\alpha^\iota})$ (by the orthogonality property of the subgroups with respect to e). Thus, the decryption keys $\mathrm{SK}(\vec{y})$ and the public parameters MPK given to $\mathcal{A}$ only reveal information about $\mathcal{R}^{(2\ell)}(1), \ldots, \mathcal{R}^{(2\ell)}(\ell), \mathcal{R}^{(2\ell)}(\ell+2), \ldots, \mathcal{R}^{(2\ell)}(2\ell)$, and leak no information about $\mathcal{R}^{(2\ell)}(\ell+1)$. Therefore, the quantity, from which EK_0^* is derived in Hyb_4, namely, $T = e(w, \widetilde{u}_{\ell+1}) = e(w, u^{\alpha^{\ell+1}}) \cdot \boxed{e(w, \hat{g}^{\mathcal{R}^{(2\ell)}(\ell+1)})}$ has min-entropy $\mathsf{H}_\infty(T) = \log p_2 = \Theta(\lambda)$ coming from $\mathcal{R}^{(2\ell)}(\ell+1)$. This holds as long as the $\mathbb{G}_{p_2}$-component of w is not the identity element of the group, which happens with probability $1 - \frac{1}{p_2}$. Hence, with overwhelming probability the relation $\lambda = \mathsf{H}_\infty(T) - 2\log\frac{1}{\epsilon} - O(1)$ is satisfied by $\epsilon = 2^{-\Omega(\lambda)}$. Further, note that $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ is sampled from a pairwise independent hash family $\mathbb{H}_2$ uniformly and independently of T. Thus, by the left-over hash lemma (Lemma 2.1) stated in Section 2.3, it follows that the statistical distance $\Delta((\mathcal{H}(T), \mathcal{H}), (U_\lambda, \mathcal{H})) \leq 2^{-\Omega(\lambda)}$, where U_λ is the uniform random variable over $\{0,1\}^\lambda$. Thus, in other words, EK_0^* generated in Hyb_4 is nearly uniformly distributed over $\{0,1\}^\lambda$. Hence, Lemma 3.4 follows. □

4 Our Attribute-Based Encryption Scheme

Here, we present an attribute-based encryption (ABE) scheme for monotone span programs attaining the least number of components in the ciphertext among existing similar works. Our ABE scheme is developed in composite order bilinear groups and its selective security is proven under the Subgroup Decision assumptions, which are the most well-studied static assumptions in this setting, through the enriched Déjà Q technique.

4.1 Notion

Definition 4.1 (Monotone Span Programs or Monotone Access Structures: MAS [Bei96, LOS+10, LW11]). Let $\mathbb{U} \subset \mathbb{Z}_n$ be an attribute universe containing ℓ attributes (say), for some $\ell, n \in \mathbb{N}$. Without loss of generality, let us denote the ℓ attributes of $\mathbb{U}$ as $1, \ldots, \ell$. A monotone span program or monotone access structure (MAS) over $\mathbb{U}$ is a labeled matrix $\mathscr{A} = (\boldsymbol{M}, \rho)$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$, for some $m, m' \in \mathbb{N}$, and ρ is a labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$, i.e., $\rho : [m] \to \mathbb{U}$. Let $\Gamma \subseteq \mathbb{U}$ be an attribute set. Let for all $\iota \in [m]$, $\vec{M}_\iota \in \mathbb{Z}_n^{m'}$ denotes the ι^{th} row of the matrix $\boldsymbol{M}$. Consider the set $\mathbb{I}_\Gamma = \{\iota \in [m] \mid \rho(\iota) \in \mathbb{U}\} \subseteq [m]$. The MAS $\mathscr{A} = (\boldsymbol{M}, \rho)$ accepts Γ if and only if there exists $\{\eta_\iota\}_{\iota \in \mathbb{I}_\Gamma} \subset \mathbb{Z}_n$ such that $\vec{1} = \sum_{\iota \in \mathbb{I}_\Gamma} \eta_\iota \vec{M}_\iota$, where $\vec{1} = (1, 0, \ldots, 0) \in \mathbb{Z}_n^{m'}$.

Definition 4.2 (Linear Secret-Sharing Scheme: LSS [Bei96, LOS+10, LW11]). Let $\mathbb{U} = \{1, \ldots, \ell\} \subset \mathbb{Z}_n$ be an attribute universe. A linear secret-sharing (LSS) scheme for the set $\mathfrak{A}$ of MAS's over $\mathbb{U}$ consists of the following two polynomial-time algorithms:

- $\mathsf{LSS.Distribute}(\mathscr{A} = (\boldsymbol{M}, \rho), \varsigma) \to \{\kappa_\iota\}_{\iota \in [m]}$: This algorithm takes as input an MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$ and $\rho : [m] \to \mathbb{U}$ is a labeling of the rows of $\boldsymbol{M}$, along with a secret $\varsigma \in \mathbb{Z}_n$ to be shared. It first selects $y_2, \ldots, y_{m'} \xleftarrow{\$} \mathbb{Z}_n$, and sets $\vec{y} = (\varsigma, y_2, \ldots, y_{m'}) \in \mathbb{Z}_n^{m'}$. Next, it computes $\kappa_\iota = \vec{M}_\iota \vec{y}^{\mathsf{T}} \in \mathbb{Z}_n$, for $\iota \in [m]$, where $\vec{M}_\iota$ represents the ι^{th} row of $\boldsymbol{M}$. The share associated with the attribute $\rho(\iota)$ is defined to be κ_ι, for $\iota \in [m]$. It outputs the set $\{\kappa_\iota\}_{\iota \in [m]}$ of the m shares.
- $\mathsf{LSS.Reconstruct}(\mathscr{A} = (\boldsymbol{M}, \rho), \Gamma) \to (\mathbb{I}_\Gamma, \{\eta_\iota\}_{\iota \in \mathbb{I}_\Gamma})$: This algorithm takes as input an MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$, $m, m' \in \mathbb{N}$, and $\rho : [m] \to \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$, together with a set $\Gamma \subseteq \mathbb{U}$ of attributes accepted by $\mathscr{A}$. Since, $\mathscr{A}$ accepts Γ, there exists $\mathbb{I}_\Gamma \subseteq [m]$, and $\{\eta_\iota\}_{\iota \in \mathbb{I}_\Gamma} \subset \mathbb{Z}_n$ such that $\vec{1} = \sum_{\iota \in \mathbb{I}_\Gamma} \eta_\iota \vec{M}_\iota$, where $\vec{M}_\iota$ denotes the ι^{th} row of $\boldsymbol{M}$. This algorithm determines and outputs $\mathbb{I}_\Gamma$ and $\{\eta_\iota\}_{\iota \in \mathbb{I}_\Gamma}$.

Consider MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$ and $\rho : [m] \to \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$. Observe that for any $\Gamma \subset \mathbb{U}$ accepted by $\mathscr{A}$ and any secret $\varsigma \in \mathbb{Z}_n$, if

$\{\kappa_\iota\}_{\iota\in[m]} \xleftarrow{\$} \mathsf{LSS.Distribute}(\mathscr{A}=(\boldsymbol{M},\rho),\varsigma)$ and $(\mathbb{I}_\Gamma,\{\eta_\iota\}_{\iota\in\mathbb{I}_\Gamma}) \xleftarrow{\$} \mathsf{LSS.Reconstruct}(\mathscr{A}=(\boldsymbol{M},\rho),\Gamma)$, then

$$\sum_{\iota\in\mathbb{I}_\Gamma}\eta_\iota\kappa_\iota = \sum_{\iota\in\mathbb{I}_\Gamma}\eta_\iota\vec{M}_\iota\vec{y}^\intercal = \vec{1}\vec{y}^\intercal = \varsigma.$$

We will use the following result on LSS schemes in our security proof.

Lemma 4.1 ([Bei96]). *Consider an* MAS $\mathscr{A}=(\boldsymbol{M},\rho)$ *over an attribute universe* $\mathbb{U}=\{1,\ldots,\ell\}\subset\mathbb{Z}_n$, *where* $\boldsymbol{M}\in\mathbb{Z}_n^{m\times m'}$ *and* $\rho:[m]\to\mathbb{U}$ *is the labeling of the rows of* $\boldsymbol{M}$ *with attributes in* $\mathbb{U}$. *Let* $\Gamma\subseteq\mathbb{U}$ *be a set of attributes. If* $\mathscr{A}$ *does not accept* Γ, *then there exists a vector* $\vec{d}=(-1,d_2,\ldots,d_{m'})\in\mathbb{Z}_n^{m'}$ *for which* $\vec{M}_\iota\vec{d}^\intercal=0$, *for all* $\iota\in\mathbb{I}_\Gamma=\{\iota\in[m]\mid\rho(\iota)\in\Gamma\}$. *Moreover, the vector* $\vec{d}$ *can be determined by a polynomial-time algorithm.*

Definition 4.3 (Attribute-Based Encryption: ABE [ALDP11,YAHK14,Att14]). Let $\mathbb{U}=\{1,\ldots,\ell\}\subset\mathbb{Z}_n$ be an attribute universe and $\mathfrak{A}$ be a family of MAS's over $\mathbb{U}$. A (key-policy) attribute-based encryption (ABE) scheme for attributes in $\mathbb{U}$ supporting MAS's in $\mathfrak{A}$ consists of the following polynomial-time algorithms:

- $\mathsf{ABE.Setup}(1^\lambda,\mathbb{U})\to(\text{MPK},\text{MSK})$: The setup authority takes as input the unary encoded security parameter 1^λ along with the attribute universe $\mathbb{U}$. It publishes the public parameters MPK, while keeps the master secret key MSK to itself.
- $\mathsf{ABE.KeyGen}(\text{MPK},\text{MSK},\mathscr{A})\to\text{SK}(\mathscr{A})$: Taking as input the public parameters MPK, the master secret key MSK, and an MAS $\mathscr{A}=(\boldsymbol{M},\rho)\in\mathfrak{A}$, the setup authority creates a decryption key $\text{SK}(\mathscr{A})$ (which includes the description of the MAS $\mathscr{A}$ in the clear) to a legitimate decrypter.
- $\mathsf{ABE.Encrypt}(\text{MPK},\Gamma)\to(\text{CT}(\Gamma),\text{EK})$: On input the public parameters MPK together with an attribute set $\Gamma\subseteq\mathbb{U}$, an encrypter prepares a ciphertext $\text{CT}(\Gamma)$ (which includes the attribute set Γ in the clear) and a session key EK.
- $\mathsf{ABE.Decrypt}(\text{MPK},\text{SK}(\mathscr{A}),\text{CT}(\Gamma))\to\text{EK}'$ or $\perp$: A decrypter takes as input the public parameters MPK, a decryption key $\text{SK}(\mathscr{A})$ corresponding to its MAS $\mathscr{A}=(\boldsymbol{M},\rho)\in\mathfrak{A}$, and a ciphertext $\text{CT}(\Gamma)$ created for the attribute set $\Gamma\subseteq\mathbb{U}$. It either outputs a session key EK' or outputs $\perp$ indicating failure.

The algorithms ABE.Setup, ABE.KeyGen, and ABE.Encrypt are randomized, while the algorithm ABE.Decrypt is deterministic.

■ **Correctness**: An ABE is correct if for any security parameter λ, any $n\in\mathbb{N}$, any attribute universe $\mathbb{U}\subset\mathbb{Z}_n$, any MAS $\mathscr{A}=(\boldsymbol{M},\rho)$ over $\mathbb{U}$, and any attribute set $\Gamma\subseteq\mathbb{U}$ accepted by $\mathscr{A}$, we have

$$\mathsf{Pr}\Big[(\text{MPK},\text{MSK})\xleftarrow{\$}\mathsf{ABE.Setup}(1^\lambda,\mathbb{U});\ \text{SK}(\mathscr{A})\xleftarrow{\$}\mathsf{ABE.KeyGen}(\text{MPK},\text{MSK},\mathscr{A});$$
$$(\text{CT}(\Gamma),\text{EK})\xleftarrow{\$}\mathsf{ABE.Encrypt}(\text{MPK},\Gamma)\ :\ \mathsf{ABE.Decrypt}(\text{MPK},\text{SK}(\mathscr{A}),\text{CT}(\Gamma))=\text{EK}\Big]=1.$$

■ **Security**: The selective indistinguishability-based (SEL-IND) security of an ABE scheme is formalized in terms of the following experiment involving a probabilistic polynomial-time adversary $\mathcal{A}$ and a probabilistic polynomial-time challenger $\mathcal{B}$:

- $\mathcal{A}$ declares a challenge attribute set $\Gamma^*\subseteq\mathbb{U}$ to $\mathcal{B}$.
- $\mathcal{B}$ generates $(\text{MPK},\text{MSK})\xleftarrow{\$}\mathsf{ABE.Setup}(1^\lambda,\mathbb{U})$ and provides MPK to $\mathcal{A}$.
- To prepare the challenge, $\mathcal{B}$ first creates $(\text{CT}^*,\text{EK}_0^*)\xleftarrow{\$}\mathsf{ABE.Encrypt}(\text{MPK},\Gamma^*)$ and samples $\text{EK}_1^*\xleftarrow{\$}\{0,1\}^\lambda$. Next, $\mathcal{B}$ selects a bit $\beta\in\{0,1\}$ and gives $(\text{CT}^*,\text{EK}_\beta^*)$ to $\mathcal{A}$.
- $\mathcal{A}$ may adaptively query any polynomial number of decrytion keys. In response to a decryption key query of $\mathcal{A}$ corresponding to MAS $\mathscr{A}=(\boldsymbol{M},\rho)\in\mathfrak{A}$ subject to the restriction that $\mathscr{A}$ does not accept Γ^*, $\mathcal{B}$ creates $\text{SK}(\mathscr{A})\xleftarrow{\$}\mathsf{ABE.KeyGen}(\text{MPK},\text{MSK},\mathscr{A})$ and hands $\text{SK}(\mathscr{A})$ to $\mathcal{A}$.
- Finally, $\mathcal{A}$ outputs a guess bit $\beta'\in\{0,1\}$.

An ABE scheme is said to be SEL-IND secure if for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE,SEL\text{-}IND}}(\lambda)=\big|\mathsf{Pr}[\beta=\beta']-1/2\big|\le\mathsf{negl}(\lambda),$$

for some negligible function negl.

4.2 Construction

$\mathsf{ABE.Setup}(1^\lambda, \mathbb{U}) \to (\text{MPK}, \text{MSK})$: The setup authority takes as input the unary encoded security parameter 1^λ along with an attribute universe $\mathbb{U} = \{1, \ldots, \ell\}$. It proceeds as follows:

1. It first generates $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{symmetric}, \texttt{composite})$.
2. Next, it selects $\mu, \alpha, \gamma \xleftarrow{\$} \mathbb{Z}_n$, $g \xleftarrow{\$} \mathbb{G}_{p_1}$, and $\breve{g}, \breve{r}_0, \breve{r}_1, \ldots, \breve{r}_{\ell+1} \xleftarrow{\$} \mathbb{G}_{p_3}$.
3. Then, it computes $h_0 = g^\gamma \breve{r}_0$, $u = g^\mu$, $\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}} \breve{r}_{\iota'}$, for $\iota' \in [\ell + 1]$, and $e(g, \widetilde{u}_{\ell+1})$.
4. After that, it uniformly samples $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ from a pairwise independent hash family $\mathbb{H}_2$.
5. It publishes the public parameters $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, h_0, \{\widetilde{u}_{\iota'}\}_{\iota' \in \mathbb{U}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$, while keeps the master secret key $\text{MSK} = (p_1p_2p_3, \mu, \alpha, \gamma)$.

$\mathsf{ABE.KeyGen}(\text{MPK}, \text{MSK}, \mathscr{A}) \to \text{SK}(\mathscr{A})$: On input the public parameters $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, h_0, \{\widetilde{u}_{\iota'}\}_{\iota' \in \mathbb{U}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$, the master secret key $\text{MSK} = (p_1p_2p_3, \mu, \alpha, \gamma)$, and an MAS $\mathscr{A} = (\boldsymbol{M}, \rho)$ belonging to the family $\mathfrak{A}$ of MAS's over $\mathbb{U}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$ and $\rho : [m] \to \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$, the setup authority executes the following steps:

1. It first computes m shares $\{\kappa_\iota\}_{\iota \in [m]} \xleftarrow{\$} \mathsf{LSS.Distribute}(\mathscr{A} = (\boldsymbol{M}, \rho), \varsigma = \mu\alpha^{\ell+1})$ of the secret $\varsigma = \mu\alpha^{\ell+1}$.
2. For $\iota \in [m]$, it performs the following:
   a) It picks $\wp_\iota \xleftarrow{\$} \mathbb{Z}_n$, $\breve{r}'_\iota, \breve{r}'_{\iota,\iota'} \xleftarrow{\$} \mathbb{G}_{p_3}$, for $\iota' \in \mathbb{U} \backslash \{\rho(\iota)\}$.
   b) It computes $k_\iota = g^{\kappa_\iota} g^{(\gamma + \mu\alpha^{\rho(\iota)})\wp_\iota} \breve{r}'_\iota$, $k'_\iota = g^{\wp_\iota}$, $k''_{\iota,\iota'} = g^{\mu\alpha^{\iota'}\wp_\iota} \breve{r}'_{\iota,\iota'}$, for $\iota' \in \mathbb{U} \backslash \{\rho(\iota)\}$.
3. It provides the decryption key $\text{SK}(\mathscr{A}) = (\mathscr{A}, \{k_\iota, k'_\iota, \{k''_{\iota,\iota'}\}_{\iota' \in \mathbb{U} \backslash \{\rho(\iota)\}}\}_{\iota \in [m]})$ to a legitimate decrypter.

$\mathsf{ABE.Encrypt}(\text{MPK}, \Gamma) \to (\text{CT}(\Gamma), \text{EK})$: On input the public parameters $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, h_0, \{\widetilde{u}_{\iota'}\}_{\iota' \in \mathbb{U}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$ along with an attribute set $\Gamma \subseteq \mathbb{U}$, an encrypter operates as follows:

1. It picks $\theta, \breve{\nu}_\Gamma \xleftarrow{\$} \mathbb{Z}_n$, and sets $\breve{r}_\Gamma = \breve{g}^{\breve{\nu}_\Gamma}$.
2. It sets $c_1 = g^\theta$, $c_2 = (h_0 \prod_{\iota' \in \Gamma} \widetilde{u}_{\iota'})^\theta \breve{r}_\Gamma = g^{(\gamma + \mu \sum_{\iota' \in \Gamma} \alpha^{\iota'})\theta} (\breve{r}_0 \prod_{\iota' \in \Gamma} \breve{r}_{\iota'})^\theta \breve{r}_\Gamma$, and $T = e(g, \widetilde{u}_{\ell+1})^\theta$.
3. It outputs the ciphertext $\text{CT}(\Gamma) = (\Gamma, c_1, c_2)$ and the session key $\text{EK} = \mathcal{H}(T)$.

$\mathsf{ABE.Decrypt}(\text{MPK}, \text{SK}(\mathscr{A}), \text{CT}(\Gamma)) \to \text{EK}'$ or $\perp$: A decrypter takes as input the public parameters $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, h_0, \{\widetilde{u}_{\iota'}\}_{\iota' \in \mathbb{U}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$, its decryption key $\text{SK}(\mathscr{A}) = (\mathscr{A}, \{k_\iota, k'_\iota, \{k''_{\iota,\iota'}\}_{\iota' \in \mathbb{U} \backslash \{\rho(\iota)\}}\}_{\iota \in [m]})$ corresponding to its legitimate MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$ and $\rho : [m] \to \mathbb{U}$ is a labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$, together with a ciphertext $\text{CT}(\Gamma) = (\Gamma, c_1, c_2)$ prepared for some attribute set $\Gamma \subseteq \mathbb{U}$. If $\mathscr{A}$ does not accept Γ, then it outputs $\perp$. Otherwise, it executes the following steps:

1. It first determines $(\mathbb{I}_\Gamma, \{\eta_\iota\}_{\iota \in \mathbb{I}_\Gamma}) \xleftarrow{\$} \mathsf{LSS.Reconstruct}(\mathscr{A} = (\boldsymbol{M}, \rho), \Gamma))$.
2. Next, it computes $b_1 = \prod_{\iota \in \mathbb{I}_\Gamma} (k_\iota \prod_{\iota' \in \Gamma \backslash \{\rho(\iota)\}} k''_{\iota,\iota'})^{\eta_\iota}$, $b_2 = \prod_{\iota \in \mathbb{I}_\Gamma} (k'_\iota)^{\eta_\iota}$, and $T' = \dfrac{e(c_1, b_1)}{e(c_2, b_2)}$.
3. It retrieves the session key as $\text{EK}' = \mathcal{H}(T')$.

■ **Correctness**: The correctness of the proposed ABE scheme can be verified as follows: Consider any decryption key

$$\text{SK}(\mathscr{A}) = \Big(\mathscr{A} = (\boldsymbol{M}, \rho), \Big\{k_\iota = g^{\kappa_\iota} g^{(\gamma + \mu\alpha^{\rho(\iota)})\wp_\iota} \breve{r}'_\iota, k'_\iota = g^{\wp_\iota}, \{k''_{\iota,\iota'} = g^{\mu\alpha^{\iota'}\wp_\iota} \breve{r}'_{\iota,\iota'}\}_{\iota' \in \mathbb{U} \backslash \rho(\iota)}\Big\}_{\iota \in [m]}\Big)$$

corresponding to some MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$ and $\rho : [m] \to \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$. Let $\text{CT}(\Gamma) = (\Gamma, c_1 = g^\theta, c_2 = g^{(\gamma + \mu \sum_{\iota' \in \Gamma} \alpha^{\iota'})\theta} (\breve{r}_0 \prod_{\iota' \in \Gamma} \breve{r}_{\iota'})^\theta \breve{r}_\Gamma)$ be a ciphertext prepared for some attribute set $\Gamma \subseteq \mathbb{U}$ accepted by the MAS $\mathscr{A}$. Since, $\mathscr{A}$ accepts Γ, by the correctness of the LSS scheme, $\sum_{\iota \in \mathbb{I}_\Gamma} \eta_\iota \kappa_\iota = \mu\alpha^{\ell+1}$, where $(\mathbb{I}_\Gamma, \{\eta_\iota\}_{\iota \in \mathbb{I}_\Gamma}) \xleftarrow{\$} \mathsf{LSS.Reconstruct}(\mathscr{A} =$

$(\boldsymbol{M}, \rho), \Gamma))$. Thus, we have

$$\begin{aligned}
b_1 &= \prod_{\iota \in \mathbb{I}_\Gamma} \Big(k_\iota \prod_{\iota' \in \Gamma \setminus \{\rho(\iota)\}} k''_{\iota,\iota'} \Big)^{\eta_\iota} \\
&= \prod_{\iota \in \mathbb{I}_\Gamma} \Big(g^{\kappa_\iota} g^{(\gamma + \mu\alpha^{\rho(\iota)})\wp_\iota} \breve{r}'_\iota \prod_{\iota' \in \Gamma \setminus \{\rho(\iota)\}} g^{\mu\alpha^{\iota'}\wp_\iota} \breve{r}'_{\iota,\iota'} \Big)^{\eta_\iota} \\
&= g^{\sum\limits_{\iota \in \mathbb{I}_\Gamma} \eta_\iota \kappa_\iota} \; g^{\sum\limits_{\iota \in \mathbb{I}_\Gamma} \left\{ \eta_\iota \wp_\iota \left(\gamma + \mu \sum\limits_{\iota' \in \Gamma} \alpha^{\iota'} \right) \right\}} \prod_{\iota \in \mathbb{I}_\Gamma} \Big(\breve{r}'_\iota \prod_{\iota' \in \Gamma \setminus \{\rho(\iota)\}} \breve{r}'_{\iota,\iota'} \Big)^{\eta_\iota} \\
&= g^{\mu\alpha^{\ell+1}} g^{\sum\limits_{\iota \in \mathbb{I}_\Gamma} \left\{ \eta_\iota \wp_\iota \left(\gamma + \mu \sum\limits_{\iota' \in \Gamma} \alpha^{\iota'} \right) \right\}} \prod_{\iota \in \mathbb{I}_\Gamma} \Big(\breve{r}'_\iota \prod_{\iota' \in \Gamma \setminus \{\rho(\iota)\}} \breve{r}'_{\iota,\iota'} \Big)^{\eta_\iota} \\
&= u^{\alpha^{\ell+1}} g^{\sum\limits_{\iota \in \mathbb{I}_\Gamma} \left\{ \eta_\iota \wp_\iota \left(\gamma + \mu \sum\limits_{\iota' \in \Gamma} \alpha^{\iota'} \right) \right\}} \prod_{\iota \in \mathbb{I}_\Gamma} \Big(\breve{r}'_\iota \prod_{\iota' \in \Gamma \setminus \{\rho(\iota)\}} \breve{r}'_{\iota,\iota'} \Big)^{\eta_\iota}, \text{ since } u = g^\mu.
\end{aligned}$$

On the other hand,

$$b_2 = \prod_{\iota \in \mathbb{I}_\Gamma} (k'_\iota)^{\eta_\iota} = \prod_{\iota \in \mathbb{I}_\Gamma} (g^{\wp_\iota})^{\eta_\iota} = g^{\sum\limits_{\iota \in \mathbb{I}_\Gamma} \eta_\iota \wp_\iota}.$$

Therefore,

$$\begin{aligned}
T' = \frac{e(c_1, b_1)}{e(c_2, b_2)} &= \frac{e\Big(g^\theta, u^{\alpha^{\ell+1}} g^{\sum\limits_{\iota \in \mathbb{I}_\Gamma} \{\eta_\iota \wp_\iota (\gamma + \mu \sum\limits_{\iota' \in \Gamma} \alpha^{\iota'})\}}\Big)}{e\Big(g^{(\gamma + \mu \sum\limits_{\iota' \in \Gamma} \alpha^{\iota'})\theta}, g^{\sum\limits_{\iota \in \mathbb{I}_\Gamma} \eta_\iota \wp_\iota}\Big)} \\
&\qquad \text{(by orthogonality of the subgroups of } \mathbb{G} \text{ with respect to } e) \\
&= \frac{e(g,u)^{\theta\alpha^{\ell+1}} e(g,g)^{(\sum\limits_{\iota \in \mathbb{I}_\Gamma} \eta_\iota \wp_\iota)(\gamma + \mu \sum\limits_{\iota' \in \Gamma} \alpha^{\iota'})\theta}}{e(g,g)^{(\sum\limits_{\iota \in \mathbb{I}_\Gamma} \eta_\iota \wp_\iota)(\gamma + \mu \sum\limits_{\iota' \in \Gamma} \alpha^{\iota'})\theta}} \\
&= e(g,u)^{\theta\alpha^{\ell+1}} = e(g, \widetilde{u}_{\ell+1})^\theta = T.
\end{aligned}$$

Hence, $\text{EK}' = \mathcal{H}(T') = \mathcal{H}(T) = \text{EK}$.

4.3 Security Analysis

Theorem 4.1 (Security of Our ABE Scheme). *The* ABE *scheme proposed in Section 4.2 is* SEL-IND *secure, as per the security model described in Section 4.1, under the* SD-I *and* SD-II *assumptions.*

Proof. In order to prove Theorem 4.1, we again consider a sequence of hybrid experiments. The first hybrid corresponds to the real SEL-IND security experiment described in Section 4.1 and the final hybrid corresponds to one in which the adversary has no advantage. The sequence of hybrid experiments is described below:

Sequence of Hybrid Experiments

Hyb_0: This experiment corresponds to the real SEL-IND security experiment described in Section 4.1. More precisely, this experiment proceeds as follows:

- The adversary $\mathcal{A}$ submits a challenge attribute set $\Gamma^* \subseteq \mathbb{U}$ to the challenger $\mathcal{B}$.
- $\mathcal{B}$ forms $\Big(\text{MPK} = ((n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, h_0 = g^\gamma \breve{r}_0, \{\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}} \breve{r}_{\iota'}\}_{\iota' \in \mathbb{U}}, e(g, \widetilde{u}_{\ell+1} = u^{\alpha^{\ell+1}} \breve{r}_{\ell+1}), \mathcal{H}),$ $\text{MSK} = (p_1, p_2, p_3, \mu, \alpha, \gamma)\Big) \xleftarrow{\$} \mathsf{ABE.Setup}(1^\lambda, \mathbb{U} = \{1, \ldots, \ell\})$ and provides $\mathcal{A}$ with MPK.

- In order to create the challenge for $\mathcal{A}$, $\mathcal{B}$ begins by preparing $\Big(\text{CT}^* = (\Gamma^*,\ c_1^* = g^\theta, c_2^* = g^{(\gamma+\mu\sum_{\iota'\in\Gamma^*}\alpha^{\iota'})\theta}\breve{r}_{\Gamma^*}), \text{EK}_0^* = \mathcal{H}(e(g,\widetilde{u}_{\ell+1} = g^{\mu\alpha^{\ell+1}}\breve{r}_{\ell+1})^\theta)\Big) \xleftarrow{\$} \mathsf{ABE.Encrypt}(\text{MPK}, \Gamma^*)$ and samples $\text{EK}_1^* \xleftarrow{\$} \{0,1\}^\lambda$. Next, $\mathcal{B}$ selects a bit $\beta \xleftarrow{\$} \{0,1\}$ and gives $(\text{CT}^*, \text{EK}_\beta^*)$ to $\mathcal{A}$.
- In response to a decryption key query of $\mathcal{A}$ corresponding to some MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m\times m'}$ and $\rho : [m] \to \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$, subject to the restriction that $\mathscr{A}$ does not accept Γ^*, $\mathcal{B}$ generates $\text{SK}(\mathscr{A}) = \Big(\mathscr{A}, \{k_\iota = g^{\kappa_\iota} g^{(\gamma+\mu\alpha^{\rho(\iota)})\wp_\iota}\breve{r}'_\iota, k'_\iota = g^{\wp_\iota}, \{k''_{\iota,\iota'} = g^{\mu\alpha^{\iota'}\wp_\iota}\breve{r}'_{\iota,\iota'}\}_{\iota'\in\mathbb{U}\setminus\{\rho(\iota)\}}\}_{\iota\in[m]}\Big) \xleftarrow{\$} \mathsf{ABE.KeyGen}(\text{MPK}, \text{MSK}, \mathscr{A})$ and hands $\text{SK}(\mathscr{A})$ to $\mathcal{A}$.
- At the end, $\mathcal{A}$ outputs a guess bit $\beta' \in \{0,1\}$.

Hyb_1: This experiment proceeds in the following way:

- $\mathcal{A}$ submits a challenge attribute set $\Gamma^* \subseteq \mathbb{U}$ to $\mathcal{B}$.
- To setup the ABE system, $\mathcal{B}$ operates as follows:
  1. $\mathcal{B}$ first generates $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{symmetric}, \texttt{composite})$.
  2. Then, $\mathcal{B}$ selects $\mu, \alpha, \breve{\gamma} \xleftarrow{\$} \mathbb{Z}_n$, $g \xleftarrow{\$} \mathbb{G}_{p_1}$, $\breve{g}, \breve{r}'_0, \breve{r}_1, \ldots, \breve{r}_{2\ell} \xleftarrow{\$} \mathbb{G}_{p_3}$, and sets $\underline{\gamma = \breve{\gamma} - \mu\sum_{\iota'\in\Gamma^*}\alpha^{\iota'}}$, $u = g^\mu$, $\underline{\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}}\breve{r}_{\iota'}, \text{ for } \iota' \in [2\ell]}$.
  3. Next, $\mathcal{B}$ forms $h_0 = g^{\breve{\gamma}}(\prod_{\iota'\in\Gamma^*}\widetilde{u}_{\iota'})^{-1}\breve{r}'_0 = g^{\breve{\gamma}-\sum_{\iota'\in\Gamma^*}\alpha^{\iota'}}(\prod_{\iota'\in\Gamma^*}\breve{r}_{\iota'})^{-1}\breve{r}'_0 = g^\gamma\breve{r}_0$, where $\breve{r}_0 = (\prod_{\iota'\in\Gamma^*}\breve{r}_{\iota'})^{-1}\breve{r}'_0$, and $e(g, \widetilde{u}_{\ell+1})$.
  4. After that, $\mathcal{B}$ uniformly samples $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ from a pairwise independent hash family $\mathbb{H}_2$.
  5. $\mathcal{B}$ provides $\mathcal{A}$ with $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, h_0, \{\widetilde{u}_{\iota'}\}_{\iota'\in\mathbb{U}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$.
- In order to frame the challenge, $\mathcal{B}$ proceeds as follows:
  1. First, $\mathcal{B}$ chooses $\theta \in \mathbb{Z}_n$ and $\breve{r}_{\Gamma^*} \xleftarrow{\$} \mathbb{G}_{p_3}$.
  2. It forms $c_1^* = g^\theta$, $c_2^* = (g^\theta)^{\breve{\gamma}}(\breve{r}'_0)^\theta\breve{r}_{\Gamma^*} = g^{(\gamma+\sum_{\iota'\in\Gamma^*}\alpha^{\iota'})\theta}(\breve{r}_0\prod_{\iota'\in\Gamma^*}\breve{r}_{\iota'})^\theta\breve{r}_{\Gamma^*}$, $\text{EK}_0^* = \mathcal{H}(e(g^\theta, \widetilde{u}_{\ell+1}))$, and selects $\text{EK}_1^* \xleftarrow{\$} \{0,1\}^\lambda$.
  3. Next, $\mathcal{B}$ picks a bit $\beta \xleftarrow{\$} \{0,1\}$ and gives $(\text{CT}^* = (\Gamma^*, c_1^*, c_2^*), \text{EK}_\beta^*)$ to $\mathcal{A}$.
- In response to a decryption key query of $\mathcal{A}$ corresponding to an MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m\times m'}$ and $\rho : [m] \to \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$, subject to the restriction that $\mathscr{A}$ does not accept Γ^*, $\mathcal{B}$ executes the following steps:
  1. Since, $\mathscr{A} = (\boldsymbol{M}, \rho)$ does not accept Γ^*, thanks to Lemma 4.1 of Section 4.1, $\mathcal{B}$ can determine in polynomial time $\vec{d} = (-1, d_2, \ldots, d_{m'}) \in \mathbb{Z}_n^{m'}$ such that $\vec{M}_\iota\vec{d}^{\,\intercal} = 0$, for all row vectors $\vec{M}_\iota$ of $\boldsymbol{M}$ such that $\iota \in \mathbb{I}_{\Gamma^*}$, where $\mathbb{I}_{\Gamma^*} = \{\iota \in [m] : \rho(\iota) \in \Gamma^*\}$. $\mathcal{B}$ picks $y'_2, \ldots, y'_{m'} \xleftarrow{\$} \mathbb{Z}_n$ and sets $\underline{\vec{y} = (\mu\alpha^{\ell+1}, -\mu\alpha^{\ell+1}d_2 + y'_2, \ldots, -\mu\alpha^{\ell+1}d_{m'} + y'_{m'}) = -\mu\alpha^{\ell+1}\vec{d} + \vec{y}'}$, where $\vec{y}' = (0, y'_2, \ldots, y'_{m'}) \in \mathbb{Z}_n^{m'}$. Thus, for any $\iota \in [m]$, $\vec{M}_\iota\vec{y}^{\,\intercal} = -\mu\alpha^{\ell+1}\vec{M}_\iota\vec{d}^{\,\intercal} + \vec{M}_\iota\vec{y}'^{\intercal}$.
  2. For all $\iota \in [m]$, $\mathcal{B}$ generates the decryption key components corresponding to the ι^{th} row of $\boldsymbol{M}$ as follows:

     i) ($\iota \in \mathbb{I}_{\Gamma^*}$): In this case, $\mathcal{B}$ performs the following:

     a) $\mathcal{B}$ computes the share $\underline{\kappa_\iota = \vec{M}_\iota\vec{y}'^{\intercal} = \vec{M}_\iota\vec{y}^{\,\intercal}}$ corresponding to the ι^{th} row of $\boldsymbol{M}$, as $\vec{M}_\iota\vec{d}^{\,\intercal} = 0$ in this case.

     b) $\mathcal{B}$ selects $\wp'_\iota \xleftarrow{\$} \mathbb{Z}_n$, $\breve{r}''_\iota, \breve{r}''_{\iota,\iota'} \xleftarrow{\$} \mathbb{G}_{p_3}$, for $\iota' \in \mathbb{U}\setminus\{\rho(\iota)\}$, and sets $\underline{\wp_\iota = \wp'_\iota + \alpha^{\ell+1-\rho(\iota)}}$.

c) $\mathcal{B}$ computes

$$\begin{aligned}
k_\iota &= g^{\kappa_\iota} g^{\check{\gamma}\wp_\iota}\Big(\prod_{\iota'\in\Gamma^*\setminus\{\rho(\iota)\}}\widetilde{u}_{\iota'}\Big)^{-\wp'_\iota}\Big(\prod_{\iota'\in\Gamma^*\setminus\{\rho(\iota)\}}\widetilde{u}_{\ell+1+\iota'-\rho(\iota)}\Big)^{-1}\check{r}''_\iota\\
&= g^{\kappa_\iota} g^{\check{\gamma}\wp_\iota-\wp'_\iota\mu\sum_{\iota'\in\Gamma^*\setminus\{\rho(\iota)\}}\alpha^{\iota'}-\mu\sum_{\iota'\in\Gamma^*\setminus\{\rho(\iota)\}}\alpha^{\ell+1+\iota'-\rho(\iota)}}\check{r}'_\iota\\
&= g^{\kappa_\iota} g^{\left(\check{\gamma}-\mu\sum_{\iota'\in\Gamma^*\setminus\{\rho(\iota)\}}\alpha^{\iota'}\right)\wp_\iota}\check{r}'_\iota = g^{\kappa_\iota} g^{\left(\gamma+\mu\alpha^{\rho(\iota)}\right)\wp_\iota}\check{r}'_\iota,\\
&\quad\text{where } \check{r}'_\iota = \Big(\prod_{\iota'\in\Gamma^*\setminus\{\rho(\iota)\}}\check{r}_{\iota'}\Big)^{-\wp'_\iota}\Big(\prod_{\iota'\in\Gamma^*\setminus\{\rho(\iota)\}}\check{r}_{\ell+1+\iota'-\rho(\iota)\}}\Big)^{-1}\check{r}''_\iota,\\
k'_\iota &= g^{\wp_\iota},\ \text{and}\\
k''_{\iota,\iota'} &= \widetilde{u}^{\wp'_\iota}_{\iota'}\widetilde{u}_{\ell+1+\iota'-\rho(\iota)}\check{r}''_{\iota,\iota'} = g^{\mu\wp'_\iota\alpha^{\iota'}+\mu\alpha^{\ell+1+\iota'-\rho(\iota)}}\check{r}'_{\iota,\iota'}\\
&= g^{\mu\wp_\iota\alpha^{\iota'}}\check{r}'_{\iota,\iota'},\ \text{for } \iota'\in\mathbb{U}\setminus\{\rho(\iota)\},\\
&\quad\text{where } \check{r}'_{\iota,\iota'} = \check{r}^{\wp'_\iota}_{\iota'}\check{r}_{\ell+1+\iota'-\rho(\iota)}\check{r}''_{\iota,\iota'}.
\end{aligned}$$

ii) ($\iota \notin \mathbb{I}_{\Gamma^*}$): Observe that in this case, the share corresponding to the ι^{th} row of $\boldsymbol{M}$ is $\kappa_\iota = -\mu\alpha^{\ell+1}\vec{M}_\iota\vec{d}^{\intercal} + \vec{M}_\iota\vec{y}'^{\intercal} = \vec{M}_\iota\vec{y}^{\intercal}$. $\mathcal{B}$ forms the decryption key components as follows:

a) $\mathcal{B}$ picks $\wp'_\iota \xleftarrow{\$} \mathbb{Z}_n$, $\check{r}''_\iota, \check{r}''_{\iota,\iota'} \xleftarrow{\$} \mathbb{G}_{p_3}$, for $\iota'\in\mathbb{U}\setminus\{\rho(\iota)\}$, and sets $\wp_\iota = \wp'_\iota + \alpha^{\ell+1-\rho(\iota)}\vec{M}_\iota\vec{d}^{\intercal}$.

b) $\mathcal{B}$ computes

$$\begin{aligned}
k_\iota &= g^{\vec{M}_\iota\vec{y}'^{\intercal}} g^{\check{\gamma}\wp_\iota}\Big(\prod_{\iota'\in\Gamma^*}\widetilde{u}_{\iota'}\Big)^{-\wp'_\iota}\Big(\prod_{\iota'\in\Gamma^*}\widetilde{u}_{\ell+1+\iota'-\rho(\iota)}\Big)^{-\vec{M}_\iota\vec{d}^{\intercal}}\widetilde{u}^{\wp'_\iota}_{\rho(\iota)}\check{r}''_\iota\\
&= g^{\vec{M}_\iota\vec{y}'^{\intercal}} g^{\check{\gamma}\wp_\iota-\wp'_\iota\mu\sum_{\iota'\in\Gamma^*}\alpha^{\iota'}-\vec{M}_\iota\vec{d}^{\intercal}\mu\sum_{\iota'\in\Gamma^*}\alpha^{\ell+1+\iota'-\rho(\iota)}+\mu\wp'_\iota\alpha^{\rho(\iota)}}\check{r}'_\iota\\
&= g^{-\mu\alpha^{\ell+1}\vec{M}_\iota\vec{d}^{\intercal}+\vec{M}_\iota\vec{y}'^{\intercal}} g^{\check{\gamma}\wp_\iota-\wp_\iota\mu\sum_{\iota'\in\Gamma^*}\alpha^{\iota'}+\mu\wp'_\iota\alpha^{\rho(\iota)}+\mu\alpha^{\ell+1}\vec{M}_\iota\vec{d}^{\intercal}}\check{r}'_\iota\\
&= g^{\kappa_\iota} g^{(\gamma+\mu\alpha^{\rho(\iota)})\wp_\iota}\check{r}'_\iota,\\
&\quad\text{where } \check{r}'_\iota = \Big(\textstyle\prod_{\iota'\in\Gamma^*}\check{r}_{\iota'}\Big)^{-\wp'_\iota}\Big(\displaystyle\prod_{\iota'\in\Gamma^*}\check{r}_{\ell+1+\iota'-\rho(\iota)}\Big)^{-\vec{M}_\iota\vec{d}^{\intercal}}\check{r}^{\wp'_\iota}_{\rho(\iota)}\check{r}''_\iota,\\
k'_\iota &= g^{\wp_\iota},\ \text{and}\\
k''_{\iota,\iota'} &= \widetilde{u}^{\wp'_\iota}_{\iota'}\widetilde{u}^{\vec{M}_\iota\vec{d}^{\intercal}}_{\ell+1+\iota'-\rho(\iota)}\check{r}''_{\iota,\iota'} = g^{\mu\wp'_\iota\alpha^{\iota'}+\mu\alpha^{\ell+1+\iota'-\rho(\iota)}\vec{M}_\iota\vec{d}^{\intercal}}\check{r}'_{\iota,\iota'}\\
&= g^{\mu\wp_\iota\alpha^{\iota'}}\check{r}'_{\iota,\iota'},\ \text{for } \iota'\in\mathbb{U}\setminus\{\rho(\iota)\},\\
&\quad\text{where } \check{r}'_{\iota,\iota'} = \check{r}^{\wp'_\iota}_{\iota'}\check{r}^{\vec{M}_\iota\vec{d}^{\intercal}}_{\ell+1+\iota'-\rho(\iota)}\check{r}''_{\iota,\iota'}.
\end{aligned}$$

3. $\mathcal{B}$ hands $\text{SK}(\mathscr{A}) = \Big(\mathscr{A}, \Big\{k_\iota, k'_\iota, \{k''_{\iota,\iota'}\}_{\iota'\in\mathbb{U}\setminus\{\rho(\iota)\}}\Big\}_{\iota\in[m]}\Big)$ to $\mathcal{A}$.

- $\mathcal{A}$ eventually outputs a guess bit $\beta'\in\{0,1\}$.

Hyb_2: This experiment is similar to Hyb_1 with the only exception that while creating the challenge, $\mathcal{B}$ selects $w \xleftarrow{\$} \mathbb{G}_{p_1p_2}$, $\check{r}_{\Gamma^*} \xleftarrow{\$} \mathbb{G}_{p_3}$, and sets $c^*_1 = w$, $c^*_2 = w^{\check{\gamma}}\check{r}_{\Gamma^*}$, and $\text{EK}^*_0 = \mathcal{H}(e(w,\widetilde{u}_{\ell+1}))$.

Hyb_3: This experiment is the same as Hyb_2 with the exception that while setting up the ABE system, $\mathcal{B}$ sets $\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}}\hat{g}^{\sum_{j\in[2\ell]}\hat{\vartheta}_j\hat{\alpha}^{\iota'}_j}\check{r}_{\iota'}$, for $\iota'\in[2\ell]$, where $\hat{\vartheta}_1,\ldots,\hat{\vartheta}_{2\ell},\hat{\alpha}_1,\ldots,\hat{\alpha}_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$, $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$, while as before $u = g^\mu$, for $\mu \xleftarrow{\$} \mathbb{Z}_n$. This in turn affects the distributions of MPK, EK^*_0, and all the $\text{SK}(\mathscr{A})$'s provided to $\mathcal{A}$.

Hyb_4: In this experiment, while setting up the ABE system, $\mathcal{B}$ sets $\underline{\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}} \hat{g}^{\mathcal{R}^{(2\ell)}(\iota')} \breve{r}_{\iota'}}$, for $\iota' \in [2\ell]$, where $\mathcal{R} : [2\ell] \rightarrow \mathbb{Z}_{p_2}$ is a truly random function. This further changes the distributions of MPK, EK_0^*, and all the $\mathrm{SK}(\mathscr{A})$'s given to $\mathcal{A}$. The rest of the experiment proceeds identically to Hyb_3.

Hyb_5: This experiment is analogous to Hyb_4 with the only exception that while framing the challenge, $\mathcal{B}$ selects $\underline{\mathrm{EK}_0^* \xleftarrow{\$} \{0,1\}^\lambda}$.

Analysis

Let $\mathsf{Adv}_{\mathcal{A}}^{(i)}(\lambda)$ be the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the challenge bit, in Hyb_i, for $i \in \{0, \ldots, 5\}$. By definition, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE,SEL\text{-}IND}}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda)$. Also, note that in Hyb_5, both EK_0^* and EK_1^* are uniformly distributed over $\{0,1\}^\lambda$. Therefore, the view of $\mathcal{A}$ in Hyb_5 is statistically independent of the challenge bit $\beta \xleftarrow{\$} \{0,1\}$ selected by the challenger $\mathcal{B}$. Hence, $\mathsf{Adv}_{\mathcal{A}}^{(5)}(\lambda) = 0$. Moreover, it readily follows that the distributions of MPK, CT^*, EK_0^*, and all the $\mathrm{SK}(\mathscr{A})$'s provided to $\mathcal{A}$ in Hyb_0 and those in Hyb_1 are identical. Thus, the view of $\mathcal{A}$ in Hyb_0 and that in Hyb_1 are also the same. Therefore, $\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda)$. Hence, we have

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE,SEL\text{-}IND}}(\lambda) \leq \sum_{i \in [4]} \left| \mathsf{Adv}_{\mathcal{A}}^{(i)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(i+1)}(\lambda) \right|. \tag{4.1}$$

Lemmas 4.2–4.5 will show that the RHS of Eq. (4.1) is negligible. Hence, Theorem 4.1 follows. □

Lemma 4.2. *If the* $\mathsf{SD\text{-}I}$ *assumption holds, then for any probabilistic polynomial-time adversary* $\mathcal{A}$*, for any security parameter* λ*,* $|\mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2)}(\lambda)| \leq \mathsf{negl}(\lambda)$*, for some negligible function* negl*.*

Proof. Suppose that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ for which $|\mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2)}(\lambda)|$ is non-negligible. We construct a PPT algorithm $\mathcal{B}$ that attempts to solve the $\mathsf{SD\text{-}I}$ problem using $\mathcal{A}$ as a sub-routine. The description of $\mathcal{B}$ follows:

- $\mathcal{B}$ receives an instance of the $\mathsf{SD\text{-}I}$ problem $\varpi_{\hat{\beta}} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, \Re_{\hat{\beta}})$, where $(n = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \mathtt{symmetric}, \mathtt{composite})$, $g \xleftarrow{\$} \mathbb{G}_{p_1}$, $\breve{g} \xleftarrow{\$} \mathbb{G}_{p_3}$, and $\Re_{\hat{\beta}} = g^\sigma \xleftarrow{\$} \mathbb{G}_{p_1}$ or $g^\sigma \hat{g}^{\hat{\sigma}} \xleftarrow{\$} \mathbb{G}_{p_1 p_2}$ according as $\hat{\beta} = 0$ or 1 with $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$ and $\sigma, \hat{\sigma} \xleftarrow{\$} \mathbb{Z}_n$. $\mathcal{B}$ then initializes $\mathcal{A}$ on input 1^λ and obtains a challenge attribute set $\Gamma^* \subseteq \mathbb{U}$ from $\mathcal{A}$.
- In order to setup the ABE system, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects $\mu, \alpha, \check{\gamma}, \check{\nu}'_0, \check{\nu}_1, \ldots, \check{\nu}_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$, and sets $u = g^\mu$, $\breve{r}'_0 = \breve{g}^{\check{\nu}'_0}$, $\breve{r}_{\iota'} = \breve{g}^{\check{\nu}_{\iota'}}$, for $\iota' \in [2\ell]$, along with $\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}} \breve{r}_{\iota'}$, for $\iota' \in [2\ell]$.
  2. Next, $\mathcal{B}$ computes $h_0 = g^{\check{\gamma}} (\prod_{\iota' \in \Gamma^*} \widetilde{u}_{\iota'})^{-1} \breve{r}'_0$ and $e(g, \widetilde{u}_{\ell+1})$.
  3. Then, $\mathcal{B}$ uniformly samples $\mathcal{H} : \mathbb{G}_T \rightarrow \{0,1\}^\lambda$ from a pairwise independent hash family $\mathbb{H}_2$.
  4. $\mathcal{B}$ provides $\mathcal{A}$ with $\mathrm{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, h_0, \{\widetilde{u}_{\iota'}\}_{\iota' \in \mathbb{U}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$ to $\mathcal{A}$.
- To prepare the challenge $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first picks $\nu_{\Gamma^*} \xleftarrow{\$} \mathbb{Z}_n$, and sets $\breve{r}_{\Gamma^*} = \breve{g}^{\check{\nu}_{\Gamma^*}}$, $c_1^* = \Re_{\hat{\beta}}$, $c_2^* = \Re_{\hat{\beta}}^{\check{\gamma}} \breve{r}_{\Gamma^*}$, together with $\mathrm{EK}_0^* = \mathcal{H}(e(\Re_{\hat{\beta}}, \widetilde{u}_{\ell+1}))$. $\mathcal{B}$ also selects $\mathrm{EK}_1^* \xleftarrow{\$} \{0,1\}^\lambda$.
  2. Next, $\mathcal{B}$ chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and gives $(\mathrm{CT}^* = (\Gamma^*, c_1^*, c_2^*), \mathrm{EK}_\beta^*)$ to $\mathcal{A}$.
- In response to a decryption key query of $\mathcal{A}$ corresponding to some MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$ and $\rho : [m] \rightarrow \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$, subject to the restriction that $\mathscr{A}$ does not accept Γ^*, $\mathcal{B}$ creates the decryption key $\mathrm{SK}(\mathscr{A}) = \left(\mathscr{A}, \left\{k_\iota, k'_\iota, \{k''_{\iota,\iota'}\}_{\iota' \in \mathbb{U} \setminus \{\rho(\iota)\}}\right\}_{\iota \in [m]}\right)$ exactly as in Hyb_1. While generating $\mathrm{SK}(\mathscr{A})$, $\mathcal{B}$ computes the required random elements of $\mathbb{G}_{p_3}$ by uniformly sampling elements from $\mathbb{Z}_n$ and placing them in the exponent of $\breve{g} \xleftarrow{\$} \mathbb{G}_{p_3}$ included within the challenge $\mathsf{SD\text{-}I}$ problem instance. $\mathcal{B}$ hands $\mathrm{SK}(\mathscr{A})$ to $\mathcal{A}$.
- At the end, $\mathcal{A}$ outputs a guess bit $\beta' \in \{0,1\}$. If $\beta = \beta'$, then $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Observe that if $\hat{\beta} = 0$, i.e., $\Re_{\hat{\beta}} = g^{\sigma} \xleftarrow{\$} \mathbb{G}_{p_1}$, then $\mathcal{B}$ perfectly simulates Hyb_1 by *implicitly* viewing $\theta = \sigma$. On the other hand, if $\hat{\beta} = 1$, i.e., $\Re_{\hat{\beta}} = g^{\sigma}\hat{g}^{\hat{\sigma}} \xleftarrow{\$} \mathbb{G}_{p_1p_2}$, then $\mathcal{B}$ perfectly simulates Hyb_2 by viewing $w = \Re_{\hat{\beta}}$. This completes the proof of Lemma 4.2. □

Lemma 4.3. *If the* SD-II *assumption holds, then for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(2)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function* negl.

Proof. In order to prove Lemma 4.3, we consider a sequence of 4ℓ intermediate hybrid experiments, namely, $\{\mathsf{Hyb}_{2,\tau,0}, \mathsf{Hyb}_{2,\tau,1}\}_{\tau\in[2\ell]}$ between Hyb_2 and Hyb_3 as follows:

Sequence of Intermediate Hybrids between Hyb_2 and Hyb_3

$\mathsf{Hyb}_{2,\tau,0}(\tau \in [2\ell])$: This experiment is similar to Hyb_2 except that while setting up the ABE system, $\mathcal{B}$ sets $\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}}\hat{g}^{\hat{\vartheta}_\tau\alpha^{\iota'} + \sum_{j\in[\tau-1]}\hat{\vartheta}_j\hat{\alpha}_j^{\iota'}}\breve{r}_{\iota'}$, for $\iota' \in [2\ell]$, where $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_\tau, \hat{\alpha}_1, \ldots, \hat{\alpha}_{\tau-1} \xleftarrow{\$} \mathbb{Z}_n$, $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$, and $u = g^{\mu}$, with $\mu \xleftarrow{\$} \mathbb{Z}_n$.

$\mathsf{Hyb}_{2,\tau,1}(\tau \in [2\ell])$: This experiment is identical to $\mathsf{Hyb}_{2,\tau,0}$ except that for all $\iota \in [2\ell]$, $\mathcal{B}$ replaces $\hat{g}^{\hat{\vartheta}_\tau\alpha^{\iota}}$ by $\hat{g}^{\hat{\vartheta}_\tau\hat{\alpha}_\tau^{\iota}}$ in the expression of $\widetilde{u}_\iota$, where $\hat{\alpha}_\tau \xleftarrow{\$} \mathbb{Z}_n$, i.e., in other words, this experiment is analogous to Hyb_2 with the exception that $\mathcal{B}$ sets $\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}}\hat{g}^{\sum_{j\in[\tau]}\hat{\vartheta}_j\hat{\alpha}_j^{\iota'}}\breve{r}_{\iota'}$, for $\iota' \in [2\ell]$, where $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_\tau, \hat{\alpha}_1, \ldots, \hat{\alpha}_\tau \xleftarrow{\$} \mathbb{Z}_n$, $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$, and $u = g^{\mu}$, with $\mu \xleftarrow{\$} \mathbb{Z}_n$.

Analysis

Let $\mathsf{Adv}^{(2,\tau,\widetilde{\beta})}_{\mathcal{A}}(\lambda)$ represents the advantage of the adversary $\mathcal{A}$, i.e., the absolute difference between $1/2$ and $\mathcal{A}$'s probability of correctly guessing the challenge bit selected by the challenger $\mathcal{B}$, in $\mathsf{Hyb}_{2,\tau,\widetilde{\beta}}$, for $\tau \in [2\ell], \widetilde{\beta} \in \{0,1\}$. Clearly, $\mathsf{Hyb}_{2,0,1}$ coincides with Hyb_2 while $\mathsf{Hyb}_{2,2\ell,1}$ corresponds to Hyb_3. Hence, $\mathsf{Adv}^{(2)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(2,0,1)}_{\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) \equiv \mathsf{Adv}^{(2,2\ell,1)}_{\mathcal{A}}(\lambda)$. Therefore, we have

$$\left|\mathsf{Adv}^{(2)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda)\right| \leq \sum_{\tau\in[2\ell]}\left|\mathsf{Adv}^{(2,(\tau-1),1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda)\right| + \sum_{\tau\in[2\ell]}\left|\mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2,\tau,1)}_{\mathcal{A}}(\lambda)\right|. \tag{4.2}$$

Now, observe that for all $\tau \in [2\ell]$, $\mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{(2,\tau,1)}_{\mathcal{A}}(\lambda)$. This follows from the fact that $\alpha \mod p_1$ and $\alpha \mod p_2$ are uniformly and independently random values by the Chinese Remainder Theorem, as well as $\alpha \mod p_2$ is completely hidden to the adversary $\mathcal{A}$ given MPK, $(\mathrm{CT}^*, \mathrm{EK}^*_{\widetilde{\beta}})$, and all the $\mathrm{SK}(\mathscr{A})$'s queried by $\mathcal{A}$. Therefore, $\alpha \mod p_2$ can be replaced with $\hat{\alpha}_\tau \mod p_2$ for a fresh $\hat{\alpha}_\tau \xleftarrow{\$} \mathbb{Z}_n$. Moreover, Claim 4.1 shows that the first term in the RHS of Eq. (4.2) is negligible. Hence, Lemma 4.3 follows. □

Claim 4.1. *If the* SD-II *assumption holds, then for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}^{(2,(\tau-1),1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function* negl.

Proof. Suppose that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ for which $|\mathsf{Adv}^{(2,(\tau-1),1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2,\tau,0)}_{\mathcal{A}}(\lambda)|$ is non-negligible. Below we construct a probabilistic polynomial-time algorithm $\mathcal{B}$ that attempts to solve the SD-II problem using $\mathcal{A}$ as a sub-routine.

- $\mathcal{B}$ receives an instance of the SD-II problem $\varpi_{\hat{\beta}} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, v, s, \Re_{\hat{\beta}})$, where $(n = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^{\lambda}, \mathtt{symmetric}, \mathtt{composite})$, $g \xleftarrow{\$} \mathbb{G}_{p_1}$, $\breve{g} \xleftarrow{\$} \mathbb{G}_{p_3}$, $v = g^{\sigma}\hat{g}^{\hat{\sigma}} \xleftarrow{\$} \mathbb{G}_{p_1p_2}$, $s = \hat{g}^{\hat{\varphi}}\breve{g}^{\breve{\varphi}} \xleftarrow{\$} \mathbb{G}_{p_2p_3}$, and $\Re_{\hat{\beta}} = g^{\omega}\breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}_{p_1p_3}$ or $g^{\omega}\hat{g}^{\hat{\omega}}\breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}$ according as $\hat{\beta} = 0$ or 1 with $\hat{g} \xleftarrow{\$} \mathbb{G}_{p_2}$, and $\sigma, \hat{\sigma}, \hat{\varphi}, \breve{\varphi}, \omega, \hat{\omega}, \breve{\omega} \xleftarrow{\$} \mathbb{Z}_n$. $\mathcal{B}$ then initializes $\mathcal{A}$ on input 1^{λ} and obtains a challenge attribute set $\Gamma^* \subseteq \mathbb{U}$ from $\mathcal{A}$.
- In order to setup the ABE system, $\mathcal{B}$ proceeds as follows:
  1. $\mathcal{B}$ first selects $\alpha, \hat{\alpha}_1, \ldots, \hat{\alpha}_{\tau-1}, \hat{\vartheta}'_1, \ldots, \hat{\vartheta}'_{\tau-1}, \breve{\gamma}, \breve{\nu}'_0, \breve{\nu}'_1, \ldots, \breve{\nu}'_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$, and sets $\breve{r}'_0 = \breve{g}^{\breve{\nu}'_0}$, $\breve{r}'_{\iota'} = \breve{g}^{\breve{\nu}'_{\iota'}}$, for $\iota' \in [2\ell]$, along with $\widetilde{u}_{\iota'} = \Re_{\hat{\beta}}^{\alpha^{\iota'}} s^{\sum_{j\in[\tau-1]}\hat{\vartheta}'_j\hat{\alpha}_j^{\iota'}}\breve{r}'_{\iota'}$, for $\iota' \in [2\ell]$.

2. After that, $\mathcal{B}$ computes $h_0 = g^{\breve{\gamma}}(\prod_{\iota' \in \Gamma^*} \widetilde{u}_{\iota'})^{-1}\breve{r}'_0$ and $e(g, \widetilde{u}_{\ell+1})$.
3. Then, $\mathcal{B}$ uniformly samples $\mathcal{H} : \mathbb{G}_T \rightarrow \{0,1\}^\lambda$ from a pairwise independent hash family $\mathbb{H}_2$.
4. $\mathcal{B}$ provides $\mathcal{A}$ with $\text{MPK} = ((n, \mathbb{G}, \mathbb{G}_T, e), g, \breve{g}, h_0, \{\widetilde{u}_{\iota'}\}_{\iota' \in \mathbb{U}}, e(g, \widetilde{u}_{\ell+1}), \mathcal{H})$.

- To frame the challenge, $\mathcal{B}$ executes the following steps:
  1. $\mathcal{B}$ first selects $\breve{\nu}_{\Gamma^*} \xleftarrow{\$} \mathbb{Z}_n$, and sets $\breve{r}_{\Gamma^*} = \breve{g}^{\breve{\nu}_{\Gamma^*}}$, $c_1^* = v$, $c_2^* = v^{\breve{\gamma}}\breve{r}_{\Gamma^*}$, together with $\text{EK}_0^* = \mathcal{H}(e(v, \widetilde{u}_{\ell+1})$. $\mathcal{B}$ also selects $\text{EK}_1^* \xleftarrow{\$} \{0,1\}^\lambda$.
  2. Then, $\mathcal{B}$ chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and gives $(\text{CT}^* = (\Gamma^*, c_1^*, c_2^*), \text{EK}_\beta^*)$ to $\mathcal{A}$.
- In response to a decryption key query of $\mathcal{A}$ corresponding to some MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$ and $\rho : [m] \rightarrow \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$, subject to the restriction that $\mathscr{A}$ does not accept Γ^*, $\mathcal{B}$ creates the decryption key $\text{SK}(\mathscr{A}) = \Big(\mathscr{A}, \Big\{k_\iota, k'_\iota, \{k''_{\iota,\iota'}\}_{\iota' \in \mathbb{U} \setminus \{\rho(\iota)\}}\Big\}_{\iota \in [m]}\Big)$ exactly as in $\mathsf{Hyb}_{2,\tau-1,1}$. While generating $\text{SK}(\mathscr{A})$, $\mathcal{B}$ computes the required random elements of $\mathbb{G}_{p_3}$ by uniformly sampling elements from $\mathbb{Z}_n$ and placing them in the exponent of $\breve{g} \xleftarrow{\$} \mathbb{G}_{p_3}$ included within the challenge $\mathsf{SD\text{-}II}$ problem instance. $\mathcal{B}$ hands $\text{SK}(\mathscr{A})$ to $\mathcal{A}$.
- At the end, $\mathcal{A}$ outputs a guess bit $\beta' \in \{0,1\}$. If $\beta = \beta'$, then $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Observe that in case $\hat{\beta} = 0$, i.e., $\Re_{\hat{\beta}} = g^\omega \breve{g}^{\breve{\omega}} \xleftarrow{\$} \mathbb{G}_{p_1 p_3}$, then for all $\iota' \in [2\ell]$, $\widetilde{u}_{\iota'} = (g^\omega)^{\alpha^{\iota'}} \hat{g}^{\sum_{j \in [\tau-1]} \hat{\varphi}\hat{\vartheta}'_j \hat{\alpha}_j^{\iota'}} \breve{r}_{\iota'}$, where $\breve{r}_{\iota'} = (\breve{g}^{\breve{\omega}})^{\alpha^{\iota'}} (\breve{g}^{\breve{\varphi}})^{\sum_{j \in [\tau-1]} \hat{\vartheta}'_j \hat{\alpha}_j^{\iota'}} \breve{r}'_{\iota'}$, and thus $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{2,\tau-1,1}$. On the other hand, if $\hat{\beta} = 1$, i.e., $\Re_{\hat{\beta}} = g^\omega \hat{g}^{\hat{\omega}} \breve{g}^{\breve{\omega}}$, then for all $\iota' \in [2\ell]$, $\widetilde{u}_{\iota'} = (g^\omega)^{\alpha^{\iota'}} \hat{g}^{\hat{\omega}\alpha^{\iota'} + \sum_{j \in [\tau-1]} \hat{\varphi}\hat{\vartheta}'_j \hat{\alpha}_j^{\iota'}} \breve{r}_{\iota'}$, where $\breve{r}_{\iota'} = (\breve{g}^{\breve{\omega}})^{\alpha^{\iota'}} (\breve{g}^{\breve{\varphi}})^{\sum_{j \in [\tau-1]} \hat{\vartheta}'_j \hat{\alpha}_j^{\iota'}} \breve{r}'_{\iota'}$, and hence $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{2,\tau,0}$. This completes the proof of Claim 4.1. □

Lemma 4.4. *For any (possibly computationally unbounded) adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(4)}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function negl.*

Proof. The difference between Hyb_3 and Hyb_4 is that for all $\iota' \in [2\ell]$, $\widetilde{u}_{\iota'}$ is set as $\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}} \hat{g}^{\sum_{j \in [2\ell]} \hat{\vartheta}_j \hat{\alpha}_j^{\iota'}} \breve{r}_{\iota'}$ in Hyb_3, whereas, $\widetilde{u}_{\iota'} = u^{\alpha^{\iota'}} \hat{g}^{\mathcal{R}^{(2\ell)}(\iota')} \breve{r}_{\iota'}$ in Hyb_4, where $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_{2\ell}, \hat{\alpha}_1, \ldots, \hat{\alpha}_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$, and $\mathcal{R}^{(2\ell)} : [2\ell] \rightarrow \mathbb{Z}_{p_2}$ is a truly random function. Hence, Lemma 4.4 follows in a similar fashion as Lemma 3.3 from the core lemma of the Déjà Q framework (Lemma 2.2) stated in Section 2.4 and the fact that $\hat{\alpha}_1 \mod p_2, \ldots, \hat{\alpha}_{2\ell} \mod p_2$ are distinct with overwhelming probability over $\hat{\alpha}_1, \ldots, \hat{\alpha}_{2\ell} \xleftarrow{\$} \mathbb{Z}_n$. □

Lemma 4.5. *For any (possibly computationally unbounded) adversary $\mathcal{A}$, for any security parameter λ, $|\mathsf{Adv}_{\mathcal{A}}^{(4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(5)}(\lambda)| \leq \mathsf{negl}(\lambda)$, for some negligible function negl.*

Proof. The only difference between Hyb_4 and Hyb_5 is that while preparing the challenge, $\mathcal{B}$ selects $\text{EK}_0^* \xleftarrow{\$} \{0,1\}^\lambda$ in Hyb_5 rather than computing $\text{EK}_0^* = \mathcal{H}(T)$, with $T = e(w, \widetilde{u}_{\ell+1})$ as in Hyb_4, where $w \xleftarrow{\$} \mathbb{G}_{p_1 p_2}$, $\widetilde{u}_{\ell+1} = u^{\alpha^{\ell+1}} \hat{g}^{\mathcal{R}^{(2\ell)}(\ell+1)} \breve{r}_{\ell+1}$, and $\mathcal{H} : \mathbb{G}_T \rightarrow \{0,1\}^\lambda$ is uniformly sampled from a pairwise independent hash family $\mathbb{H}_2$. Now, note that in Hyb_4, while creating the queried decryption keys $\text{SK}(\mathscr{A}) = \Big(\mathscr{A}, \Big\{k_\iota, k'_\iota, \{k''_{\iota,\iota'}\}_{\iota' \in \mathbb{U} \setminus \{\rho(\iota)\}}\Big\}_{\iota \in [m]}\Big)$ corresponding to some MAS $\mathscr{A} = (\boldsymbol{M}, \rho) \in \mathfrak{A}$, where $\boldsymbol{M} \in \mathbb{Z}_n^{m \times m'}$ and $\rho : [m] \rightarrow \mathbb{U}$ is the labeling of the rows of $\boldsymbol{M}$ with attributes in $\mathbb{U}$, the components k_ι, k'_ι, and $k''_{\iota,\iota'}$ are formed exactly as in hyb_1 in the following manner:

i) ($\iota \in \mathbb{I}_{\Gamma^*}$)

$$k_\iota = g^{\overrightarrow{M}_\iota \vec{y}' \tau} g^{\breve{\gamma}(\wp'_\iota + \alpha^{\ell+1-\rho(\iota)})} \Big(\prod_{\iota' \in \Gamma^* \setminus \{\rho(\iota)\}} \widetilde{u}_{\iota'}\Big)^{-\wp'_\iota} \Big(\prod_{\iota' \in \Gamma^* \setminus \{\rho(\iota)\}} \widetilde{u}_{\ell+1+\iota'-\rho(\iota)}\Big)^{-1} \breve{r}''_\iota,$$

$$k'_\iota = g^{\wp'_\iota + \alpha^{\ell+1-\rho(\iota)}}, \; k''_{\iota,\iota'} = \widetilde{u}_{\iota'}^{\wp'_\iota} \widetilde{u}_{\ell+1+\iota'-\rho(\iota)} \breve{r}''_{\iota,\iota'},$$

ii) $(\iota \notin \mathbb{I}_{\Gamma^*})$

$$k_\iota = g^{\overrightarrow{M}_\iota \vec{y}'^{\intercal}} g^{\check{\gamma}(\wp'_\iota + \overrightarrow{M}_\iota \vec{d}^{\intercal} \alpha^{\ell+1-\rho(\iota)})} \Big(\prod_{\iota' \in \Gamma^*} \widetilde{u}_{\iota'} \Big)^{-\wp'_\iota} \Big(\prod_{\iota' \in \Gamma^*} \widetilde{u}_{\ell+1+\iota'-\rho(\iota)} \Big)^{-\overrightarrow{M}_\iota \vec{d}^{\intercal}} \widetilde{u}_{\rho(\iota)}^{\wp'_\iota} \check{r}''_\iota,$$

$$k'_\iota = g^{\wp'_\iota + \alpha^{\ell+1-\rho(\iota)} \overrightarrow{M}_\iota \vec{d}^{\intercal}}, \; k''_{\iota,\iota'} = \widetilde{u}_{\iota'}^{\wp'_\iota} \widetilde{u}_{\ell+1+\iota'-\rho(\iota)}^{\overrightarrow{M}_\iota \vec{d}^{\intercal}} \check{r}''_{\iota,\iota'},$$

where $\mathbb{I}_{\Gamma^*} = \{\iota \in [m] \mid \rho(\iota) \in \Gamma^*\}$. Thus, it can be readily seen that the components of the decryption keys provided to $\mathcal{A}$ depend on $\widetilde{u}_1, \ldots, \widetilde{u}_\ell, \widetilde{u}_{\ell+2}, \ldots, \widetilde{u}_{2\ell}$. Also, the master public key MPK provided to $\mathcal{A}$ includes $h_0 = g^{\check{\gamma}}(\prod_{\iota' \in \Gamma^*} \widetilde{u}_{\iota'})^{-1} \check{r}'_0$, $\widetilde{u}_{\iota'}$, for $\iota' \in \mathbb{U}$, where $\Gamma^* \subseteq \mathbb{U} = \{1, \ldots, \ell\}$, and $e(g, \widetilde{u}_{\ell+1}) = e(g, u^{\alpha^{\ell+1}})$. Hence, it is clear that the MPK involves only $\widetilde{u}_1, \ldots, \widetilde{u}_\ell$. Thus, the decryption keys $\mathrm{SK}(\mathscr{A})$'s and the public parameters MPK given to $\mathcal{A}$ only reveal information about $\mathcal{R}^{(2\ell)}(1), \ldots, \mathcal{R}^{(2\ell)}(\ell), \mathcal{R}^{(2\ell)}(\ell+2), \ldots, \mathcal{R}^{(2\ell)}(2\ell)$, and leak no information about $\mathcal{R}^{(2\ell)}(\ell+1)$. Therefore, the quantity, from which EK_0^* is derived in Hyb_4, namely, $T = e(w, \widetilde{u}_{\ell+1}) = e(w, u^{\alpha^{\ell+1}}) \cdot \boxed{e(w, \hat{g}^{\mathcal{R}^{(2\ell)}(\ell+1)})}$ has min-entropy $\mathsf{H}_\infty(T) = \log p_2 = \Theta(\lambda)$ coming from $\mathcal{R}^{(2\ell)}(\ell+1)$. This holds as long as the $\mathbb{G}_{p_2}$-component of w is not the identity element of the group, which happens with probability $1 - \frac{1}{p_2}$. Hence, with overwhelming probability the relation $\lambda = \mathsf{H}_\infty(T) - 2\log\frac{1}{\epsilon} - O(1)$ is satisfied by $\epsilon = 2^{-\Omega(\lambda)}$. Further, note that $\mathcal{H} : \mathbb{G}_T \to \{0,1\}^\lambda$ is sampled from a pairwise independent hash family $\mathbb{H}_2$ uniformly and independently of T. Thus, by the left-over hash lemma (Lemma 2.1) stated in Section 2.3, it follows that the statistical distance $\Delta((\mathcal{H}(T), \mathcal{H}), (U_\lambda, \mathcal{H})) \leq 2^{-\Omega(\lambda)}$, where U_λ is the uniform random variable over $\{0,1\}^\lambda$. Hence, Lemma 4.5 follows. □

5 Our Online-Offline Multi-Input Inner Product Encryption Scheme

In this section we introduce the notion of online-offline multi-input functional encryption (OO-MIFE) and present the first OO-MIFE construction for the bounded-norm multi-input inner product functionality in the private key setting. Our construction is developed in asymmetric bilinear groups of prime order with security under the k-LIN assumption.

5.1 Notion

Definition 5.1 (Multi-Input Bounded-Norm Inner Product Functionality [ARW16]). A multi-input bounded-norm inner product function family $\mathfrak{F}_m^{\ell,\mathsf{B}}$ over $\mathbb{Z}_n$, for some $n, \ell, m, \mathsf{B} \in \mathbb{N}$ with $n \gg m\mathsf{B}$, consists of functions $\mathscr{F}_{\vec{y}^{(1)},\ldots,\vec{y}^{(m)}} : (\mathbb{Z}_n^\ell)^m \to \mathbb{Z}_n$ associated with a tuple of vectors $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$, where $\mathscr{F}_{\vec{y}^{(1)},\ldots,\vec{y}^{(m)}}(\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}) = (\sum_{j \in [m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle) \mod n$, for $\vec{x}^{(1)}, \ldots, \vec{x}^{(m)} \in \mathbb{Z}_n^\ell$ with the norm of component inner products, $|\langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle| \leq \mathsf{B}$, for $j \in [m]$.

In order to simplify naming conventions, we will omit "bounded-norm" for the rest of the paper, but we will always refer to a multi-input inner product functionality with this property.

Definition 5.2 (Online-Offline Private Key Multi-Input Inner Product Encryption: OO-MIPE). A private key online-offline multi-input inner product encryption scheme for an inner product function family $\mathfrak{F}_m^{\ell,\mathsf{B}}$ over $\mathbb{Z}_n$ consists of the following polynomial-time algorithms:

- $\mathsf{OO\text{-}MIPE.Setup}(1^\lambda, \ell, m, \mathsf{B}) \to (\mathrm{PP}, \{\mathrm{ENK}^{(j)}\}_{j \in [m]}, \mathrm{MSK})$: The setup authority takes as input the unary encoded security parameter 1^λ, the length $\ell \in \mathbb{N}$ of vectors, the arity $m \in \mathbb{N}$ of the multi-input inner product functionality, and the bound $\mathsf{B} \in \mathbb{N}$. It generates m private encryption keys $\{\mathrm{ENK}^{(j)}\}_{j \in [m]}$, the master secret key MSK, and the public parameters PP. It publishes PP, provides $\mathrm{ENK}^{(j)}$ to the j^{th} encrypter, for $j \in [m]$, while keeps MSK to itself. Observe that we are considering private key setting and hence PP are not sufficient to encrypt. It merely includes some public informations such as the group description in a bilinear-map-based construction.
- $\mathsf{OO\text{-}MIPE.OfflineKeyGen}(\mathrm{PP}, \mathrm{MSK}) \to \mathrm{IT}_{\mathrm{SK}}$: Taking as input the public parameters PP and the master secret key MSK, the setup authority creates and stores an intermediate decryption key $\mathrm{IT}_{\mathrm{SK}}$.

$\mathsf{OO\text{-}MIPE.OnlineKeyGen}(\text{PP}, \text{MSK}, \text{IT}_{\text{SK}}, (\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})) \rightarrow \text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$: On input the public parameters PP, the master secret key MSK, a fresh intermediate decryption key IT_{SK} generated in the offline phase, together with an m-tuple of vectors $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$, the setup authority provides a decryption key $\text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$ (which includes the tuple $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$ in the clear) to a legitimate decrypter.

$\mathsf{OO\text{-}MIPE.OfflineEncrypt}(\text{PP}, j, \text{ENK}^{(j)}) \rightarrow \text{IT}_{\text{CT}^{(j)}}$: The j^{th} encrypter takes as input the public parameters PP, its index j, together with its private encryption key $\text{ENK}^{(j)}$, prepares an intermediate ciphertext $\text{IT}_{\text{CT}^{(j)}}$, and stores it.

$\mathsf{OO\text{-}MIPE.OnlineEncrypt}(\text{PP}, j, \text{IT}_{\text{CT}^{(j)}}, \vec{x}^{(j)}) \rightarrow \text{CT}^{(j)}(\vec{x}^{(j)})$: The j^{th} encrypter upon input the public parameters PP, its index j, a fresh intermediate ciphertext $\text{IT}_{\text{CT}^{(j)}}$ generated in the offline phase, and a vector $\vec{x}^{(j)} \in \mathbb{Z}_n^\ell$, outputs a ciphertext $\text{CT}^{(j)}(\vec{x}^{(j)})$, which includes the index j.

$\mathsf{OO\text{-}MIPE.Decrypt}(\text{PP}, \text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}), \{\text{CT}^{(j)}(\vec{x}^{(j)})\}_{j\in[m]}) \rightarrow \sum_{j\in[m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle$ or $\perp$: On input the public parameters PP, a decryption key $\text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$ corresponding to a vector tuple $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$, together with a set of m ciphertexts $\{\text{CT}^{(j)}(\vec{x}^{(j)})\}_{j\in[m]}$, where $\text{CT}^{(j)}(\vec{x}^{(j)})$ encrypts the vector $\vec{x}^{(j)} \in \mathbb{Z}_n^\ell$, for $j \in [m]$, a decrypter either outputs the multi-input inner product function value $\sum_{j\in[m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle$ or $\perp$ indicating failure.

The algorithm $\mathsf{OO\text{-}MIPE.Decrypt}$ is deterministic while all the others are randomized.

■ **Correctness**: A $\mathsf{OO\text{-}MIPE}$ scheme is correct if for any security parameter $\lambda \in \mathbb{N}$, any $n, \ell, m, \mathtt{B} \in \mathbb{N}$, any $(\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}), (\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$ with $|\langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle| \leq \mathtt{B}$, for $j \in [m]$, we have

$$\begin{aligned}
&\mathsf{Pr}\Big[(\text{PP}, \{\text{ENK}^{(j)}\}_{j\in[m]}, \text{MSK}) \xleftarrow{\$} \mathsf{OO\text{-}MIPE.Setup}(1^\lambda, \ell, m, \mathtt{B});\ \text{IT}_{\text{SK}} \xleftarrow{\$} \mathsf{OO\text{-}MIPE.OfflineKeyGen}(\text{PP}, \text{MSK});\\
&\quad \text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \xleftarrow{\$} \mathsf{OO\text{-}MIPE.OnlineKeyGen}(\text{PP}, \text{MSK}, \text{IT}_{\text{SK}}, (\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}));\\
&\quad \{\text{IT}_{\text{CT}^{(j)}} \xleftarrow{\$} \mathsf{OO\text{-}MIPE.OfflineEncrypt}(\text{PP}, j, \text{ENK}^{(j)})\}_{j\in[m]};\\
&\quad \{\text{CT}^{(j)}(\vec{x}^{(j)}) \xleftarrow{\$} \mathsf{OO\text{-}MIPE.OnlineEncrypt}(\text{PP}, j, \text{IT}_{\text{CT}^{(j)}}, \vec{x}^{(j)})\}_{j\in[m]}\ :\\
&\quad \mathsf{OO\text{-}MIPE.Decrypt}(\text{PP}, \text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}), \{\text{CT}^{(j)}(\vec{x}^{(j)})\}_{j\in[m]}) = \sum_{j\in[m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle\Big] = 1.
\end{aligned}$$

■ **Security**: The selective indistinguishability-based ($\mathsf{SEL\text{-}IND}$) security notion for a private key $\mathsf{OO\text{-}MIPE}$ scheme in the multi-challenge model is formalized through the following experiment involving a probabilistic polynomial-time adversary $\mathcal{A}$ and a probabilistic polynomial-time challenger $\mathcal{B}$:

- For each $j \in [m]$, $\mathcal{A}$ submits some polynomial number q_j of pairs of vectors $\{(\vec{x}^{(j,t_j,0)}, \vec{x}^{(j,t_j,1)})\}_{t_j\in[q_j]}$.
- $\mathcal{B}$ generates $(\text{PP}, \{\text{ENK}^{(j)}\}_{j\in[m]}, \text{MSK}) \xleftarrow{\$} \mathsf{OO\text{-}MIPE.Setup}(1^\lambda, \ell, m, \mathtt{B})$ and provides PP to $\mathcal{A}$.
- $\mathcal{B}$ selects a bit $\beta \xleftarrow{\$} \{0,1\}$ and creates $\text{CT}^{*(j,t_j)} \xleftarrow{\$} \mathsf{OO\text{-}MIPE.OnlineEncrypt}(\text{PP}, j, \mathsf{OO\text{-}MIPE.OfflineEncrypt}(\text{PP}, j, \text{ENK}^{(j)}), \vec{x}^{(j,t_j,\beta)})$, for $j \in [m], t_j \in [q_j]$. $\mathcal{B}$ gives $\{\text{CT}^{*(j,t_j)}\}_{j\in[m],t_j\in[q_j]}$ to $\mathcal{A}$.
- $\mathcal{A}$ may adaptively make any polynomial number of decryption key queries. In response to a decryption key query of $\mathcal{A}$ corresponding to some m-tuple of vectors $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$ subject to the restriction that for any $(t_1, \ldots, t_m) \in [q_1] \times \ldots \times [q_m]$, $\sum_{j\in[m]} \langle \vec{x}^{(j,t_j,0)}, \vec{y}^{(j)} \rangle = \sum_{j\in[m]} \langle \vec{x}^{(j,t_j,1)}, \vec{y}^{(j)} \rangle$, $\mathcal{B}$ forms the decryption key $\text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \xleftarrow{\$} \mathsf{OO\text{-}MIPE.OnlineKeyGen}(\text{PP}, \text{MSK}, \mathsf{OO\text{-}MIPE.OfflineKeyGen}(\text{PP}, \text{MSK}), (\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}))$ and hands the decryption key $\text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$ to $\mathcal{A}$.
- $\mathcal{A}$ eventually outputs a guess bit $\beta' \in \{0,1\}$.

A private key $\mathsf{OO\text{-}MIPE}$ scheme is said to be $\mathsf{SEL\text{-}IND}$ secure if for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OO\text{-}MIPE,SEL\text{-}IND}}(\lambda) = \big|\mathsf{Pr}[\beta = \beta'] - 1/2\big| \leq \mathsf{negl}(\lambda),$$

for some negligible function negl.

Remark 5.1. As observed in [BSW11, O'N10], for single-input FE, e.g., NIPPE or ABE, the single and multi-challenge security models are equivalent through a standard hybrid argument. However, as pointed out in [GGG+14], such a hybrid argument does not work in case of MIFE. In this work, we consider the stronger of the two models, i.e., the multi-challenge security [GGG+14] for OO-MIPE.

5.2 Construction

OO-MIPE.Setup$(1^\lambda, \ell, m, \mathtt{B}) \to (\text{PP}, \{\text{ENK}^{(j)}\}_{j\in[m]}, \text{MSK})$: The setup authority takes as input the unary encoded security parameter 1^λ, the length ℓ of vectors, the arity m of the multi-input inner product function, and the bound $\mathtt{B}$. It proceeds as follows:

1. It first generates $(n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \mathtt{asymmetric}, \mathtt{prime})$ such that $n \gg m\mathtt{B}$.
2. Next it selects $\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(m)} \xleftarrow{\$} \mathbb{Z}_n^{(k+1)\times k}$, $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(m)} \xleftarrow{\$} \mathbb{Z}_n^{\ell\times(k+1)}$, $\boldsymbol{N}^{(1)}, \ldots, \boldsymbol{N}^{(m)} \xleftarrow{\$} \mathbb{Z}_n^{k\times(k+1)}$, $\vec{f}^{(1)}, \ldots, \vec{f}^{(m)} \xleftarrow{\$} \mathbb{Z}_n^k$, for some appropriate $k \in \mathbb{N}$, $g_1 \xleftarrow{\$} \mathbb{G}_1$, and $g_2 \xleftarrow{\$} \mathbb{G}_2$.
3. Then, it computes $\boldsymbol{A}_1^{(j)} = g_1^{\boldsymbol{A}^{(j)}}$, $\boldsymbol{D}_1^{(j)} = g_1^{\boldsymbol{W}^{(j)}\boldsymbol{A}^{(j)}}$, $\boldsymbol{F}_1^{(j)} = g_1^{\boldsymbol{N}^{(j)}\boldsymbol{A}^{(j)}}$, for $j \in [m]$, and $G = e(g_1, g_2)$.
4. It sets the public parameters $\text{PP} = ((n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, G, \{\boldsymbol{A}_1^{(j)}, \boldsymbol{D}_1^{(j)}, \boldsymbol{F}_1^{(j)}\}_{j\in[m]})$, the encryption keys $\text{ENK}^{(j)} = \vec{f}^{(j)}$, for $j \in [m]$, and master secret key $\text{MSK} = (\{\boldsymbol{W}^{(j)}, \boldsymbol{N}^{(j)}\}_{j\in[m]}, \sum_{j\in[m]} \vec{f}^{(j)})$.

It publishes PP, provides $\text{ENK}^{(j)}$ to the j^{th} encrypter, for $j \in [m]$, while keeps MSK to itself.

OO-MIPE.OfflineKeyGen$(\text{PP}, \text{MSK}) \to \text{IT}_{\text{SK}}$: Taking as input the public parameters $\text{PP} = ((n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, G, \{\boldsymbol{A}_1^{(j)}, \boldsymbol{D}_1^{(j)}, \boldsymbol{F}_1^{(j)}\}_{j\in[m]})$ and the master secret key $\text{MSK} = (\{\boldsymbol{W}^{(j)}, \boldsymbol{N}^{(j)}\}_{j\in[m]}, \sum_{j\in[m]} \vec{f}^{(j)})$, the setup authority executes the following steps:

1. It first picks $\vec{z}^{(1)}, \ldots, \vec{z}^{(m)} \xleftarrow{\$} \mathbb{Z}_n^{k+1}$, and $\vec{h} \xleftarrow{\$} \mathbb{Z}_n^k$.
2. After that, it computes $\widetilde{\boldsymbol{k}}^{(1,j)} = g_2^{\vec{z}^{(j)} + \vec{h}\boldsymbol{N}^{(j)}}$, for $j \in [m]$, $\widetilde{\boldsymbol{k}}^{(2)} = g_2^{\vec{h}}$, and $\widetilde{K}^{(3)} = G^{\langle \sum_{j\in[m]} \vec{f}^{(j)}, \vec{h}\rangle}$.
3. It stores the intermediate decryption key $\text{IT}_{\text{SK}} = (\{\widetilde{\boldsymbol{k}}^{(1,j)}\}_{j\in[m]}, \widetilde{\boldsymbol{k}}^{(2)}, \widetilde{K}^{(3)}, \{\vec{z}^{(j)}\}_{j\in[m]})$.

OO-MIPE.OnlineKeyGen$(\text{PP}, \text{MSK}, \text{IT}_{\text{SK}}, (\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})) \to \text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$:
On input the public parameters $\text{PP} = ((n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, G, \{\boldsymbol{A}_1^{(j)}, \boldsymbol{D}_1^{(j)}, \boldsymbol{F}_1^{(j)}\}_{j\in[m]})$, the master secret key $\text{MSK} = (\{\boldsymbol{W}^{(j)}, \boldsymbol{N}^{(j)}\}_{j\in[m]}, \sum_{j\in[m]} \vec{f}^{(j)})$, a fresh intermediate decryption key $\text{IT}_{\text{SK}} = (\{\widetilde{\boldsymbol{k}}^{(1,j)}\}_{j\in[m]}, \widetilde{\boldsymbol{k}}^{(2)}, \widetilde{K}^{(3)}, \{\vec{z}^{(j)}\}_{j\in[m]})$ formed in the offline phase, along with an m-tuple of vectors $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$, the setup authority performs the following steps:

1. It sets $\boldsymbol{k}^{(1,j)} = \widetilde{\boldsymbol{k}}^{(1,j)}$, for $j \in [m]$, $\boldsymbol{k}^{(2)} = \widetilde{\boldsymbol{k}}^{(2)}$, $K^{(3)} = \widetilde{K}^{(3)}$, and $\vec{k}^{(4,j)} = \vec{y}^{(j)}\boldsymbol{W}^{(j)} - \vec{z}^{(j)}$, for $j \in [m]$.
2. It gives a legitimate decrypter with the decryption key $\text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) = ((\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}), \{\boldsymbol{k}^{(1,j)}\}_{j\in[m]}, \boldsymbol{k}^{(2)}, K^{(3)}, \{\vec{k}^{(4,j)}\}_{j\in[m]})$.

OO-MIPE.OfflineEncrypt$(\text{PP}, j, \text{ENK}^{(j)}) \to \text{IT}_{\text{CT}^{(j)}}$: The j^{th} encrypter takes as input the public parameters $\text{PP} = ((n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, G, \{\boldsymbol{A}_1^{(j)}, \boldsymbol{D}_1^{(j)}, \boldsymbol{F}_1^{(j)}\}_{j\in[m]})$, its index $j \in [m]$, along with its private encryption key $\text{ENK}^{(j)} = \vec{f}^{(j)}$. It operates as follows:

1. It first selects $\vec{u}^{(j)} \xleftarrow{\$} \mathbb{Z}_n^\ell$ and $\vec{s}^{(j)} \xleftarrow{\$} \mathbb{Z}_n^k$.
2. Next, it computes $\widetilde{\boldsymbol{c}}^{(1,j)} = g_1^{\vec{u}^{(j)}}(\boldsymbol{D}_1^{(j)})^{\vec{s}^{(j)}} = g_1^{\vec{u}^{(j)} + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{W}^{(j)\intercal}}$, $\widetilde{\boldsymbol{c}}^{(2,j)} = g_1^{\vec{f}^{(j)}}(\boldsymbol{F}_1^{(j)})^{\vec{s}^{(j)}} = g_1^{\vec{f}^{(j)} + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{N}^{(j)\intercal}}$, and $\widetilde{\boldsymbol{c}}^{(3,j)} = (\boldsymbol{A}_1^{(j)})^{\vec{s}^{(j)}} = g_1^{\vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}}$.
3. It stores the intermediate ciphertext $\text{IT}_{\text{CT}^{(j)}} = (\widetilde{\boldsymbol{c}}^{(1,j)}, \widetilde{\boldsymbol{c}}^{(2,j)}, \widetilde{\boldsymbol{c}}^{(3,j)}, \vec{u}^{(j)})$.

OO-MIPE.OnlineEncrypt$(\text{PP}, j, \text{IT}_{\text{CT}^{(j)}}, \vec{x}^{(j)}) \to \text{CT}^{(j)}(\vec{x}^{(j)})$: An encrypter upon input the public parameters $\text{PP} = ((n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, G, \{\boldsymbol{A}_1^{(j)}, \boldsymbol{D}_1^{(j)}, \boldsymbol{F}_1^{(j)}\}_{j\in[m]})$, its own index $j \in [m]$, a fresh intermediate ciphertext $\text{IT}_{\text{CT}^{(j)}} = (\widetilde{\boldsymbol{c}}^{(1,j)}, \widetilde{\boldsymbol{c}}^{(2,j)}, \widetilde{\boldsymbol{c}}^{(3,j)}, \vec{u}^{(j)})$ created in the offline phase, and a vector $\vec{x}^{(j)} \in \mathbb{Z}_n^\ell$, proceeds as follows:

1. It sets $\boldsymbol{c}^{(1,j)} = \widetilde{\boldsymbol{c}}^{(1,j)}$, $\boldsymbol{c}^{(2,j)} = \widetilde{\boldsymbol{c}}^{(2,j)}, \boldsymbol{c}^{(3,j)} = \widetilde{\boldsymbol{c}}^{(3,j)}$, and $\vec{c}^{(4,j)} = \vec{x}^{(j)} - \vec{u}^{(j)}$.
2. It outputs the ciphertext $\mathrm{CT}^{(j)}(\vec{x}^{(j)}) = (j, \boldsymbol{c}^{(1,j)}, \boldsymbol{c}^{(2,j)}, \boldsymbol{c}^{(3,j)}, \vec{c}^{(4,j)})$.

$\mathsf{OO\text{-}MIPE.Decrypt}(\mathrm{PP}, \mathrm{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}), \{\mathrm{CT}^{(j)}(\vec{x}^{(j)})\}_{j\in[m]}) \rightarrow \sum_{j\in[m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle$ or $\perp$: A decrypter takes as input the public parameters $\mathrm{PP} = ((n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, G, \{\boldsymbol{A}_1^{(j)}, \boldsymbol{D}_1^{(j)}, \boldsymbol{F}_1^{(j)}\}_{j\in[m]})$, a decryption key $\mathrm{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) = ((\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})\{\boldsymbol{k}^{(1,j)}\}_{j\in[m]}, \boldsymbol{k}^{(2)}, K^{(3)}, \{\vec{k}^{(4,j)}\}_{j\in[m]})$ corresponding to an m-tuple of vectors $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$, and m ciphertexts $\{\mathrm{CT}^{(j)}(\vec{x}^{(j)}) = (j, \boldsymbol{c}^{(1,j)}, \boldsymbol{c}^{(2,j)}, \boldsymbol{c}^{(3,j)}, \vec{c}^{(4,j)})\}_{j\in[m]}$. It executes the following:

1. It first computes
$$\widetilde{T} = \prod_{j\in[m]} \left[\frac{E_\ell(\boldsymbol{c}^{(1,j)} \boxed{g_1^{\vec{c}^{(4,j)}}}, g_2^{\vec{y}^{(j)}}) E_k(\boldsymbol{c}^{(2,j)}, \boldsymbol{k}^{(2)})}{E_{k+1}(\boldsymbol{c}^{(3,j)}, \boldsymbol{k}^{(1,j)} \boxed{g_2^{\vec{k}^{(4,j)}}})} \right]. \tag{5.1}$$
2. Next, it computes $T = \dfrac{\widetilde{T}}{K^{(3)}}$.
3. Finally, it attempts to determine a value $\psi \in \mathbb{Z}_n$ such that $T = G^\psi$, by exhaustively searching a polynomial size range of possible values and outputs ψ, if successful. Otherwise, it outputs $\perp$ indicating failure.

■ **Correctness**: The correctness of the proposed OO-MIPE construction can be verified as follows: Consider any decryption key

$$\mathrm{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) = \Big((\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}), \{\boldsymbol{k}^{(1,j)} = g_2^{\vec{z}^{(j)} + \vec{h}\boldsymbol{N}^{(j)}}\}_{j\in[m]}, \boldsymbol{k}^{(2)} = g_2^{\vec{h}}, K^{(3)} = e(g_1, g_2)^{\langle \sum_{j\in[m]} \vec{f}^{(j)}, \vec{h} \rangle}, \{\vec{k}^{(4,j)} = \vec{y}^{(j)}\boldsymbol{W}^{(j)} - \vec{z}^{(j)}\}_{j\in[m]} \Big)$$

corresponding to the m-tuple of vectors $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$, and m ciphertexts

$$\Big\{ \mathrm{CT}^{(j)}(\vec{x}^{(j)}) = \Big(j, \boldsymbol{c}^{(1,j)} = g_1^{\vec{u}^{(j)} + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{W}^{(j)\intercal}}, \boldsymbol{c}^{(2,j)} = g_1^{\vec{f}^{(j)} + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{N}^{(j)\intercal}}, \boldsymbol{c}^{(3,j)} = g_1^{\vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}}, \vec{c}^{(4,j)} = \vec{x}^{(j)} - \vec{u}^{(j)} \Big) \Big\}_{j\in[m]}$$

encrypting the vectors $\vec{x}^{(1)}, \ldots, \vec{x}^{(m)} \in \mathbb{Z}_n^\ell$. We have,

$$\begin{aligned}
\widetilde{T} &= \prod_{j\in[m]} \left[\frac{E_\ell(\boldsymbol{c}^{(1,j)} g_1^{\vec{c}^{(4,j)}}, g_2^{\vec{y}^{(j)}}) E_k(\boldsymbol{c}^{(2,j)}, \boldsymbol{k}^{(2)})}{E_{k+1}(\boldsymbol{c}^{(3,j)}, \boldsymbol{k}^{(1,j)} g_2^{\vec{k}^{(4,j)}})} \right] \\
&= \prod_{j\in[m]} \left[\frac{E_\ell(g_1^{\vec{u}^{(j)} + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{W}^{(j)\intercal}} g_1^{\vec{x}^{(j)} - \vec{u}^{(j)}}, g_2^{\vec{y}^{(j)}}) E_k(g_1^{\vec{f}^{(j)} + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{N}^{(j)\intercal}}, g_2^{\vec{h}})}{E_{k+1}(g_1^{\vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}}, g_2^{\vec{z}^{(j)} + \vec{h}\boldsymbol{N}^{(j)}} g_2^{\vec{y}^{(j)}\boldsymbol{W}^{(j)} - \vec{z}^{(j)}})} \right] \\
&= \prod_{j\in[m]} \left[\frac{e(g_1, g_2)^{\langle \vec{x}^{(j)} + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{W}^{(j)\intercal}, \vec{y}^{(j)} \rangle} e(g_1, g_2)^{\langle \vec{f}^{(j)} + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{N}^{(j)\intercal}, \vec{h} \rangle}}{e(g_1, g_2)^{\langle \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}, \vec{y}^{(j)}\boldsymbol{W}^{(j)} + \vec{h}\boldsymbol{N}^{(j)} \rangle}} \right] \\
&= \prod_{j\in[m]} \left[\frac{e(g_1, g_2)^{\langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle + \langle \vec{f}^{(j)}, \vec{h} \rangle + \vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}(\vec{y}^{(j)}\boldsymbol{W}^{(j)} + \vec{h}\boldsymbol{N}^{(j)})^\intercal}}{e(g_1, g_2)^{\vec{s}^{(j)}\boldsymbol{A}^{(j)\intercal}(\vec{y}^{(j)}\boldsymbol{W}^{(j)} + \vec{h}\boldsymbol{N}^{(j)})^\intercal}} \right] \\
&= e(g_1, g_2)^{\sum_{j\in[m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle} e(g_1, g_2)^{\langle \sum_{j\in[m]} \vec{f}^{(j)}, \vec{h} \rangle}.
\end{aligned}$$

Therefore,

$$T = \frac{\widetilde{T}}{K^{(3)}} = \frac{e(g_1, g_2)^{\sum_{j\in[m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle} e(g_1, g_2)^{\langle \sum_{j\in[m]} \vec{f}^{(j)}, \vec{h} \rangle}}{e(g_1, g_2)^{\langle \sum_{j\in[m]} \vec{f}^{(j)}, \vec{h} \rangle}} = e(g_1, g_2)^{\sum_{j\in[m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle}.$$

Hence, if $\sum_{j\in[m]} \langle \vec{x}^{(j)}, \vec{y}^{(j)} \rangle$ is contained within the specified polynomial range of values searched by OO-MIPE.Decrypt, then the algorithm correctly outputs the functional value.

5.3 Security Analysis

Theorem 5.1 (Security of Our OO-MIPE Scheme). *The* OO-MIPE *scheme proposed in Section 5.2 is* SEL-IND *secure as per the security model described in Section 5.1 under the* k-LIN *assumption.*

Proof. In order to prove Theorem 5.1, we demonstrate that any probabilistic polynomial-time adversary $\mathcal{A}$ with a non-negligible advantage in breaking the SEL-IND security (according to the security model introduced in Section 5.1) of the OO-MIPE scheme proposed in Section 5.2, which we would represent as $\mathbb{S}_{\text{OO-MIPE}} = (\text{OO-MIPE.Setup}, \text{OO-MIPE.OfflineKeyGen}, \text{OO-MIPE.OnlineKeyGen}, \text{OO-MIPE.OfflineEncrypt}, \text{OO-MIPE.OnlineEncrypt}, \text{OO-MIPE.Decrypt})$, can be employed to build a probabilistic polynomial-time adversary $\mathcal{B}$ that breaks the SEL-IND security (as per the security model of [ARW16] described in Fig. 5.1) of the private key multi-input inner product encryption (MIPE) construction of [ARW16], which we represent as $\mathbb{S}_{\text{MIPE}} = (\text{MIPE.Setup}, \text{MIPE.KeyGen}, \text{MIPE.Encrypt}, \text{MIPE.Decrypt})$. We note that the OO-MIPE.Setup algorithm is exactly identical to MIPE.Setup, while the OO-MIPE.Decrypt algorithm is the augmentation of MIPE.Decrypt by adding the terms framed by boxes in Eq. (5.1). The description of the algorithms MIPE.KeyGen and MIPE.Encrypt are provided in the relevant places below in this proof. The

The selective indistinguishability-based (SEL-IND) security notion for a private key MIPE scheme in the multi-challenge model is formalized through the following experiment involving a probabilistic polynomial-time adversary $\mathcal{B}$ and a probabilistic polynomial-time challenger $\mathcal{C}$:

- For each $j \in [m]$, $\mathcal{B}$ submits some polynomial number q_j of pairs of vectors $\{(\vec{x}^{\prime(j,t_j,0)}, \vec{x}^{\prime(j,t_j,1)})\}_{t_j\in[q_j]}$.
- $\mathcal{C}$ generates $(\text{PP}', \{\text{ENK}'^{(j)}\}_{j\in[m]}, \text{MSK}') \xleftarrow{\$} \text{MIPE.Setup}(1^\lambda, \ell, m, \mathsf{B})$ and provides PP' to $\mathcal{B}$.
- $\mathcal{C}$ selects a bit $\beta \xleftarrow{\$} \{0,1\}$ and creates $\text{CT}'^{*(j,t_j)} \xleftarrow{\$} \text{MIPE.Encrypt}(\text{PP}', j, \text{ENK}'^{(j)}, \vec{x}^{\prime(j,t_j,\beta)})$, for $j \in [m], t_j \in [q_j]$. $\mathcal{C}$ gives $\{\text{CT}'^{*(j,t_j)}\}_{j\in[m],t_j\in[q_j]}$ to $\mathcal{B}$.
- $\mathcal{B}$ may adaptively make any polynomial number of decryption key queries. In response to a decryption key query of $\mathcal{B}$ corresponding to some m-tuple of vectors $(\vec{y}^{\prime(1)}, \ldots, \vec{y}^{\prime(m)}) \in (\mathbb{Z}_n^\ell)^m$ subject to the restriction that for any $(t_1, \ldots, t_m) \in [q_1] \times \ldots \times [q_m]$, $\sum_{j\in[m]} \langle \vec{x}^{\prime(j,t_j,0)}, \vec{y}^{\prime(j)} \rangle = \sum_{j\in[m]} \langle \vec{x}^{\prime(j,t_j,1)}, \vec{y}^{\prime(j)} \rangle$, $\mathcal{C}$ forms the decryption key $\text{SK}'(\vec{y}^{\prime(1)}, \ldots, \vec{y}^{\prime(m)}) \xleftarrow{\$} \text{MIPE.KeyGen}(\text{PP}', \text{MSK}', (\vec{y}^{\prime(1)}, \ldots, \vec{y}^{\prime(m)}))$ and hands the decryption key $\text{SK}'(\vec{y}^{\prime(1)}, \ldots, \vec{y}^{\prime(m)})$ to $\mathcal{B}$.
- $\mathcal{B}$ eventually outputs a guess bit $\beta' \in \{0,1\}$.

A private key MIPE scheme is said to be SEL-IND secure if for any probabilistic polynomial-time adversary $\mathcal{B}$, for any security parameter λ, the advantage of $\mathcal{B}$ in the above experiment,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{MIPE,SEL-IND}}(\lambda) = \left|\Pr[\beta = \beta'] - 1/2\right| \leq \mathsf{negl}(\lambda),$$

for some negligible function negl.

Fig. 5.1. SEL-IND Security Model for MIPE

description of $\mathcal{B}$ is presented below:

- $\mathcal{B}$ initializes $\mathcal{A}$ on input 1^λ and receives back the pairs of vectors $\{(\vec{x}^{(j,t_j,0)}, \vec{x}^{(j,t_j,1)})\}_{t_j\in[q_j]}$ for $j \in [m]$ from $\mathcal{A}$. Then, for each $j \in [m]$, $\mathcal{B}$ picks $\vec{a}^{(j,t_j)} \xleftarrow{\$} \mathbb{Z}_n^\ell$, for $t_j \in [q_j]$. For each $j \in [m]$, $\mathcal{B}$ submits the pairs of vectors $\{(\vec{x}^{\prime(j,t_j,0)} = \vec{x}^{(j,t_j,0)} - \vec{a}^{(j,t_j)}, \vec{x}^{\prime(j,t_j,1)} = \vec{x}^{(j,t_j,1)} - \vec{a}^{(j,t_j)})\}_{t_j\in[q_j]}$ to its SEL-IND challenger $\mathcal{C}$ for $\mathbb{S}_{\text{MIPE}}$.
- $\mathcal{B}$ obtains the public parameters $\text{PP}' = ((n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, e(g_1, g_2), \{\boldsymbol{A}_1^{(j)}, \boldsymbol{D}_1^{(j)}, \boldsymbol{F}_1^{(j)}\}_{j\in[m]})$ from its SEL-IND challenger $\mathcal{C}$, where PP' is generated by $\mathcal{C}$ by running $\text{MIPE.Setup}(1^\lambda, \ell, m, \mathsf{B})$. Here, $(n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda, \texttt{asymmetric}, \texttt{prime})$, $g_1 \xleftarrow{\$} \mathbb{G}_1$, $g_2 \xleftarrow{\$} \mathbb{G}_2$, and for all $j \in [m]$, $\boldsymbol{A}_1^{(j)} = g_1^{\boldsymbol{A}^{(j)}}$, $\boldsymbol{D}_1^{(j)} = g_1^{\boldsymbol{W}^{(j)}}$, $\boldsymbol{F}_1^{(j)} = g_1^{\boldsymbol{N}^{(j)}}$ with $\boldsymbol{A}^{(j)} \xleftarrow{\$} \mathbb{Z}_n^{(k+1)\times k}$, $\boldsymbol{W}^{(j)} \xleftarrow{\$} \mathbb{Z}_n^{\ell\times(k+1)}$, $\boldsymbol{N}^{(j)} \xleftarrow{\$} \mathbb{Z}_n^{k\times(k+1)}$. Clearly, PP'

has the same distribution as the public parameters PP in $\mathbb{S}_{\mathsf{OO\text{-}MIPE}}$. $\mathcal{B}$ provides the public parameters $\text{PP} = \text{PP}'$ to $\mathcal{A}$.

- Next, $\mathcal{B}$ receives the set of challenge ciphertexts $\{\text{CT}'^{*(j,t_j)}\}_{j\in[m],t_j\in[q_j]}$ from its SEL-IND challenger $\mathcal{C}$ such that for each $j \in [m]$, $t_j \in [q_j]$, $\text{CT}'^{*(j,t_j)} = (j, \boldsymbol{c}'^{(1,j,t_j)} = g_1^{\vec{x}'^{(j,t_j,\beta)}+\vec{s}^{(j,t_j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{W}^{(j)\intercal}}, \boldsymbol{c}'^{(2,j,t_j)} = g_1^{\vec{f}^{(j)}+\vec{s}^{(j,t_j)}\boldsymbol{A}^{(j)\intercal}\boldsymbol{N}^{(j)\intercal}}, \boldsymbol{c}'^{(3,j,t_j)} = g_1^{\vec{s}^{(j,t_j)}\boldsymbol{A}^{(j)\intercal}})$ $\xleftarrow{\$}$ $\mathsf{MIPE.Encrypt}(\text{PP}', j, \text{ENK}'^{(j)} = \vec{f}^{(j)}, \vec{x}'^{(j,t_j,\beta)})$, where $\beta \xleftarrow{\$} \{0,1\}$ is the random bit selected by $\mathcal{C}$. Here, $\{\vec{f}^{(j)}\}_{j\in[m]}$ is the set of vectors uniformly sampled from $\mathbb{Z}_n^k$ by the MIPE.Setup algorithm and for each $j \in [m]$, $t_j \in [q_j]$, $\vec{s}^{(j,t_j)} \xleftarrow{\$} \mathbb{Z}_n^k$ is the randomness chosen by the MIPE.Encrypt algorithm. For each $j \in [m]$, $t_j \in [q_j]$, $\mathcal{B}$ sets the challenge ciphertext $\text{CT}^{*(j,t_j)} = (j, \boldsymbol{c}^{(1,j,t_j)} = \boldsymbol{c}'^{(1,j,t_j)}, \boldsymbol{c}^{(2,j,t_j)} = \boldsymbol{c}'^{(2,j,t_j)}, \boldsymbol{c}^{(3,j,t_j)} = \boldsymbol{c}'^{(3,j,t_j)}, \vec{c}^{(4,j,t_j)} = \vec{a}^{(j,t_j)})$. $\mathcal{B}$ gives the set of ciphertexts $\{\text{CT}^{*(j,t_j)}\}_{j\in[m],t_j\in[q_j]}$ to $\mathcal{A}$.

- In response to a decryption key query of $\mathcal{A}$ corresponding to an m-tuple of vectors $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) \in (\mathbb{Z}_n^\ell)^m$ subject to the restriction that for any $(t_1, \ldots, t_m) \in [q_1] \times \ldots \times [q_m]$, $\sum\limits_{j\in[m]} \langle \vec{x}^{(j,t_j,0)}, \vec{y}^{(j)} \rangle = \sum\limits_{j\in[m]} \langle \vec{x}^{(j,t_j,1)}, \vec{y}^{(j)} \rangle$, $\mathcal{B}$ queries a decryption key to $\mathcal{C}$ corresponding to the tuple $(\vec{y}'^{(1)}, \ldots, \vec{y}'^{(m)}) = (\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$ and receives back a decryption key $\text{SK}'(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) = \Big((\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}), \{\boldsymbol{k}'^{(1,j)} = g_2^{\vec{y}^{(j)}\boldsymbol{W}^{(j)}+\vec{h}\boldsymbol{N}^{(j)}}\}_{j\in[m]}, \boldsymbol{k}'^{(2)} = g_2^{\vec{h}}, K'^{(3)} = e(g_1, g_2)^{\langle \sum\limits_{j\in[m]} \vec{f}^{(j)}, \vec{h} \rangle}\Big)$ $\xleftarrow{\$}$ $\mathsf{MIPE.KeyGen}\Big(\text{PP}', \text{MSK}' = (\{\boldsymbol{W}^{(j)}, \boldsymbol{N}^{(j)}\}_{j\in[m]}, \sum\limits_{j\in[m]} \vec{f}^{(j)}), (\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})\Big)$ from $\mathcal{C}$. Here, $\vec{h} \xleftarrow{\$} \mathbb{Z}_n^k$ is a random vector selected by the MIPE.KeyGen algorithm. Observe that $(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)})$ constitutes a valid decryption key query in the SEL-IND security experiment for $\mathbb{S}_{\mathsf{MIPE}}$ since for any $(t_1, \ldots, t_m) \in [q_1] \times \ldots \times [q_m]$, we have

$$\begin{aligned}
\sum_{j\in[m]} \langle \vec{x}'^{(j,t_j,0)}, \vec{y}^{(j)} \rangle &= \sum_{j\in[m]} \langle \vec{x}^{(j,t_j,0)} - \vec{a}^{(j,t_j)}, \vec{y}^{(j)} \rangle \\
&= \sum_{j\in[m]} \langle \vec{x}^{(j,t_j,0)}, \vec{y}^{(j)} \rangle - \sum_{j\in[m]} \langle \vec{a}^{(j,t_j)}, \vec{y}^{(j)} \rangle \\
&= \sum_{j\in[m]} \langle \vec{x}^{(j,t_j,1)}, \vec{y}^{(j)} \rangle - \sum_{j\in[m]} \langle \vec{a}^{(j,t_j)}, \vec{y}^{(j)} \rangle \\
&= \sum_{j\in[m]} \langle \vec{x}^{(j,t_j,1)} - \vec{a}^{(j,t_j)}, \vec{y}^{(j)} \rangle \\
&= \sum_{j\in[m]} \langle \vec{x}'^{(j,t_j,1)}, \vec{y}^{(j)} \rangle.
\end{aligned}$$

  $\mathcal{B}$ selects $\vec{b}^{(1)}, \ldots, \vec{b}^{(m)} \xleftarrow{\$} \mathbb{Z}_n^{k+1}$, and provides $\mathcal{A}$ with the decryption key $\text{SK}(\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}) = \Big((\vec{y}^{(1)}, \ldots, \vec{y}^{(m)}), \{\boldsymbol{k}^{(1,j)} = \boldsymbol{k}'^{(1,j)} g_2^{-\vec{b}^{(j)}}\}_{j\in[m]}, \boldsymbol{k}^{(2)} = \boldsymbol{k}'^{(2)}, K^{(3)} = K'^{(3)}, \{\vec{k}^{(4,j)} = \vec{b}^{(j)}\}_{j\in[m]}\Big)$.

- $\mathcal{A}$ eventually outputs a guess bit $\beta' \in \{0,1\}$. $\mathcal{B}$ also outputs β' as its guess bit in its SEL-IND experiment for $\mathbb{S}_{\mathsf{MIPE}}$.

Note that all the challenge ciphertexts and decryption keys queried by $\mathcal{A}$ are correctly simulated by $\mathcal{B}$. This can be readily verified by checking the decryption equation of $\mathbb{S}_{\mathsf{OO\text{-}MIPE}}$. Moreover, the ciphertexts and decryption keys are randomized by $\mathcal{B}$ to have the proper distributions. Also, it is immediate that for all $j \in [m]$, $t_j \in [q_j]$, $\beta \in \{0,1\}$, if $\text{CT}'^{*(j,t_j)}$ encrypts $\vec{x}'^{(j,t_j,\beta)}$, then $\text{CT}^{*(j,t_j)}$ encrypts $\vec{x}^{(j,t_j,\beta)}$ as well. Hence, it follows that the simulation of the SEL-IND security experiment for $\mathbb{S}_{\mathsf{OO\text{-}MIPE}}$ by $\mathcal{B}$ is perfect. Further, if $\mathcal{A}$ correctly guesses the challenge bit in this simulated SEL-IND experiment for $\mathbb{S}_{\mathsf{OO\text{-}MIPE}}$, then $\mathcal{B}$ correctly guesses the challenge bit in its SEL-IND security experiment for $\mathbb{S}_{\mathsf{MIPE}}$. Hence, Theorem 5.1 follows as $\mathbb{S}_{\mathsf{MIPE}}$ is SEL-IND secure under the k-LIN assumption. □

6 Conclusion

In this paper, we have employed the extended Déjà Q framework [CM14, Wee16] for constructing NIPPE and ABE systems with succinct ciphertexts and decryption keys without relying on q-type complexity assumptions. We have also defined and constructed online-offline MIPE scheme. Although, the Déjà Q framework has already proven itself as a strong technique for avoiding the need of q-type assumptions in constructing various cryptographic primitives with very short parameters, as pointed out in [CM14], the current version of this technique is not readily amenable to use in prime order bilinear group setting. Developing a prime order variant of this classic framework is undoubtedly a fascinating research problem as it can lead to further efficiency improvements. On the other hand, in view of the emerging trend of performing complex applications in small device and the wide range of potentials of MIFE, it is instructive to make further progress in the field of online-offline MIFE, both towards supporting expressive multi-input functionalities of practical significance as well as achieving stronger security guarantees.

References

ABDCP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography–PKC 2015*, pages 733–751. Springer, 2015.

AL10. Nuttapong Attrapadung and Benoît Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In *International Workshop on Public Key Cryptography–PKC 2010*, pages 384–402. Springer, 2010.

ALDP11. Nuttapong Attrapadung, Benoît Libert, and Elie De Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *International Workshop on Public Key Cryptography–PKC 2011*, pages 90–108. Springer, 2011.

ALS15. Shweta Agrawal, Benoit Libert, and Damien Stehle. Fully secure functional encryption for inner products, from standard assumptions. Cryptology ePrint Archive, Report 2015/608, 2015.

AR16. Shweta Agrawal and Alon Rosen. Online-offline functional encryption for bounded collusions. Cryptology ePrint Archive, Report 2016/361, 2016.

ARW16. Michel Abdalla, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. Cryptology ePrint Archive, Report 2016/425, 2016.

Att14. Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Advances in Cryptology–EUROCRYPT 2014*, pages 557–577. Springer, 2014.

Att15. Nuttapong Attrapadung. Dual system encryption framework in prime-order groups. Cryptology ePrint Archive, Report 2015/390, 2015.

BBG05. 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.

Bei96. Amos Beimel. *Secure schemes for secret sharing and key distribution.* Technion-Israel Institute of technology, Faculty of computer science, 1996.

BF01. Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology–CRYPTO 2001*, pages 213–229. Springer, 2001.

BGJS15. Saikrishna Badrinarayanan, Divya Gupta, Abhishek Jain, and Amit Sahai. Multi-input functional encryption for unbounded arity functions. In *Advances in Cryptology–ASIACRYPT 2015*, pages 27–51. Springer, 2015.

BKS15. Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. Cryptology ePrint Archive, Report 2015/158, 2015.

BLR+15. Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Advances in Cryptology–EUROCRYPT 2015*, pages 563–594. Springer, 2015.

Boy08. Xavier Boyen. The uber-assumption family. In *International Conference on Pairing-Based Cryptography–Pairing 2008*, pages 39–56. Springer, 2008.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference–TCC 2011*, pages 253–273. Springer, 2011.

CCL+13. Cheng Chen, Jie Chen, Hoon Wei Lim, Zhenfeng Zhang, Dengguo Feng, San Ling, and Huaxiong Wang. Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In *CryptographersŠ Track at the RSA Conference–CT-RSA 2013*, pages 50–67. Springer, 2013.

CGW15. Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system abe in prime-order groups via predicate encodings. In *Advances in Cryptology–EUROCRYPT 2015*, pages 595–624. Springer, 2015.

Che06. Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *Advances in Cryptology–EUROCRYPT 2006*, pages 1–11. Springer, 2006.

CLR16. Jie Chen, Benoît Libert, and Somindu Ramanna. Non-zero inner product encryption with short ciphertexts and private keys. In *Security and Cryptography for Networks–SCN 2016*, pages 23–41. Springer, 2016.

CM14. Melissa Chase and Sarah Meiklejohn. Déja q: Using dual systems to revisit q-type assumptions. In *Advances in Cryptology–EUROCRYPT 2014*, pages 622–639. Springer, 2014.

CW14. Jie Chen and Hoeteck Wee. Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In *International Conference on Security and Cryptography for Networks–SCN 2014*, pages 277–297. Springer, 2014.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Rafols, and Jorge Villar. An algebraic framework for diffie-hellman assumptions. In *Advances in Cryptology–CRYPTO 2013*, pages 129–147. Springer, 2013.

EMN+09. Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In *International Conference on Information Security Practice and Experience–ISPEC 2009*, pages 13–23. Springer, 2009.

Fre10. David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Advances in Cryptology–EUROCRYPT 2010*, pages 44–61. Springer, 2010.

GGG+14. Shafi Goldwasser, S Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Advances in Cryptology–EUROCRYPT 2014*, pages 578–602. Springer, 2014.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *Advances in Cryptology–CRYPTO 2013*, pages 479–499. Springer, 2013.

GJN15. Vipul Goyal, Aayush Jain, and Adam O' Neill. Multi-input functional encryption with unbounded-message security. Cryptology ePrint Archive, Report 2015/1113, 2015.

GKP+13. Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *Advances in Cryptology–CRYPTO 2013*, pages 536–553. Springer, 2013.

GMC08. Fuchun Guo, Yi Mu, and Zhide Chen. Identity-based online/offline encryption. In *International Conference on Financial Cryptography and Data Security*, pages 247–261. Springer, 2008.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *13th ACM conference on Computer and communications security*, pages 89–98. ACM, 2006.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. *Journal of the ACM (JACM)*, 62(6):45, 2015.

HLR10. Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. Constant size ciphertexts in threshold attribute-based encryption. In *International Workshop on Public Key Cryptography–PKC 2010*, pages 19–34. Springer, 2010.

HW14. Susan Hohenberger and Brent Waters. Online/offline attribute-based encryption. In *International Workshop on Public Key Cryptography–PKC 2014*, pages 293–310. Springer, 2014.

ILL89. Russell Impagliazzo, Leonid A Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Twenty-first annual ACM symposium on Theory of computing*, pages 12–24. ACM, 1989.

LOS+10. Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology–EUROCRYPT 2010*, pages 62–91. Springer, 2010.

LW10. Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *Theory of Cryptography Conference–TCC 2010*, pages 455–479. Springer, 2010.

LW11. Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 568–588. Springer, 2011.

LZ09. Joseph K Liu and Jianying Zhou. An efficient identity-based online/offline encryption scheme. In *International Conference on Applied Cryptography and Network Security–ACNS 2009*, pages 156–167. Springer, 2009.

O'N10. Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010.

OSW07. Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *14th ACM conference on Computer and communications security*, pages 195–203. ACM, 2007.

OT15. Tatsuaki Okamoto and Katsuyuki Takashima. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. *Designs, Codes and Cryptography*, 77(2-3):725–771, 2015.

SHI+12. Yumi Sakemi, Goichiro Hanaoka, Tetsuya Izu, Masahiko Takenaka, and Masaya Yasuda. Solving a discrete logarithm problem with auxiliary input on a 160-bit elliptic curve. In *International Workshop on Public Key Cryptography–PKC 2012*, pages 595–608. Springer, 2012.

SW05. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.

Tak14. Katsuyuki Takashima. Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In *International Conference on Security and Cryptography for Networks–SCN 2014*, pages 298–317. Springer, 2014.

Vad02. Salil P Vadhan. Randomness extractors and their many guises. In *Foundations of Computer Science–FOCS 2002*, page 9. IEEE, 2002.

Wat05. Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005*, pages 114–127. Springer, 2005.

Wat11. Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *International Workshop on Public Key Cryptography–PKC 2011*, pages 53–70. Springer, 2011.

Wee16. Hoeteck Wee. Déjà q: Encore! un petit ibe. In *Theory of Cryptography Conference–TCC 2016*, pages 237–258. Springer, 2016.

YAHK14. Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In *International Workshop on Public Key Cryptography–PKC 2014*, pages 275–292. Springer, 2014.